Cisco Service Portal Designer Guide

Release 9.4
July, 2012

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

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

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

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

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

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco's trademarks can be found at www.cisco.com/go/trademarks. Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

Cisco Service Portal Designer Guide
© 2012 Cisco Systems, Inc. All rights reserved.

CONTENTS

About this Guide xiii

CHAPTER 1

Service Designer 1-1

Overview 1-1

Service Designer Components 1-2

Preparing to Design Services 1-2

What is a Perfect Service? 1-4

Key Terms 1-4

What is in this Chapter? 1-5

What is not in this Chapter? 1-6

Creating and Managing Service Groups 1-6

Viewing and Searching for Service Groups 1-7

General Information about a Service Group 1-7

Service Group Authorizations and Reviews 1-9

Service Group Permissions 1-11

Deleting a Service Group 1-12

Creating and Managing Services 1-12

General Information about a Service 1-12

Viewing and Searching for Services 1-16

Copying a Service 1-17

Exporting and Importing a Service 1-17

Deleting a Service 1-19

Configuring the Offer Tab 1-19

Service Bundles 1-19

Prerequisites and Recommended Accessories 1-20

Pricing a Service 1-21

Formatting the Service in the Catalog 1-24

Configuring the Delivery Plan 1-28

Core Concepts 1-28

Working in the Plan Tab 1-28

General Subtab 1-33

Graphical Workflow Designer 1-42

Task Participants 1-49

Email 1-52

Task Instructions 1-53

Checklists 1-54
Authorizations 1-54
Permission to Order a Service 1-59
Creating a Scheduled Start Task 1-60
Bundling Services 1-62
Creating a Bundle 1-63
Preventing Bundling 1-64
Reviewing Included Tasks 1-64
Reviewing Included Participants 1-65
Pricing Bundles 1-66
Discounting the Price of a Bundle 1-66
Customer View of Bundles 1-67
Can a customer cancel a bundle once it has been ordered? 1-67
Exporting and Importing Bundles 1-67
Data for Bundled Services 1-67
Namespace Variables for Bundled Services 1-68
Managing Dictionaries 1-68
Commonly Asked Questions about Dictionaries 1-68
Using Active Form Components in a Service 1-71
Adding Forms to a Service 1-71
The Effect of Changing Form Components on Services 1-75
Managing Categories 1-75
Defining Categories 1-76
Configuring a Category 1-77
Defining the Appearance of Categories 1-78
Deleting a Category 1-79
Removing Categories, Subcategories, and Services 1-79
Managing Keywords 1-80
Adding a New Keyword 1-80
Associating Keywords with Services or Service Offerings 1-81
Removing Keywords from Services or Service Offerings 1-82
Deleting Keywords 1-82
Select Person/Queue Dialog Box 1-82
HTML Editor Tools Summary 1-84

CHAPTER 2

Active Form Components 2-1

Overview 2-1

Overview of a Service Form 2-1

Common Uses of Active Form Components 2-2

Lightweight Namespaces 2-2

Who is the Audience for this Chapter 2-3

What is Covered in this Chapter 2-3

What is not Covered in this Chapter 2-4

Service Form Framework 2-4

Overview 2-4

Dictionaries 2-4

Active Form Components – Forms 2-12

Access Control 2-27

Active Form Rules 2-31

Active Form Behavior 2-75

Service Bundles 2-77

Service Form Performance and Security Considerations 2-77

Sending Minimal Data to the Browser 2-77

Using the Pre-Load and On-Load Events 2-78

Comparing the Pre-Load and Post-Submit Events 2-79

Using ISF 2-79

Using the "Editable on server-side only" Check Box 2-79

Using Distributing vs. Validating Data Retrieval Rules 2-80

Performance of Data Retrieval Rules 2-81

ISF Application Programming Interface (API) 2-82

What is ISF? 2-82

When Should You Use ISF and JavaScript? 2-82

ISF Components 2-83

Integrating ISF Code into Service Forms 2-92

JavaScripts 2-93

Libraries 2-96

Adding JavaScript Functions to a Form 2-97

Associated Controls (Buttons and Links) 2-103

ISF Coding and Best Practices 2-104

JavaScript/ISF Development Environment 2-104

Architecture/Storing ISF Scripts 2-105

Recommended Naming and Coding Standards 2-106

Writing the Code 2-107

Testing 2-110

Design Guidelines 2-110
Best Practices for Using Active Form Components 2-111
Naming Conventions 2-111
Form-Dictionary Relationship and Form Granularity 2-112
Rules, ISF and the Requisition Life Cycle 2-114
Changing a Dictionary or Active Form Component 2-115
Coding SQL Entry Data Retrieval Rules 2-115
Using Customer and Initiator Lightweight Namespaces 2-115
Putting It Together 2-116
Overview 2-116
Use Case Analysis 2-116
Detailed Design 2-117
Scenario #1: Dynamically Adjusting Form Appearance and Behavior 2-117
Scenario #2: Manipulating Customer and Initiator Information 2-119
Scenario #3: Securing Sensitive Data 2-121
Scenario #4: Computing a Value in a Form 2-122
Formatting 2-123
Server-Side Associated Controls 2-125

CHAPTER 3 **Lifecycle Center** 3-1
Overview 3-1
The Case for Service Item Lifecycle Management 3-1
Designing Service Item-Aware Services 3-2
The End User View of Service Items 3-3
Service Items and Service Item Manager 3-4
The Service Item Manager Taxonomy 3-4
Managing the Service Item Manager Screens 3-5
Designing Service Items 3-6
Managing Service Items 3-15
Defining Standards 3-19
Managing Standards 3-21
Configuring Service Item Dictionaries 3-22
Defining Service Item-Based Dictionaries 3-22
Specifying Fields in Virtual Machine Dictionaries 3-27
Configuring Active Form Components 3-27
Service Item-Based Dictionary Display Properties 3-27
Using Service Items in Data Retrieval Rules 3-28
Using Standards in Data Retrieval Rules 3-29
Configuring the Delivery Plan 3-29

Configuring an Internal Service Item Task 3-30
Configuring an External Service Item Task 3-33
Configuring a VMware Operation 3-33
Inbound Responses from vCenter Server 3-42
Miscellaneous Considerations 3-42
An End User's View of Service Items 3-44
Viewing My Service Items 3-44
User Preference for My Items Portlet 3-46
Importing Service Items and Standards 3-47
Overview 3-47
Importing Virtual Machines and Standards from vCenter 3-47
Importing Service Items and Standards from File 3-48
Importing Service Items and Standards using Service Link 3-64
Service Item Import DTD 3-65
Best Practices 3-66
What is a Service Item? 3-66
Data Integration Options with Lifecycle Management 3-66
Using SIBDs Rather than Order-on-Behalf 3-67
Using Service Items for Capacity Management 3-68
Administration 3-68
Overview 3-68
Roles for the Service Item Manager Module 3-69
Configuring Access to My Service Items 3-69
Database Administration 3-71
VMware Adapter Error Messages 3-73

CHAPTER 4 **Portal Manager** 4-1

Overview 4-1
Portal Manager Roles and Capabilities 4-2
Prerequisites 4-2
Portal Designer 4-3
Portlet Taxonomy 4-3
Managing the Portal Designer Screens 4-3
Content Portlets 4-6
Defining a Content Portlet 4-6
Portlet Content 4-6
Portlet View 4-7
Portlet Filter 4-10
Portlet Permissions 4-11

Reserved Portlets 4-12
Search Portlet 4-12
Order Status Portlet 4-12
Approvals Portlet 4-14
Custom Content 4-15
Creating Custom Content Table 4-15
HTML and JavaScript Portlets 4-17
HTML Portlets 4-17
JavaScript Portlets 4-18
JSR Portlets 4-21
Overview 4-21
Deploying JSR Portlets 4-21
Managing JSR Portlets 4-24
Portal Pages 4-25
Creating a Portal Page 4-25
Portal Page General Information 4-26
Portal Page Content 4-28
Portal Page Permissions 4-30
Subscribed Users 4-30
Site Homepage 4-30
Portal Settings 4-31
Common Settings 4-31
Organizational Unit Settings 4-32
Keywords 4-32
Authentication Settings 4-32
Reference Data 4-36
Content Definition 4-37
Core Entities 4-37
HTML/JavaScripts 4-45
Service Items 4-45
Standards 4-46
An End User's View of the Portal 4-46
Overview 4-46
Portal Modules 4-46
Portal Home Pages 4-47
View Mode of a Portal Page 4-47
Edit Mode of a Portal Page 4-49
Adding/Creating a Portal Page 4-51
Importing and Exporting Portal Content 4-52

Exporting Portal Content 4-52
Importing Portal Content 4-54
Portal Access Control 4-56
Overview 4-56
Roles for the Portal Designers 4-56
Roles for the Portal End Users 4-57

CHAPTER 5 **Catalog Deployer 5-1**
Overview 5-1
Overview 5-1
Catalog Deployer Operation 5-1
Catalog Deployer Features and Functionality 5-2
Catalog Deployer Usage 5-2
Terminology 5-3
Additional Resources 5-4
Catalog Deployer and Configuration Management 5-4
Overview 5-4
Typical Configuration Management 5-4
Configuration Management 5-5
Catalog Deployer Architecture 5-6
Catalog Deployer Capabilities 5-6
Entities Supported by Catalog Deployer 5-8
Configuring Catalog Deployer 5-9
Overview 5-9
Prerequisites 5-9
Overview of Configuring Implementations and Sites 5-10
Instructions for Configuring Implementations and Sites 5-11
Configure All Sites 5-15
Who Uses Configuration Management Tools? 5-16
Application Roles and Capabilities 5-16
Catalog Deployer Performance Considerations 5-17
Running Catalog Deployer 5-19
Overview 5-19
Using Catalog Deployer 5-19
Hiding/Adjusting the View and Search Pane 5-20
Searching Deployment Packages 5-22
Creating and Deploying a Deployment Package 5-23
Deployment Packages in Detail 5-34
Sample Deployment Scenarios 5-39

Overview 5-39
Initial Deployment 5-39
Where Should Entities be Homed? 5-40
Deploying a Service which needs a New Queue 5-42
Deploying Services that use a new Email Template 5-43
Renaming a Queue and Service Team 5-43
Changing a Category and its Icon 5-44
Renaming Entities after a Service Portal Upgrade 5-45
Adding a Custom Functional Position 5-45
Deploying to an Environment with Browser Cache Enabled 5-46
Branded Content Libraries 5-46
Overview 5-46
Deploying a Branded Library 5-46

CHAPTER 6 **Namespace** 6-1
Overview 6-1
About Namespaces 6-1
Namespace Definition 6-1
References 6-2
Nodes 6-2
Expressions 6-3
Configuring an Expression 6-3
CN (Common Name) Assignments 6-4
ID (Identifier) Assignments 6-4
LOGINNAME Assignments 6-4
QUEUE Assignments 6-5
Email Templates 6-5
Defining an Email Template 6-5
Namespace Usage in Email Templates 6-7
Recipients 6-7
Subject 6-8
Body 6-8
Service Manager Task Details URL 6-10
Namespace Processing and Alternate Values 6-10
Demand Center Templates 6-10
Authorizations, Reviews and Delivery Tasks 6-12
Authorizations and Reviews 6-12
Namespace Usage for Authorizations and Reviews 6-12
Delivery Plans and Tasks 6-14

Namespace Usage for Delivery Tasks 6-15
Conditional Statements 6-17
Namespace Reference 6-23
Namespace Objects and their Relationships 6-23
Email Namespace Elements 6-24
Conditional Namespace Elements 6-26
Organizational Unit-Based Namespaces 6-27
Person-Based Namespaces 6-27
Lightweight Namespaces 6-33
Process Namespaces 6-38
Requisition Namespaces 6-38
Message Namespaces 6-38
Demand Center Namespaces 6-39

INDEX

About this Guide

Objectives

The *Cisco Service Portal Designer Guide* explains the complete process of designing services in your Cisco Service Portal (Service Portal) service catalog and making them available for ordering. It describes each component of service design and the considerations you need to make when creating them.

The main module covered by this guide is Service Designer. Because service design requires a wide range of both business and technical skills, the Service Designer module provides features addressing the entire range—from presenting each service in an appealing way to coding JavaScript functions that handle complex data validations. However, service designers are not expected to be programmers. This guide covers the many techniques that non-technical designers can leverage to create powerful and dynamic service requests. It also presents the core JavaScript functions available for manipulating the appearance and behavior of service forms.

Audience

This guide is intended for the service designers responsible for the service catalog. Although it does not presume that everyone reading this book is a programmer, it does expect that you are able to supplement the information provided here with a basic understanding of SQL and JavaScript if necessary.

Document Organization

The *Cisco Service Portal Designer Guide* is divided into the following six chapters:

- Chapter 1, "Service Designer": This chapter concentrates on the Services component of Service Designer, the main tool for configuring a service definition which is part of your service catalog.
- Chapter 2, "Active Form Components": This chapter includes guidelines and instructions for using active form components to customize service forms to your requirements.
- Chapter 3, "Lifecycle Center": This chapter explains how to configure services to manage the lifecycle of a service item.
- Chapter 4, "Portal Manager": This chapter contains instructions for configuring your databases for use with Service Portal.
- Chapter 5, "Catalog Deployer": This chapter describes Catalog Deployer—a content deployment and configuration management tool.

- Chapter 6, "Namespace": This chapter describes "Business Engine" namespaces that are used in email templates, tasks, and task plans.

Conventions

This document uses the following conventions:

Convention	Indication
bold font	Commands and keywords and user-entered text appear in **bold** font.
italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
[ ]	Elements in square brackets are optional.
{x | y | z }	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
< >	Nonprinting characters such as passwords are in angle brackets.
[ ]	Default responses to system prompts are in square brackets.
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.
Choose **Menu item > Submenu item** from the X menu.	Selections from a menu path use this format. For example: Choose **Import > Formats** from the File menu.

Note Means *reader take note*.

Tip Means *the following information will help you solve a problem*.

Caution Means *reader be careful*. In this situation, you might perform an action that could result in equipment damage or loss of data.

Timesaver Means *the described action saves time*. You can save time by performing the action described in the paragraph.

Warning **Means *reader be warned*. In this situation, you might perform an action that could result in bodily injury.**

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as an RSS feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service. Cisco currently supports RSS Version 2.0.

CHAPTER 1

Service Designer

- Overview, page 1-1
- Creating and Managing Service Groups, page 1-6
- Creating and Managing Services, page 1-12
- Configuring the Delivery Plan, page 1-28
- Bundling Services, page 1-62
- Managing Dictionaries, page 1-68
- Using Active Form Components in a Service, page 1-71
- Managing Categories, page 1-75
- Managing Keywords, page 1-80
- Select Person/Queue Dialog Box, page 1-82
- HTML Editor Tools Summary, page 1-84

Overview

Service Designer enables you to design and package services as products, and to catalog these services for end users to browse through and order.

Service Designer is used to:

- Construct service fulfillment plans
- Configure service ordering permissions
- Configure authorization flows from a service delivery perspective
- Design the look and behavior of service forms
- Create categories and keywords customers may use to search for a particular service
- Link email templates with processes that require email notifications

Catalog designers use Service Designer to build and manage *service forms*. A service form is an interactive web page through which service requisitions are entered and tracked in Request Center. The basic building block of a service form is a *dictionary*, a group of individual data elements (fields) that allow users and service performers to enter and view data required to fulfill the service request. The appearance and behavior of a service form is determined by how the dictionaries and their component fields are configured as part of the *active form components* that are used in the service definition.

Active form components provide the potential for *reusability* across service forms. With careful and thoughtful design, a designer may create an active form component from a commonly used dictionary, or set of dictionaries, and configure them only once. Then this form component can be included in as many services as necessary, with no additional configuration.

One or more dictionaries can be included in an active form component, and one or more form components can be included in a service. A form component may also contain no dictionaries, acting only to hold form rules, depending on your design. In a larger project, it may be useful to have one or two designers focus on the creation of active form components, allowing service designers to be "consumers" of the reusable form components as they define services.

Service Designer Components

The Service Designer module is comprised of these major components:

Component	Use this component to:
Services	Create and modify service groups and service definitions, including the delivery (fulfillment) plan and presentation of the service, as well as the active form components used by the service form.
Dictionaries	Create and modify the dictionaries that specify the data fields required in a service.
Active Form Components	Create and modify reusable form components which specify both the service's look (via the configuration of previously defined dictionaries) and feel, via the definition of rules which can dynamically adjust both the form's appearance and behavior.
Scripts	Write JavaScript functions to supplement the rules defined in active form components and maintain JavaScript libraries.
Categories	Specify how services and service categories are displayed in My Services and My Services Executive.
Keywords	Define and manage the keywords used in the service catalog search engine.
Objectives	Define and manage the measurable service delivery objectives that are associated with service offerings defined via Demand Center.

All of the above components may not be visible to all users of Service Designer. The components you see when you choose the Service Designer module correspond with the role you were granted in Organization Designer.

Preparing to Design Services

Service design is a complex task, made easier by thoughtful planning.

Designing services in Service Designer is easier and faster if you have collected the information you need beforehand. Collect as much information as possible by interviewing members of your organization or the organization you are designing for, reading reviews of software tools already used within the company, and gathering paper forms and internal data sources currently used to manage services.

Then complete the following preparatory work before using Service Designer:

Step 1 Gather information about the services you offer.

Before you start to implement the service catalog, do you know:

- What services you will provide?
- Who provides these services?
- Who consumes these services?
- What approvals or reviews are needed before users can purchase services?
- Who is authorized or required to approve or review service requisitions?
- What policies (global or local) govern these approvals and reviews?
- What service levels govern the delivery of these services?
- What steps are followed in the normal course of service delivery?
- Who performs these delivery activities?
- Who manages the service delivery?
- What happens if a service delivery is late or interrupted?
- What information must the consumer supply in order to purchase a service?
- What information the service delivery team needs from the consumer?
- Are there any existing data sources that you need to include?
- Are there any existing forms that you can use as models for order forms?
- What assets or items will the services deliver? Will you want to track these items after the services have delivered them?

Step 2 Begin by working in Organization Designer and Administration.

Before creating services in Service Designer, it is important to complete the following work in Organization Designer:

- Create organizational units for your service consumers and service delivery teams.
- Create people and work queues to staff these organizational units.
- Create functional positions for your consumer business units and service teams, and assign people to them. It is particularly important to have an assigned manager for both business units and service teams, because the system uses this position in certain built-in pricing processes.
- Configure user roles to grant the appropriate capabilities and permissions to each organization or user.

In the Administration module you should:

- Create email templates, which are standard emails that are created and configured to be sent to users during key system moments, such as when a task needs an authorization, escalation, or once service delivery is complete. These email templates will later be associated with services in Service Designer.

Step 3 Create Service Items in Service Item Manager (Optional).

The Service Item Manager module (available only to customers who license Lifecycle Center) allows designers to designate certain items to be “service items”, whose history can be tracked within Request Center. A service item definition can provide a template for a dictionary to be used in a service.

Step 4 Create Dictionaries and Active Form Components in Service Designer.

- Dictionaries contain the fields used in various active, or reusable, form components.

- Form components are comprised of one or more dictionaries; access control rules to specify usage of each dictionary; display properties for rendering the dictionaries and their fields on service forms; and the behavioral rules associated with each dictionary and potentially individual fields in a particular dictionary. Each active form component is configured once and added to one or more service forms.

Step 5 Create Categories, Keywords, and Service Groups in Service Designer.

Ideally, service groups, categories and keywords should be specified in conjunction with creating a service. Although possible, it is less efficient to create categories and keywords and to associate them with services *after* the services are created.

What is a Perfect Service?

The Service Designer module allows you to construct the "perfect service." But what does this mean?

A "perfect" service is a service that is so well-defined that the end user understands everything they need to know:

- What am I ordering in this service?
- What is included?
- How much does it cost?
- Do I need this service? (or, Is this the right service for me?)
- How do I order this service?
- How long will it take to be delivered?

The service may also include more detailed information such as:

- Additional task instructions to the service delivery provider
- A checklist of sequential tasks that must be completed in order to fulfill the service request
- Safeguards for when services are delivered late, using conditional statements

A perfect service should clearly communicate what the service is, and what the expectations surrounding service delivery are at each stage of the process. End users should be able to rely on the information they see in My Services. Likewise, the service definition and expectations should be equally clear to the service team who will receive the service request and deliver the service.

The more intuitive the process, the less likely it is that end users will have to call the fulfillment or service delivery team with questions.

Key Terms

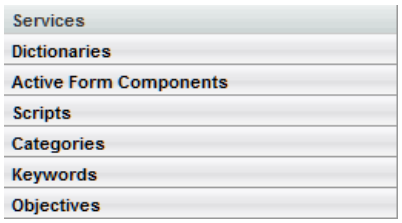
The following are some key terms to be familiar with when using Service Designer.

Term	Definition
Active Form Components	Reusable forms that are built from one or more dictionaries and configured for use in one or more service forms. Active form components are the building blocks of a service form, and dictionaries, along with active form rules, are the building blocks of a form component.
Authorization	A task during which the performer reviews, approves, or rejects the service requested.

Term	Definition
Customer	The individual to whom a service is being delivered. The customer and initiator can be the same person.
Dictionary	Reusable groups of fields created for use on a form component that may, in turn, be used in multiple service forms. A dictionary defines the individual data items that are used in a service request.
Email Template	A standard email that can be sent upon the initiation, completion or other milestone associated with of a particular task. Email templates can be associated with particular services.
Escalation	Notifications triggered at specified intervals after a task is not completed by its due date.
Initiator	The requestor of, or person who orders, a service from the service catalog. The customer and initiator can be the same person.
Interactive Service Forms (ISF)	A JavaScript API that allows designers to customize the behavior of a service form using JavaScript. ISF coding supplements the use of active form rules to add further interactivity and a richer user interface to the service form.
Moments	Request Center manages the events from Ordering through Service Completion as a sequence of discrete system moments or phases. The completion of one moment is the prerequisite for the beginning of the next moment in the sequence.
Service	A process packaged and presented as a product the end user can order/request.
Service Link	The module that defines integrations with external systems; such integrations can be used within a delivery plan as external tasks, reviews, or authorizations.
Service Group	A folder that contains a group of similar services. Services are organized into service groups as a way to facilitate the service design process.
Service Item	A product or intangible asset that can be provisioned via a service request and whose history can be tracked in the My Services and Service Item Manager modules.
Service Team	The individuals (or groups of individuals) who perform the steps to deliver the service.

What is in this Chapter?

The seven components of Service Designer mentioned earlier are shown below:



This chapter concentrates on the Services component, the main tool for configuring a service definition which are part of your service catalog. It briefly discusses Categories and Keywords which are simple-to-use components that allow service designers to cross-reference services with search words for use in the My Services and My Services Executive modules.

What is not in this Chapter?

Configuring dictionaries and active form components is an activity that typically precedes and is a prerequisite for defining a service. Details on defining and using both dictionaries and active form components are given in Chapter 2, "Active Form Components".

Scripts are JavaScript functions and libraries that can supplement the active form component rules in dynamically changing the appearance or behavior of a service form. Details on defining and using JavaScript functions and libraries, and Interactive Service Forms (ISF), a JavaScript API for interacting with dictionaries and fields, are given in Chapter 2, "Active Form Components".

Namespace is a term used to describe a set of valid names that address the data objects used within Request Center, exposing these objects to service designers. This allows designers to use these elements in an email, to dynamically resolve the recipient, subject, or references within the email body; in workflow, to conditionally execute reviews, authorizations, or tasks in a delivery plan; in expressions which determine the person or queue to which an authorization or delivery task is assigned; and in attributes which configure tasks and authorizations. A complete list of the namespaces available and a detailed explanation of their usage is included in Chapter 6, "Namespace".

A service's fulfillment plan may include authorizations and delivery tasks that are performed within the My Services Authorizations and Service Manager modules, as well as tasks that must be delegated to an external system. For example, a user might enter a request in Request Center that must be handled by a Help Desk or other system. The integration module, Service Link, handles the configuration of requests that must communicate with an external system. Details are given in the *Cisco Service Portal Integration Guide*.

A robust Request Center installation needs access to information about the organization, its structure, and people in order for service requests to be available to the right community of people and for fulfillment plans to be routed to the appropriate organizations and personnel. Setting up foundational information about the organization must typically precede service development; such prerequisite activities are explained in the *Cisco Service Portal Configuration Guide*.

Objectives can be defined for a service, but these are not used by Request Center. Rather, they are useful for portfolio designers who specify component services for their portfolios. Details on objectives are available in the *Service Designer Online Help*.

Request Center can be used to create and track service items, corporate assets that can be uniquely identified and which can be provisioned, modified, or removed from use via a service request. A service item may influence the design of a dictionary as well as the service's task plan. Details on configuring service items and integrating them into service definitions are given in Chapter 3, "Lifecycle Center".

Creating and Managing Service Groups

A service group is a folder that you create in Service Designer to organize a set of similar or related services "owned" by a particular service team.

Service group folders are different from the categories that end users browse through in My Services. In fact, customers never see the name of the service group to which a given service belongs.

Service groups enable service designers to:

- Configure authorization and escalation processes for services in the group
- Configure permission to order services in the group
- Assign functional positions that are used by the group

For example, you might create a “Telephone Services” service group containing the services for acquiring and setting up telephone service for your organization. This group would have a service team assigned to deliver the services and may have other people responsible for the management of the Telephone Services group itself. The service group specifies who can order cellular phones and has an authorization process in place once a phone has been ordered. The Telephone Services group may also have a series of escalation messages configured to alert people when they are late authorizing or reviewing a service. By configuring authorization processes, ordering permissions, and escalation notifications at the service group level, you avoid having to configure each individual service.

Viewing and Searching for Service Groups

Service Groups are created and configured in the Services component of Service Designer. All service groups are listed alphabetically in the Services panel at the left side of the page. You can:

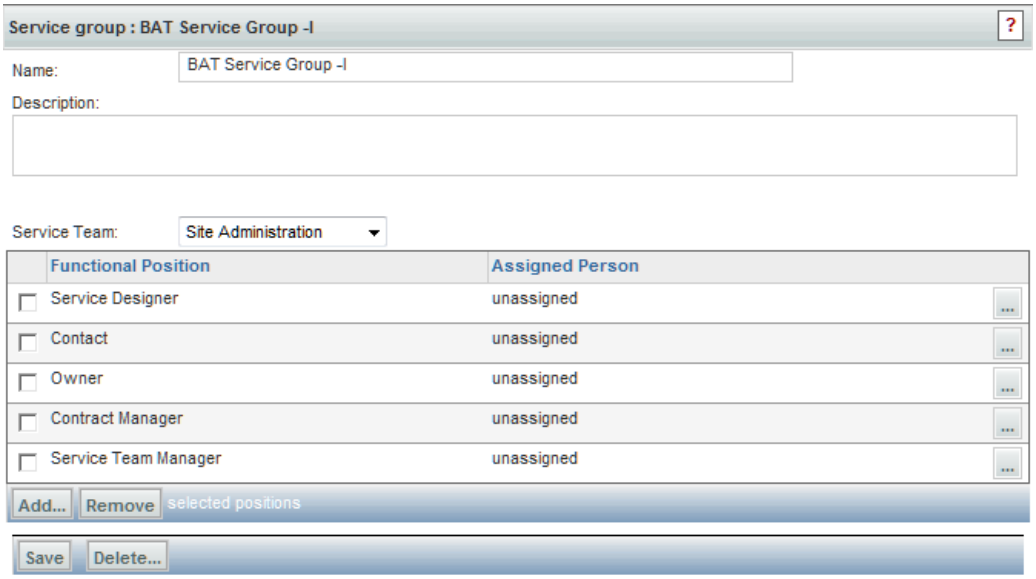
- Choose a service group name to open the details and configuration tabs for that service group, or
- Expand the **+** icon next to the service group name to see what services “live” within the group.

To add a service group, click **New** at the top of the Services panel, and choose **New Service Group**.

General Information about a Service Group

Configuring a service group involves entering the information summarized in the table below.

Field	Definition
Name	The **name** for the service group. Required. The name should be specific to your organization and needs. End users do not see this name, and the name is editable after service group creation. Sample names include “End User IT Desktop Support,” “End User IT Desktop Software,” or “Identity Management”.
Description	A brief **description** for the service group; optional but recommended.
Service Team	The **Service Team** that is responsible for services in this service group, that is, the team that “owns” the services in the service group.
Functional Positions	Functional positions that have been associated with service groups, and the member of the specified service team who is currently assigned to that position.



Service Team

Service Teams are organizational units created and managed in the Organization Designer module.

It is critical to specify the appropriate service team for a service group. “Service Team” is listed as a default participant when configuring dictionary Access Control. This allows members of the service team to view or edit dictionaries in the service delivery moment of service fulfillment. All members of the service team are automatically able to perform work on all tasks defined in services in their service group. Other service teams need to be listed as “Additional Participants” in the Access Control subtab for the form components used in the service in order to perform tasks in the service.

If you change the service team assigned to a service group:

- You may need to reconfigure the “Access Control” specified for active form components used in services in this group, to explicitly add the queue associated with the previously specified service team to the list of Additional Participants. This is required if any tasks have been assigned to that queue.
- All person assignments to functional positions are removed. This is because only members of the service team that owns the service group can fill functional positions in that group. Any such assignments need to be respecified.

Functional Positions

A functional position is a job description associated with one of the following:

- Organizational Unit
- Service
- Service Group

In Service Designer, you may assign functional positions to be performers of activities in the authorization, review, and delivery processes, to avoid referring directly to people or queues. You may also use functional positions to identify the recipients of escalation notifications. For example, an email

may be directed to the "Escalation Manager" for a particular organization or service group, rather than being routed to a specific person or queue. Or you may simply use some of the functional positions to document the person or other entity responsible for a particular service group or service.

To add a functional position to a service group:

Step 1 Click **Add**.

The Select Functional Positions dialog box appears.

Note Functional positions added here must already exist in the system. They must be set up in Organization Designer to be available for selection.

Step 2 Check the check box next to the positions you want to add.

Step 3 Click **Add**.

Now you are ready to assign people to the functional positions for the service group. Functional positions are staffed from the associated service team.

Assign Functional Positions to People within the Service Group

To assign a person to a functional position:

Step 1 Click the "**…**" button in the row of the functional position to bring up the "Select a Person" dialog box.

The dialog box limits your selection of people to those who are members of the service team associated with the service group. People are assigned to service teams in Organization Designer.

Step 2 If you cannot readily find the desired person, you may want to use the search function. Enter valid identifying information in the top text field, or an asterisk (*) to search for all available people, and click **Search**.

Step 3 Click the radio button next to the person or queue name to choose them.

Step 4 Click **OK**.

The person's name displays in the Assigned Person column.

Service Group Authorizations and Reviews

Use the Authorizations tab for a service group to configure authorization and review tasks, and the specified order in which they occur, for services within that group. You can also customize the escalation tiers, recipients, and email messages for late authorization/review tasks for services in the service group.

Authorization and review structures can be set up at either the service group level, so that the structure applies to all services contained in the service group, or at the individual service level.

If you set up an authorization structure at the service group level, you need to choose the "Use service group authorization structure only" setting on the Authorizations tab *of each service* within the service group for which you want to use this structure. The benefit of this is that you do not need to repeatedly set up all the details for services that share the same authorization structure.

The term “authorization” is sometimes used to refer to a process involving both authorizations and reviews.

Authorization Structure

For each service group you can choose from three options:

- **Use service group authorization structure only** (the default): Uses only the site-wide authorization structure. See the Site Administration chapter of the *Cisco Service Portal Configuration Guide*. Authorizations and reviews configured at the service group or service level are ignored.
- **Use service-level authorization structure only** (will not use service group-level): Allows you to develop an authorization structure based on a unique scheme of roles, order, or escalations.
- **Use both service group-level and service-level authorization structures**: Accepts the site-wide scheme of roles, order, or escalations, and allows you to supplement it with a customized scheme that you establish to enhance the authorization process.

Authorization Types

There are two Service Group authorization types:

- **Authorizations** give the approver the opportunity to determine if the person requesting the service is eligible to receive it. If an authorization is rejected, the process stops and the service is not delivered.
- **Reviews** are for information only. A reviewer cannot reject a service or cancel the delivery process. However, the delivery process does not proceed until the reviewer clicks OK in the review task.

Another major difference between authorizations and reviews is that authorizations are performed sequentially, while reviews are performed concurrently. It is possible to have several service group authorizations or reviews, each performed by a different person. The authorizations are performed in sequence, in the order specified; if an earlier authorizer rejects the service request, no further work on the service request is done. Since reviews cannot cancel the delivery of a service, they are performed concurrently, to expedite the delivery process.

If you see the message “not currently enabled via the Administration module,” then the particular service group authorization/review is not enabled for your site. This may reflect decisions made by the site administrator regarding site-wide standards. This setting can be changed in the Administration module.

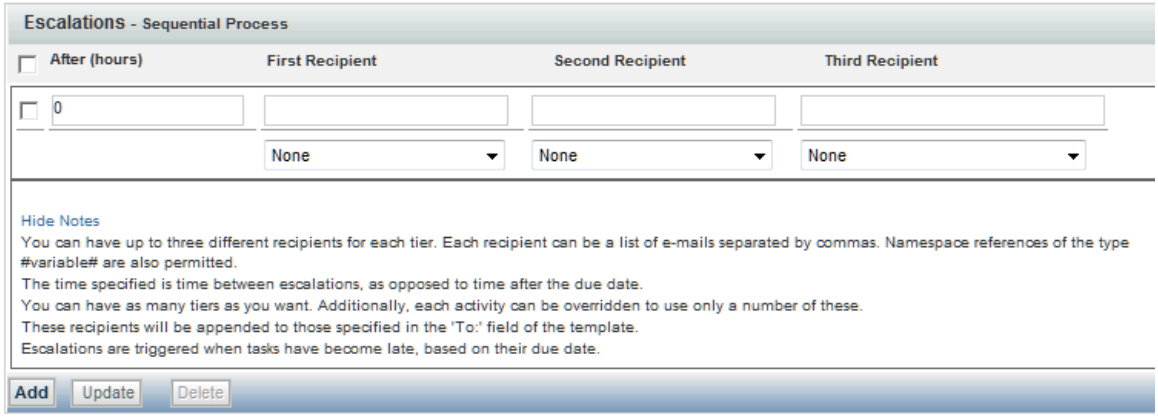
Configuring Service Group Authorizations

The configuration details that must be supplied for service group authorizations are identical to the details required to configure service-level authorizations. These, in turn, are a subset of the configuration options available for configuring delivery plan tasks. Please see the “Configuring the Delivery Plan” section on page 1-28 for details on configuring service group authorizations.

Escalations

Escalations specify the notifications that must be sent to interested parties (such as requestors, supervisors or task performers) when an authorization, review, or delivery task is late, and the schedule according to which a set of notifications should be delivered. Escalations are arranged in “tiers”. For example, the first tier of an escalation could notify the service performer when a task assigned to that

performer is one hour late. The second tier could notify the supervisor of the service team responsible for the task when the task is an additional 8 hours later than when the first tier of the escalation went into effect.



The person who receives the escalation does not have permission to go into the service and approve it or perform the task which generated the escalation. Escalations are only notifications, and do not serve to transfer the ownership of nor modify the current performer of an approval or delivery task.

The Escalation Manager is the Request Center software component responsible for sending out email notifications when an escalation is triggered. By default, the Escalation Manager is configured to check for late tasks once an hour during a standard Monday through Friday work-week where the work day runs from 8am through 9pm. To have escalation notices delivered over the weekend, or in a global installation where the concepts of “work-week” and “weekend” are elastic, you can ask the System Administrator to change the Escalation Manager so that it runs more frequently or on additional days. Details on this procedure are given in the *Cisco Service Portal Configuration Guide*.

Service Group Permissions

Use the Permissions tab for a service group to set up service group object-level permissions. These object-level permissions are types of actions that people, organizational units, groups, functional positions, and roles can perform. The table below defines the types of permissions you can grant.

Permission to:	Definition
Design services and change data in this service group	User can design services in this service group and change group data. These rights are typically reserved for the individuals who design and configure services in this service group.
View services and other information in this service group	User can view the service group and service definitions, but is unable to change them.
Order services in this service group	User can order services in this service group. This is how you allow service consumers to see and request these services.
Assign rights to people	User can assign these permissions to other people. This right is typically restricted to the service designer and owner of the service group.

Being able to design services in a service group automatically allows the user to view services in that group; however, the ability to order services in the group must be independently assigned.

The ability to order a service can also be assigned at the service level. Service-level permissions supplement, they do not override, permissions assigned at the service group-level. Therefore:

- If the same set of users has permission to order all services in the service group, assign the ordering permission at the service group level. This saves time and eases maintenance.
- If different sets of users have permission to order different services in the group, assign only users for all services at the service group level and assign service-specific users at the service level.
- Rather than assigning permissions to individual people, it is much more efficient to assign the permissions to an organizational unit. All members of the organizational unit inherit the permission. If a set of permissions must be assigned to a disparate set of people, you can place these people in a group, or make them members of a role, and assign the permission to that group or role. Membership in a single group or role is much easier to maintain than changing the permissions assigned to numerous people.

Deleting a Service Group

You can delete a service group if you no longer need it. You cannot delete a service group in which services still exist. If services do exist, delete or transfer them to other service groups before deleting the group.

Creating and Managing Services

Once you have set up a service group, you can create services in that group.

To create a new service definition:

Step 1 Within the Services component, choose **New > New Service**.

Step 2 In the Name field, type a name for the service. Maximum field length is 200 characters.

Step 3 In the Description field, type a brief summary of the service. This summary should include all necessary information the end user needs to know, or might ask, about the service. Maximum field length is 4000 characters.

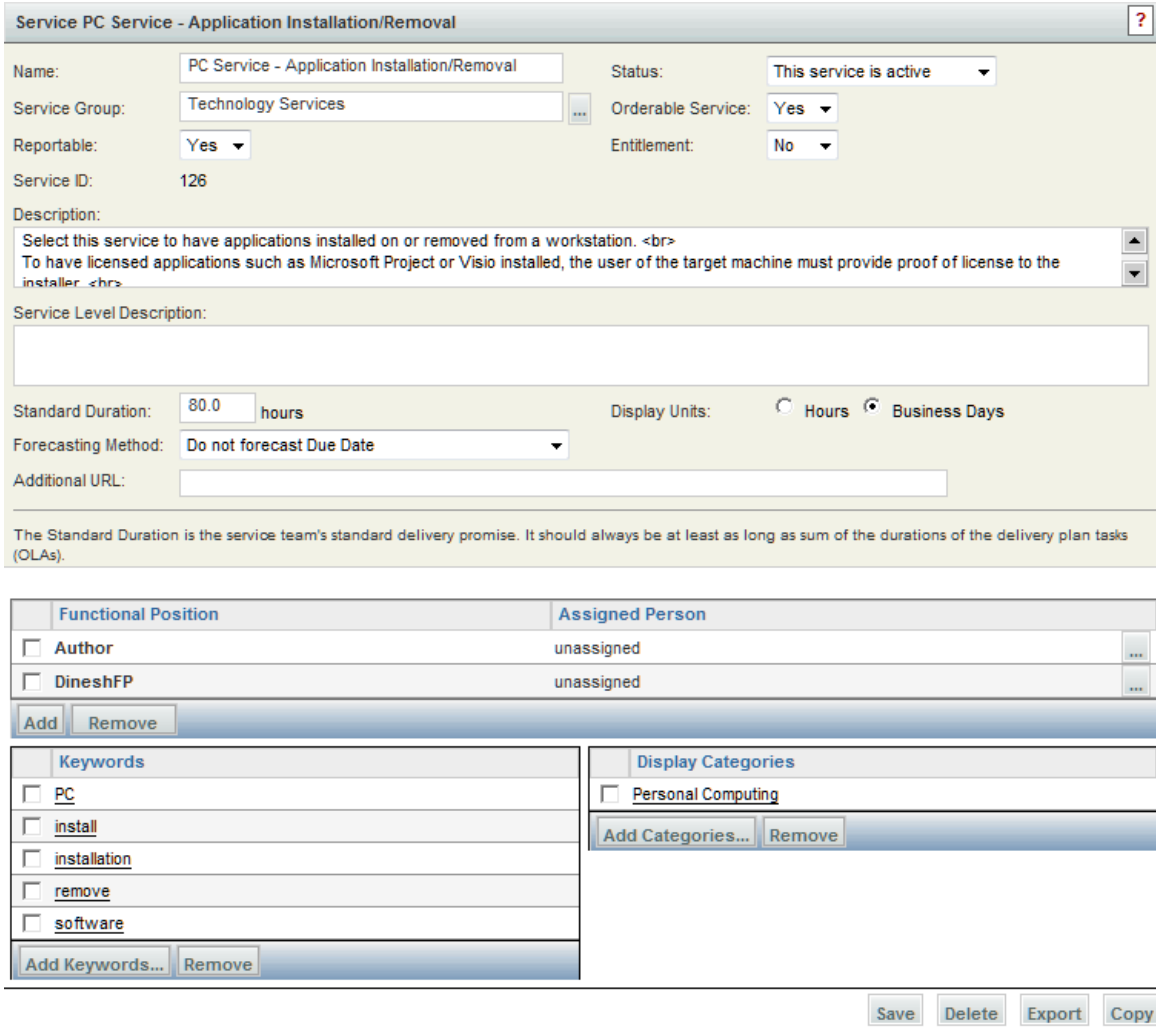
Step 4 Enter a URL if you want to further describe the service or link to supporting information. The service description in My Services will display a “More information” link to this URL.

Step 5 Click **Add This Service**.

General Information about a Service

After you create a new service, you begin to configure it by entering information on the General tab.

The General tab is primarily used to influence the consumer experience as the end user browses the service catalog. You use the General tab to set the context for the service by including descriptive information, as well as to configure the important attributes and settings that affect reporting and display behavior of a service.



To begin configuration, you can start anywhere on the page. Use the table below as a reference for completing the fields.

Field	Definition
Name	The name of the service. This is the name the end user will see in the service catalog. The service name should not exceed 200 characters. For example: Computer Memory Upgrade
Status	A status of active means the service is searchable and available for ordering in My Services. A status of not active prevents the service from being searchable or orderable within the service catalog. A service may be inactive if it is still in draft form, has expired, or if it is a service you plan to offer at intervals, but not all the time.
Service Group	The service group to which the service belongs.

Field	Definition
Orderable Service	A setting that enables (Yes) or disables (No) the "Order" link for the service in My Services. Nonorderable services are typically created to provide customers with nonactionable information in the service catalog.
Reportable	Specify whether the service data should be loaded into the data mart so that it can be used in ad-hoc reports or queries constructed using Ad-Hoc Reports and Report Designer in the Advanced Reporting module. See the *Cisco Service Portal Reporting Guide* for additional information and best practices for making services reportable.
Entitlement	Setting this option to "Yes" allows any end user to request this service without requiring an authorization. If a site level, department level or service group level authorization is in place, setting Entitlement to "Yes" ignores those authorizations for this service.
Service ID	A read-only field that indicates the unique identifier of the service in your Request Center database. You will need this number when you use Service Migrate to promote this service from a Development environment to a Production environment.
Description	A text field for entering a succinct description of the product or service. This information is displayed to the consumer in My Services and should be as clear as possible so the end user knows exactly what they are ordering. The description may not exceed 4000 characters.
Service Level Description	This is an optional text field. Anything entered here is displayed to the consumer in My Services and should be thought of as a promise to the consumer about what level of service they should expect. The description may not exceed 4000 characters.
Standard Duration	The Standard Duration is the standard (typical) delivery time for this service after all authorizations and reviews have been completed. This information is available to the customer before they request a service. *The Standard Duration is not the due date for the service to be completed and is not used to calculate due dates*. This field does not take into consideration the calendars of the service fulfillment team. It calculates hours into days, and is used to calculate the Standard Compliance metric for the service used in Advanced Reporting.
Display Units	The Standard Duration can be displayed in Hours or Business Days. If you choose to display by business days, the number of hours is converted to days using the "Working Hours per Day" setting on the Plan tab. For example, if the standard duration for a service is 3 business days, you would enter 24 hours in the Standard Duration field, assuming an 8-hour workday.
Forecasting Method	Choose the method that will be used to forecast the service due date to the end user in My Services after the service is submitted. This forecast should always be considered approximate. No matter which forecasting method is chosen, Service Portal recalculates all due dates after all authorizations and reviews for the request have been completed. Choosing a forecasting method has both functional implications (what the requestor of the service will see, and when) and performance implications. See the "Forecasting Due Dates" section on page 1-15 for a detailed discussion.

Field	Definition
Additional URL	The Additional URL field can be used to further describe your service or link to supporting information. For example, if the service is for desktop software, you may want to include a link to an external product site so the end user can read system requirements or ensure they are ordering the correct software. The URL must be fully qualified, beginning with "http://". The service description will include a "More Information …" link, to access the specified URL.
Functional Positions	Available functional positions and their corresponding personnel assignments for the service. Functional positions are defined in Organization Designer. You can add positions and assign people to the position, for this service. See the "Functional Positions" section on page 1-8 or the *Organization Designer Online Help*.
Keywords	The search facility in My Services returns services containing any word in the service name and service description. Add additional keywords to help facilitate a user search. For example: If the service is "Order a New Laptop" and the user can choose either a Dell or a Lenovo laptop, additional keywords might be: 'Dell' and 'Lenovo'. These are used for customer navigation only, and are not used in reporting. See the "Adding a New Keyword" section on page 1-80 for more information.
Display Categories	The categories in which the service appears in My Services.

Forecasting Due Dates

The methods for forecasting due dates are summarized in the table below and explained in detail in the following paragraphs.

Method	Functionality	Performance Implications
Estimate Due Date from task durations	The system forecasts the due date based on all tasks in the delivery plan, the anticipated duration for each task, and the assigned performer.	All tasks (authorization/review and delivery) are instantiated, and individual queue or person calendars are consulted to determine scheduled task start and end times.
Approximate Due Date using Standard Duration	The system uses the calendar associated with the working hours of the Default Service Delivery queue to approximate due dates, rather than consulting the actual participant assigned to each task.	All tasks (authorization/review and delivery) are instantiated, but individual queue or person calendars are not consulted.
Do not forecast Due Date	No due dates are forecast when the order is submitted. The user sees "TBD" as the Due Date in My Services.	Only authorization/review tasks are instantiated when the request is submitted.

There are both functional and performance implications in choosing the method for forecasting due dates. The following factors affect the accuracy of due dates forecast:

- Any authorizations or reviews associated with the service. Any initial forecast uses the specific duration associated with each of these authorizations. However, due dates are recomputed after all authorizations are completed. Variations in completion time for authorizations and reviews (typically, taking longer than specified in the delivery plan) can give an unrealistic forecast.
- Any conditional tasks included in the delivery plan. Since the conditional expression governing execution of the task can only be evaluated in the service delivery moment, any forecast cannot take conditional task execution into account. Therefore, the estimate may be inflated by included tasks that would actually not be executed. On the other hand, the estimate would always be a worst-case scenario (assuming execution of all tasks), so users could be pleasantly surprised if the service request is fulfilled sooner.

When due dates are forecast (either via Approximation or Estimate), Request Center must create (instantiate) all tasks in the delivery plan. This may take a significant amount of server processing time, depending on the number of tasks in the plan. Estimating the due dates will always take longer, since the work calendar assigned to each participant in the plan must be consulted to correctly derive the task start and end dates.

The Administration setting to "Submit, Approve, and Review Asynchronously" affects the perceived performance of Request Center when it comes to forecasting due dates. By default, this setting is off, so that "background processing of requisition submit" is disabled. Therefore, Request Center instantiates these tasks synchronously; that is, the user submits the order, Request Center creates the tasks as instructed in the forecast method, and then control of the page is returned to the user. For complex task plans, the user may wait a significant amount of time to proceed away from the order form.

If "Submit, Approve, Review Asynchronously" is on, Request Center creates tasks asynchronously; that is, in the background. Consequently, the user does not have to wait until all tasks have been created to exit from the order form and continue using Request Center. The wait time is eliminated. The difference in performance may not be obvious for requests with few (or no) authorizations or with simple delivery plans, but should be apparent for services with more complex workflows. After requisition submission, the status becomes "Ordered" until it is processed by the Business Engine. Afterwards, the status becomes "Ongoing".

Because asynchronous processing requires additional configuration steps and it might not produce any perceived difference in performance for some installations, configuring Request Center to "Submit, Approve, and Review Asynchronously" is optional. Be sure to check with your system administrator to see if this setting has been enabled before deciding on a method for forecasting due dates.

Viewing and Searching for Services

Services are visible in the **Services** component of Service Designer, which displays by default when you first enter the Service Designer module.

In the Services pane on the left, use the tree menu to expand Service Groups (by clicking on the **+** sign next to each group). Then, click on a service name to view its details in the right-hand content pane.

To do a quick search for services:

Step 1 At the top of the Services pane, click **Search**.

A Select Service window opens.

Step 2 In the text field, enter the service name or partial name, and then click **Search**.

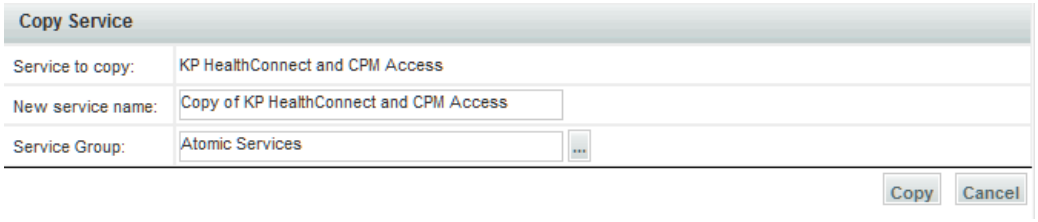
You do not need to use a wildcard character in your search string. Request Center automatically searches for any service whose name contains the search string.

Step 3 Click the desired service from the list results and click **Select**.

Copying a Service

Service Designer includes the ability to copy a service definition. Copying a service copies the complete service definition, including any associations with keywords and categories as well as the associations to included active form components.

To copy a service definition, click **Copy** on the General tab of the service definition. The Copy Service popup window appears. Enter the name of the new service and choose the service group into which the new service definition should be placed. Only service groups in which the current user has permission to "Design forms" are available for selection.



Copying a service to a new group, does not affect the service group authorizations of the target group—these remain the same.

Exporting and Importing a Service

Export/import is provided primarily for backward compatibility with previous versions of Request Center. It provides a way to copy or "clone" an existing service by downloading the service definition to the local workstation; modifying the service name; and importing the revised service definition. Cloning a service is best accomplished by "Copying a Service" as described above.

The procedure for using service export/import is:

1. Configure the service.
2. Export the service.
3. Modify the service name in the exported XML file.
4. Import the modified XML file.

Although service export/import may work in transferring a service definition from one site to another, this is not recommended. The export includes associations to people, OUs, queues, roles and other entities that are not part of the service definition. Importing the service does not create these entities in the target environment if they do not already exist; it simply drops the association without issuing any error messages. Catalog Deployer is the recommended tool for transferring Request Center content between sites.

Export a Service

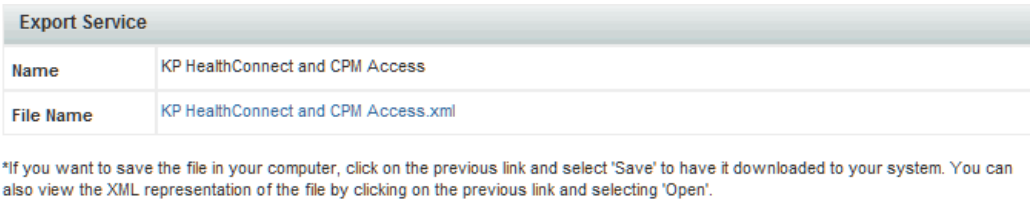
To export a service:

Step 1 In Service Designer, click the **Services** component.

Step 2 Click the service you wish to export.

Step 3 From the General tab, click **Export**.

The Export Service window appears.



Step 4 Right-click the File Name link and choose **Save Target As** to save the file to your desired location.

Service Designer exports the service in XML format to the location you designated.

Caution If you are using Internet Explorer, you may experience difficulty saving the XML file. If this occurs, go to **Tools > Internet Options > Advanced Tab > Security** and check the **Do not save encrypted pages to disk** option. This option is unchecked by default.

Edit the Exported XML File for a Service

To edit the exported XML file:

Step 1 Open the XML file in a plain text editor such as Windows Notepad.

Step 2 Change the service name in the file where it reads:

```
<ServiceDefinition name="service name here" ....>.
```

If you do not change the service name within the XML file, the import process will overwrite your original service. (It is possible to change other elements in the XML file, but this practice is not recommended. If invalid XML is produced, the import will fail.)

Step 3 Click **Save** to save the edited XML file.

The edited XML file is now ready for import.

Import a Service

You can import a service that has previously been exported to an XML file in which the ServiceDefinition has been renamed. Once the service is imported, you can edit the service as desired.

To import a service:

Step 1 In the Services component, choose **New > Import**.

The Import Data window displays.

Step 2 Browse to, or type the filename (including the path) in the Import from file: field.

You can only import an XML file that has been previously exported and saved to its default location or a location on the local machine.

You can also use the Browse function to choose a file on your workstation.

Step 3 Click **Import**.

Service Designer displays an Imported Data screen to confirm the import.

Step 4 Click **OK** to refresh the Service Catalog display.

The imported service appears in the service group specified in the XML file.

Deleting a Service

You can delete a service if you no longer need it, *only* if the service has not previously been requested, or all requisitions from the service have been purged from the system. A service cannot be deleted if service requests have been submitted against it, even if these requests have already been completed.

If you want a service to no longer be available for ordering, you can change its status to Inactive. Further, it is sometimes helpful to move the service to a different Service Group, so service designers do not need to browse through a list of inactive services to find the one they are looking for. For example, a shadow group named zz_<Service Group> would get the inactive services safely out of the way, but still maintain the service group structure.

Configuring the Offer Tab

Use the **Offer** tab to:

- Create a service bundle
- Specify recommended prerequisite or recommended accessory services
- Set pricing for the service

No Request Center behavior is associated with the Objectives subtab; objectives are used by Demand Center.

Service Bundles

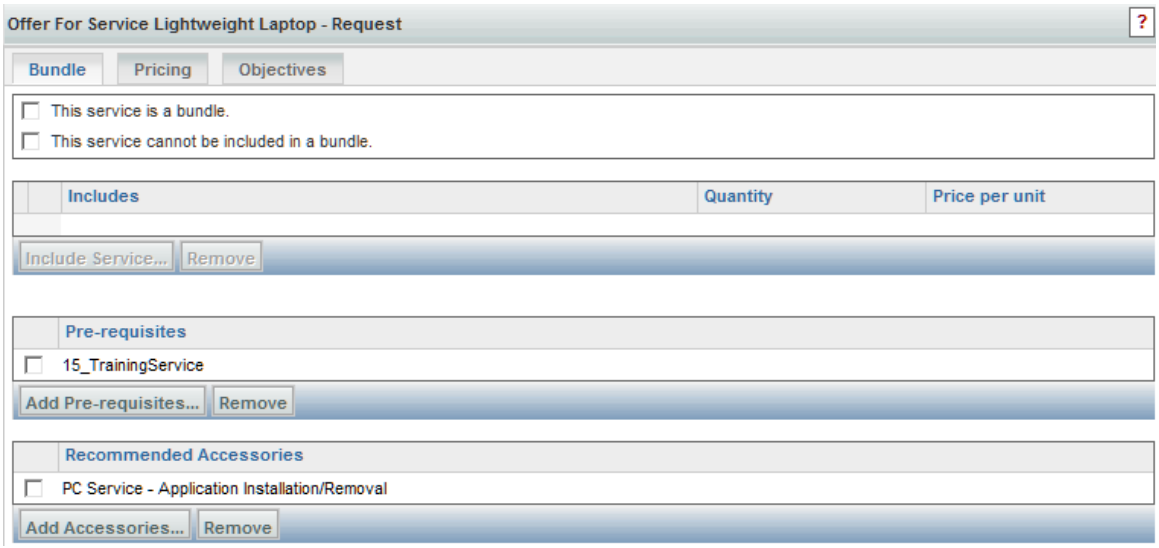
Use the Bundles subtab to indicate whether the service is part of a bundle of services. A bundle of services is a group of services which are automatically requested when the bundle (parent) is ordered.

For details on bundles see the "Bundling Services" section on page 1-62.

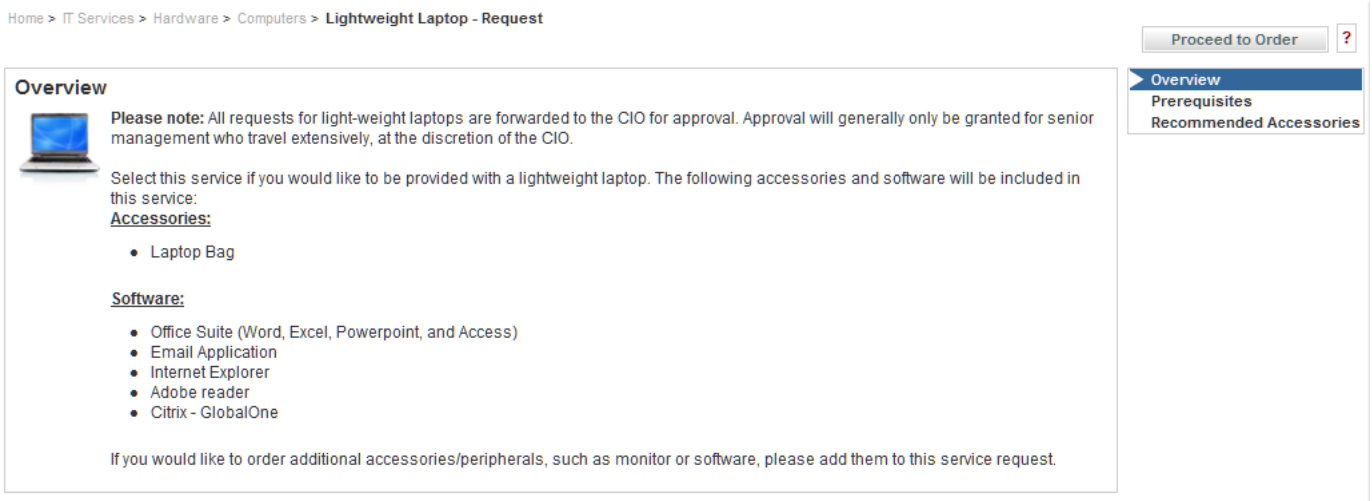
Prerequisites and Recommended Accessories

Prerequisites and Recommended Accessories allow service designers to associate additional services with the current service. These options may be specified both for bundled and nonbundled services.

- **Prerequisites** are other services that are required before ordering this service. For example, a customer must have a computer purchased and installed before ordering software installation.
- **Recommended Accessories** are other services that are recommended as add-ons to the service. For example, if the service is for a cell phone, recommended accessories might be a headset and case.



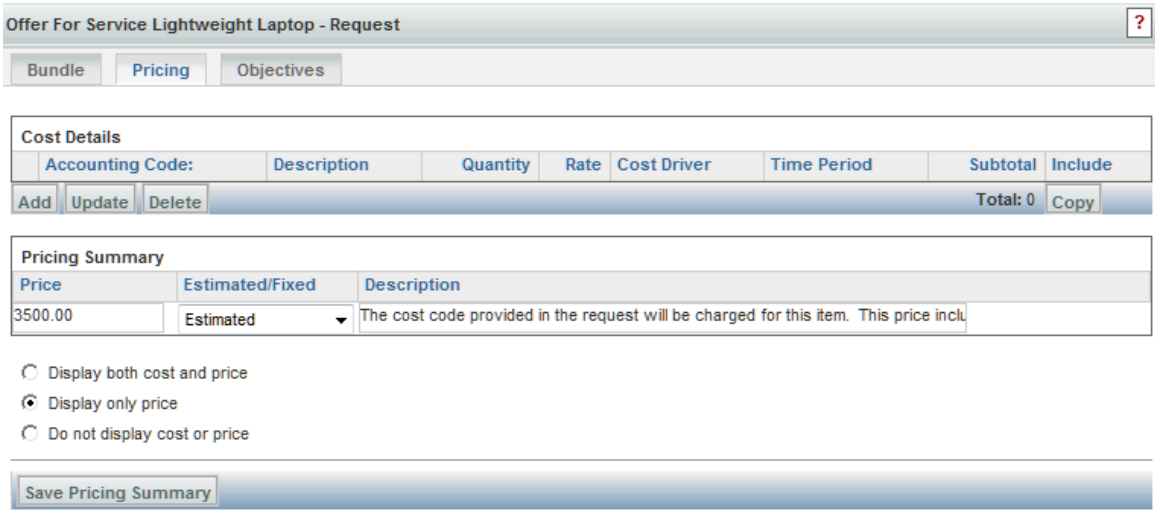
If a service has prerequisites or recommended accessories, the corresponding link appears to the right of the service's detailed description in My Services:



There is no behavior associated with either; the listed prerequisite services or accessories must be ordered individually. These are optional but can be helpful to the end user.

Pricing a Service

Use the Pricing subtab to configure summary pricing and detailed cost information for the service. Service costs and prices can be used as the basis of chargebacks for delivering the service.

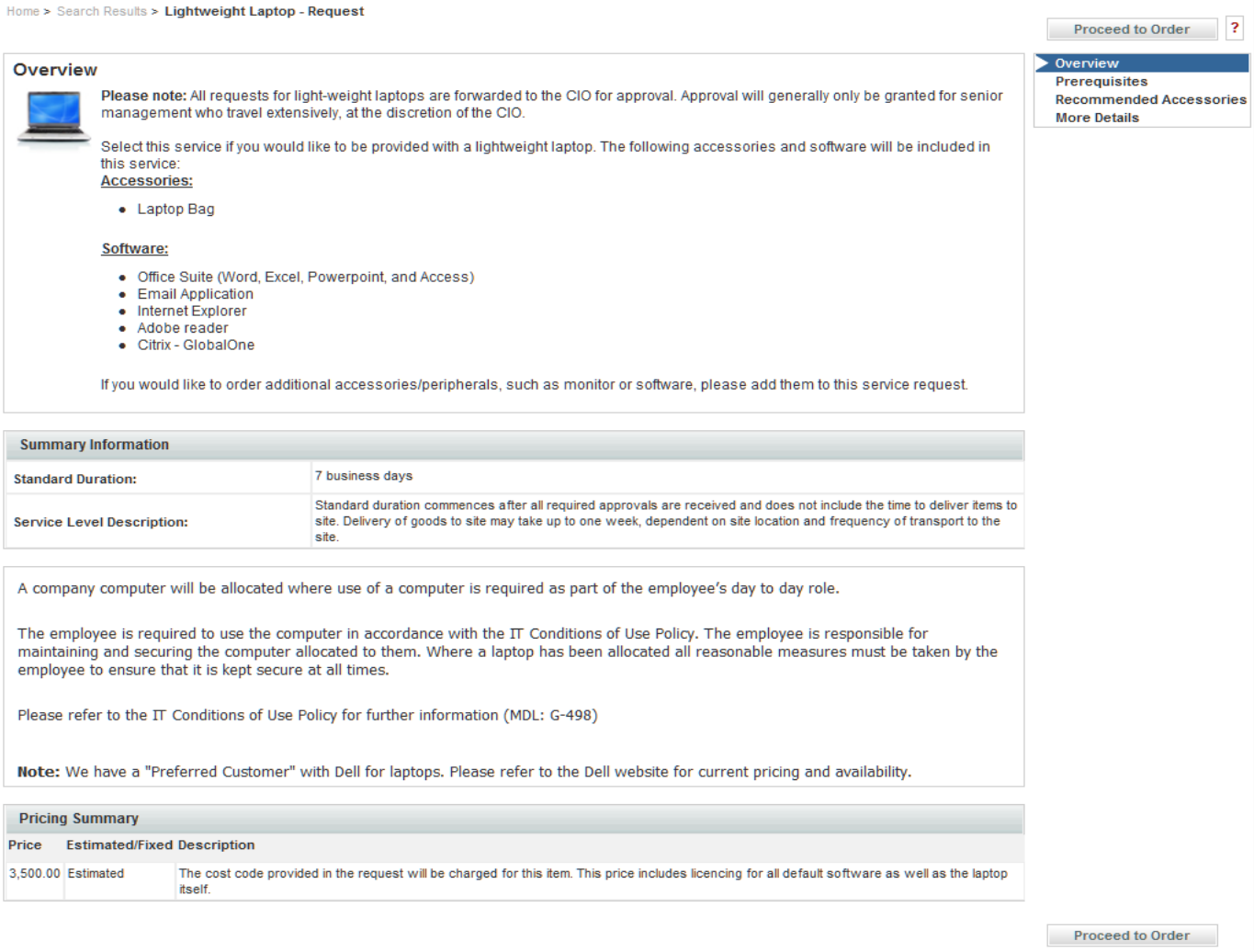


Use the table below as a reference when configuring Cost Details. Remember to click **Update** when you are done.

Field	Definition
Accounting Code	Category of expense, such as Capital expense or labor expense. Click **Add** to add a code. Click **Update** to update information.
Description	Description of the chosen code.
Quantity	Number of the chosen expense to be included.
Rate	The cost per service. The quantity and the rate are calculated and reflected in the total if you choose the Include option. You can also copy the total into the Pricing Summary if you choose **Copy**.
Cost Driver	Units appropriate for the chosen service such as BTUs or CPUs. The list of available Cost Drivers is configured in the Administration module in Lists.
Time Period	The time period in which the service expense is applied, such as Daily or Once.
Subtotal	Price based on the quantity and the rate for the given expense.
Include	Choose to include the expense in the pricing total.
Copy (button)	Click to include the Total from the Cost Details into the Pricing Summary information.

Use the table below as a reference when configuring the Pricing Summary. Remember to click **Save Pricing Summary** when you are done.

Field	Definition
Price	The summary price for this service. If you want the summary price to be the same as the total of the Cost Details, click **Copy**.
Estimated/Fixed	Type of pricing for the service: • **Fixed price**: This option indicates that the price is fixed. • **Pricing Required**: This is both a descriptive field and one that affects system behavior. It activates the Pricing moment for the service and My Services displays the price entered in the Price field as "subject to pricing." The Pricing moment occurs before the Authorization moment, if applicable, and the Delivery moment. If a "Pricing Required" task is created, it is sent to the Default Service Delivery Queue. A Service Manager user with Access Queue permission for that queue must enter a final price during the Pricing moment and click **Set Price** to advance the service request. The Pricing moment is optional. • **Time and Materials**: This option indicates that the price is based on time and materials. • **Leased**: This option indicates that the price is based on lease. • **Estimated**: This option indicates that the price is estimated. • **Priced Dynamically**: This option indicates that the service price is determined dynamically. The type of pricing appears to the customer in My Services unless you choose the "Do not display price" option. The display of cost or price to approvers and task performers is not affected by this setting.
Description	A description of the chosen pricing.
Display options	Customers can click on the service name to get more details about the service before ordering it. If an option to display pricing is chosen, the pricing summary and cost details appear on the Overview page, as shown in the sample below. • **Display both cost and price**: The individual expense lines you specify in the Cost Details area and the Pricing Summary appear to the customer. • **Display only price**: Only the Pricing Summary appears to the customer. • **Do not display cost and price**: Both the Cost Details and Pricing Summary do not appear to the customer.



Pricing Options and Dynamic Pricing

The options specified on the Pricing Summary determine how price information is displayed in the service Overview in My Services. Only the "Pricing Required" option influences the workflow of the service, by inserting a pricing moment after the request has been submitted. Because this behavior is inflexible, it may not be suitable for most scenarios where a price must be determined dynamically.

To provide more flexibility, service designers can write active form rules that dynamically set the price of the service. Like any active form rules, rules that set the price for a particular service request can be executed conditionally at any time during the authorization, review, and delivery cycle. The updated "transactional price" of the service request is shown on the Requisition Status page of My Services.

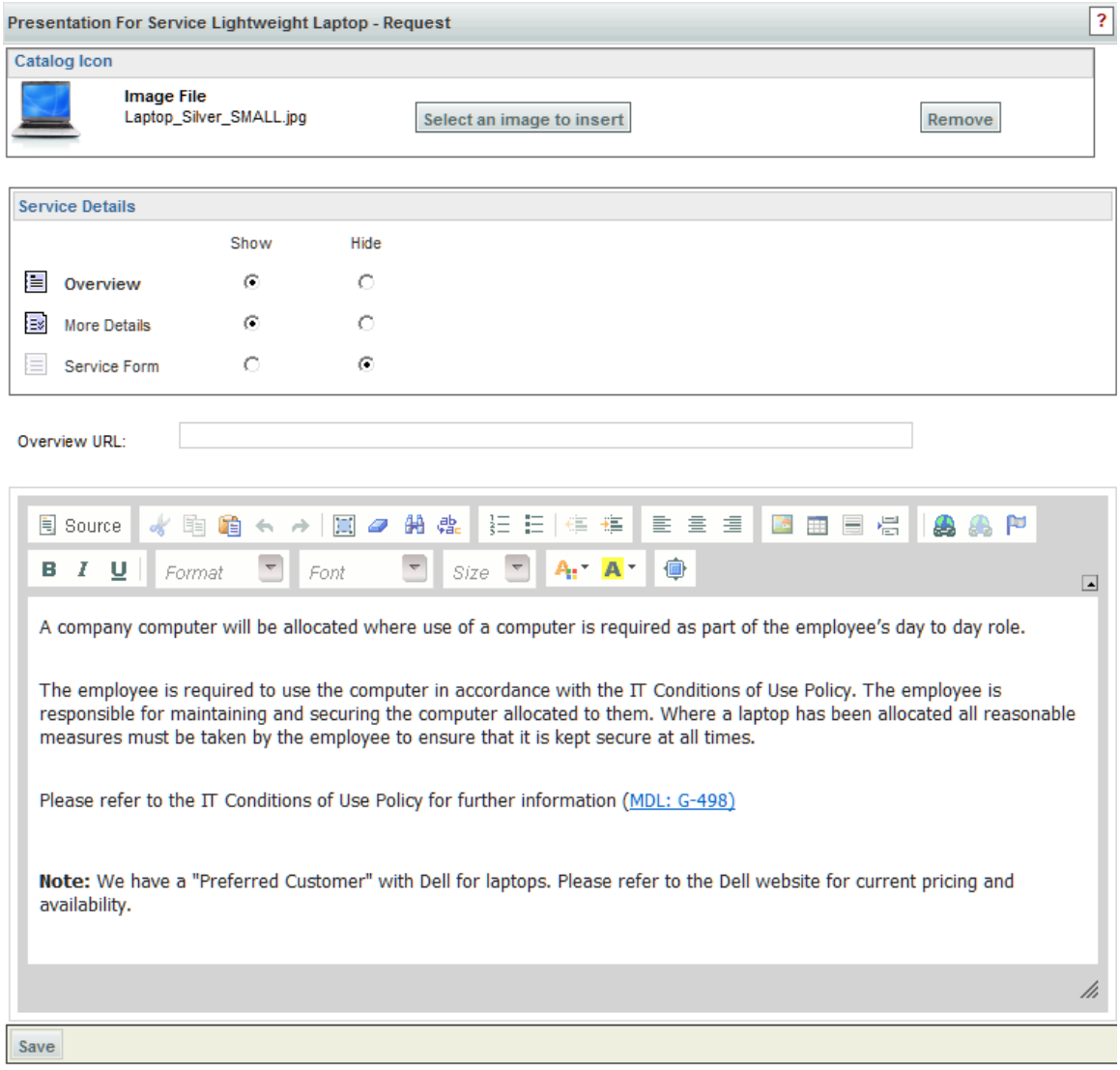
Dynamic pricing rules can be used in any services, not just those that are "Priced Dynamically". The only benefit of using this option vs. "Fixed" is that it better sets the customer's expectations.

For more information on using rules to dynamically price a service, please see Chapter 2, "Active Form Components".

Formatting the Service in the Catalog

The appearance of a service within the service catalog is configured on the service's **Presentation** tab. This includes:

- Associating an image with the service
- Entering text, which could include HTML formatting, to further describe the service
- Associating one or more external links (URLs) with the service
- Deciding which sections of the form to display and the manner in which they display

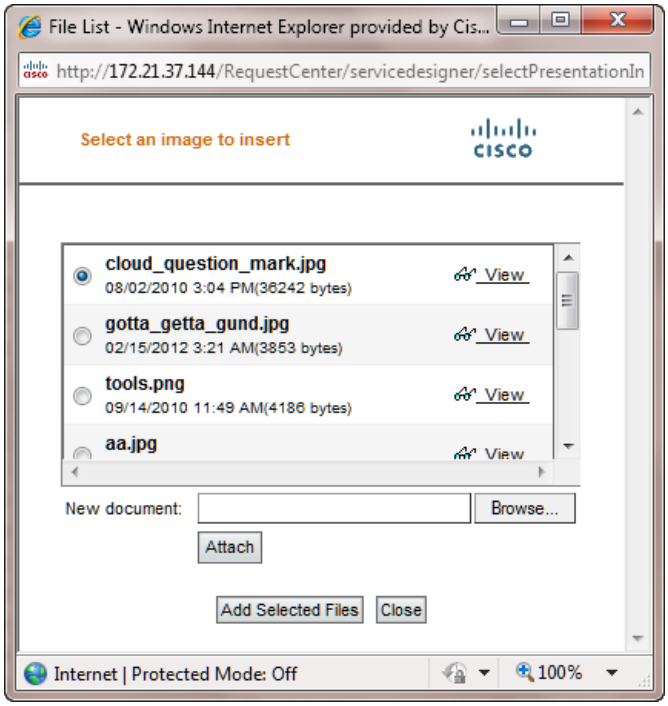


To associate an image with the service:

Step 1 Click the **Presentation** tab for the service.

Step 2 Click **Select an image to insert**.

The “Select an image to insert” dialog box appears, as shown below.



Step 3 If the image has already been uploaded to Request Center, it is listed and can be previewed by clicking the View icon. To choose a previously uploaded image, choose the image and click **Add Selected Files**.

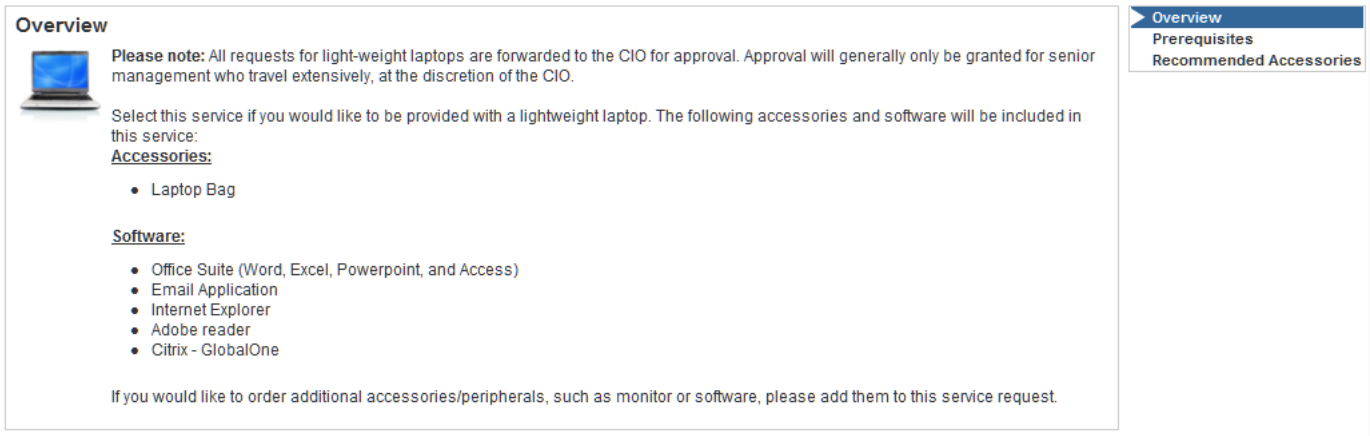
Step 4 If the image has not yet been uploaded, either enter its name in the New document: text box, or click **Browse** and find the image on your workstation. Click **Attach** to associate the image with the service.

Images which represent services can be in JPG,TIF, PNG, or GIF format. The images are displayed with a width of 64 pixels and a height of 64 pixels, so custom images should be created to maintain this aspect ratio. You may use any picture editing software to ensure that the size and aspect ratio of the image matches those dimensions. If the picture is not sized correctly, it is resized by the browser and may appear blurry or pixilated.

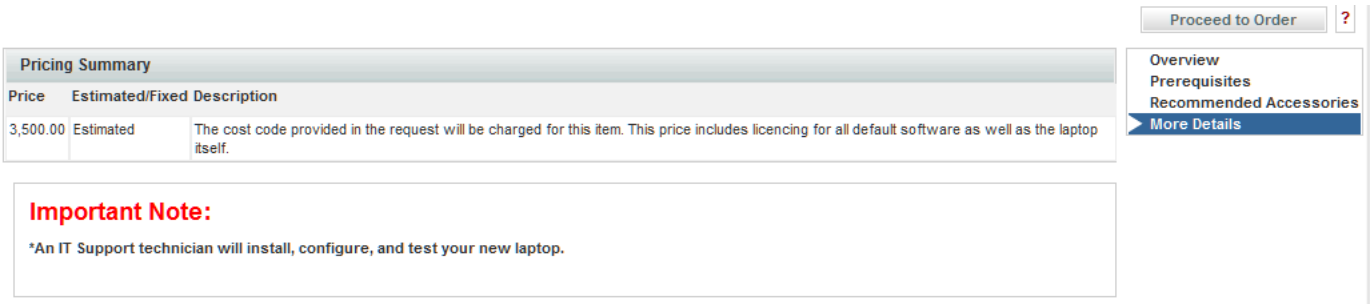
Use the Service Details section to associate additional descriptive information with the service. Any of the description entries can include HTML formatting. Additional HTML elements or even JavaScript functions could be referenced by editing HTML source via the Source button provided by the editor.

Step 1 Choose which of the following sections you want to Show (and format) content for:

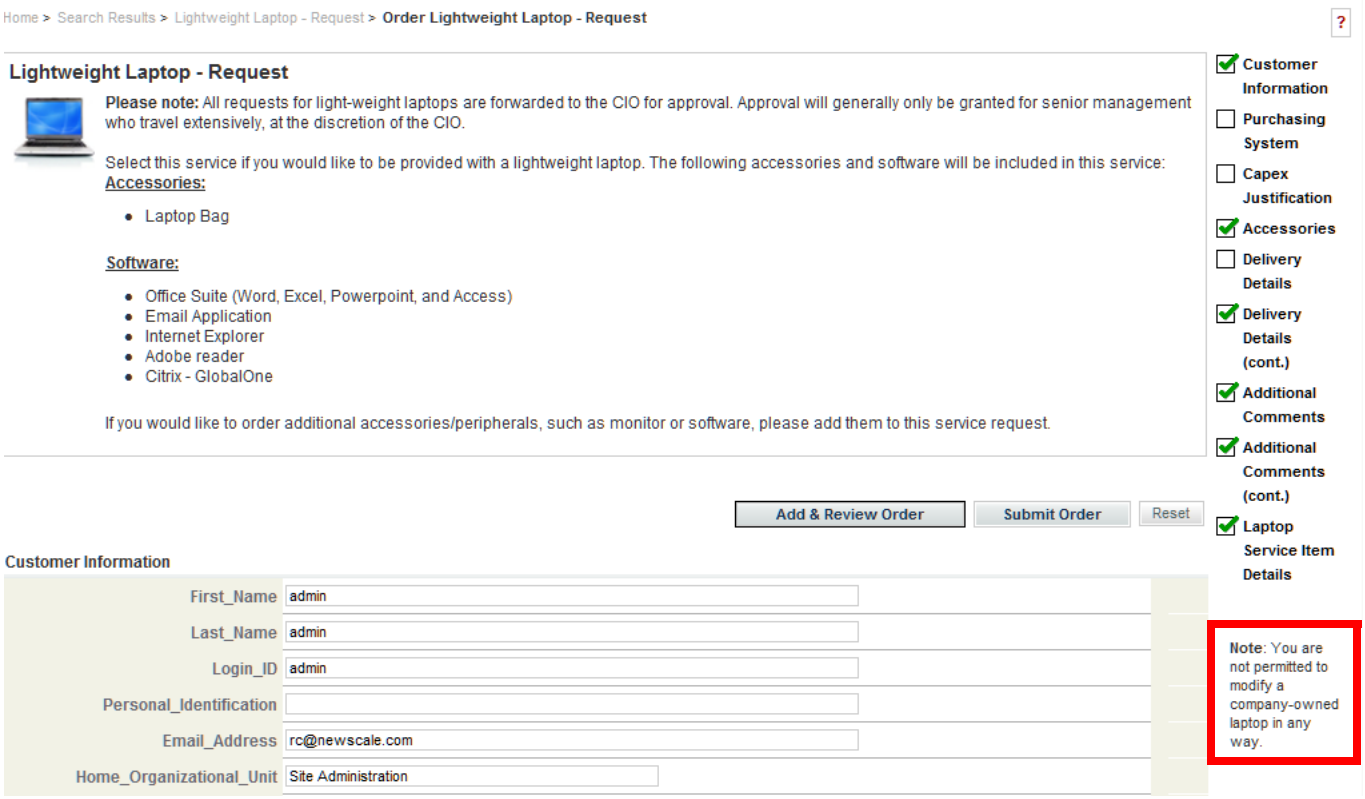
- **Overview**: Information entered here is visible in My Services when the user clicks on the service name or service icon to display an Overview of the service.



- **More Details**: If information is entered here, the My Services Overview page includes a More Details link directly below the Overview link. This information appears to the right of service information. (The "IMPORTANT NOTE" below is the More Details section.)



- **Service Form**: Information entered here is displayed on the right side of the service form, immediately below the dictionary monitor (see Note on bottom right of the image below).



When you click **Show** for a section, the bottom portion of the window populates with an HTML editor. See the "HTML Editor Tools Summary" section on page 1-84 for a legend of this toolbar.

Step 2 If you choose to show multiple content panes, click the name of the section you want to format.

Step 3 If desired, enter a URL to display within the left, right or bottom content sections. The URL must be fully qualified, beginning with http://.

Step 4 Use the included HTML editor to enter and format text and graphics (this is chosen by default).

Or, to insert HTML code, click **Source** to disable the HTML editing tools and enter your coding directly. Click **Source** again to return to the HTML editor to display the rendered HTML code.

Step 5 Click **Save** to save changes.

Preview the Service Presentation

It is a good idea to preview the service presentation in My Services before proceeding.

To do so:

Step 1 Choose the **My Services** module.

Step 2 Navigate to the service and check your work.

Step 3 Return to **Service Designer** and choose the service.

Step 4 Make any necessary modifications, and be sure to click **Save** to save changes.

Step 5 Return to **My Services** to verify the service presentation is correct.

Rather than switching back and forth between My Services and Service Designer, it might be easier to open two browser windows running Request Center; keep the Service Designer service's Presentation subtab displayed in one, and the My Services order page for the service in the other. Make changes as required in the service's presentation, remembering to save the changes by clicking **Save**. As soon as you have clicked Save, you can refresh the My Services page to review your changes.

Configuring the Delivery Plan

A service's **Plan** tab is used to define the delivery plan for a service. A delivery plan comprises one or more tasks that must be completed to deliver a service to a customer.

Core Concepts

To configure the delivery plan for a service you should have a clear understanding of the following concepts:

- Using Namespace variables and configuring expressions, as detailed in Chapter 6, "Namespace"
- Configuring and using Email Templates, as detailed in the *Cisco Service Portal Configuration Guide*
- Calculation of due dates

There are several steps involved in creating the delivery plan:

- Configure the Project Manager for the service delivery process.
- Configure individual tasks, including task name, duration, conditions, priority and other parameters.
- Specify the performer and supervisor of the task.
- Specify notification emails related to the task.
- Configure task instructions.
- Create a task checklist.
- Specify the workflow for the tasks, including the sequence for the execution of tasks, whether tasks execute concurrently or consecutively, and potentially grouping tasks that share a common milestone or have common notification requirements.

Working in the Plan Tab

Designing the workflow for a service might require several design sessions with service stakeholders. Ideally, you would interviewed all the people involved in setting up and delivering the service in order to determine what the tasks (or series of tasks) are, whether the service requires email notifications, and who the performer, supervisor, and email recipients are. Once the requirements have been collected, you use the Plan tab to configure the service to comply with those requirements.

Service Designer offers two complementary ways to specify the tasks which comprise the workflow of the service and the order in which these tasks are executed:

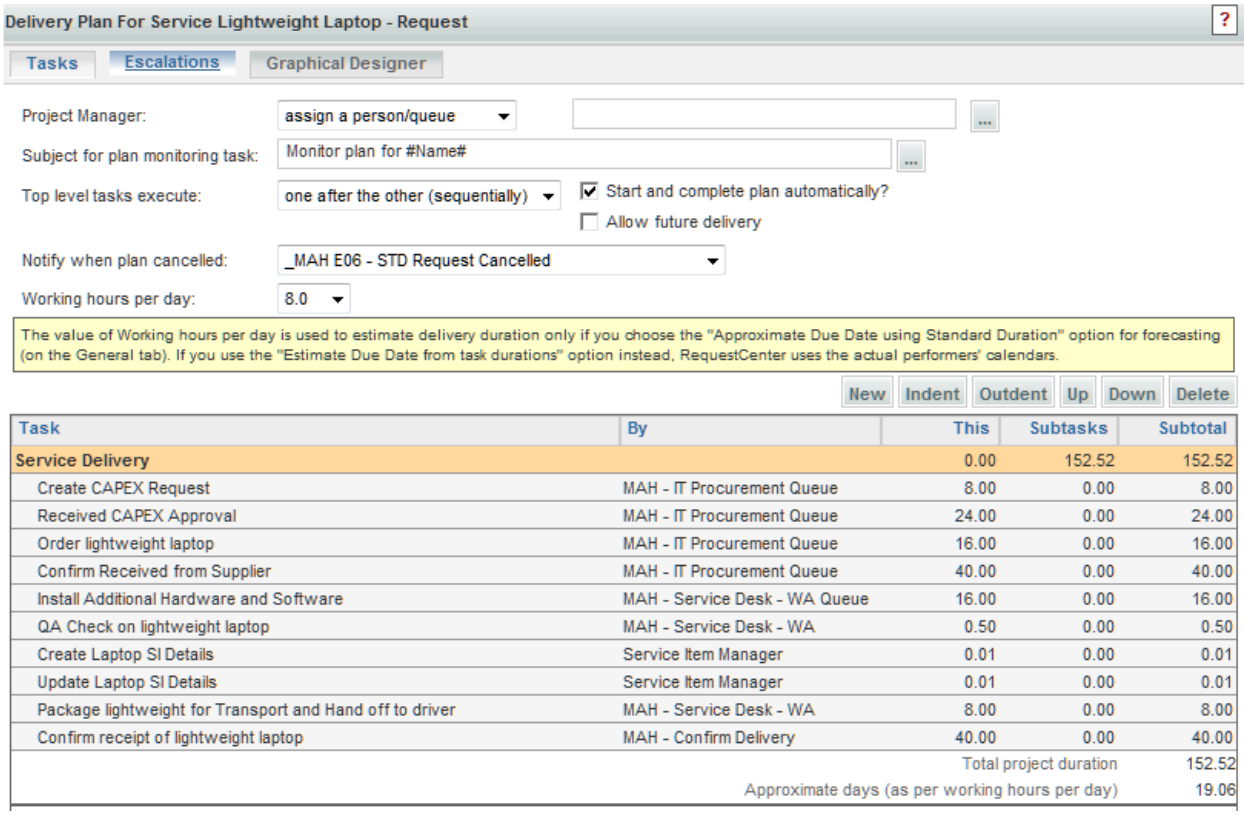
- The Tasks subtab is textually oriented; you define the tasks by listing them in the appropriate order; subtasks and parent tasks are indicated, much the same way as Project Management software shows projects and deliverables within a project.
- The Graphical Workflow Designer is a drawing tool. It allows you to draw a diagram that shows the tasks and their relationships.

The Plan tab includes the following subtabs:

- **Tasks**: The Tasks subtab is open by default. Use this to specify the tasks of the service delivery plan in the top portion of the window. Then, specify the detailed activities for each task in the lower portion of the window.
- **Escalations:** Use the Escalations subtab to specify the email notifications sent to performers, supervisors, and customers when an activity is late.
- **Graphical Designer:** The Graphical Workflow Designer allows you to draw a diagram to specify the tasks and subtasks that comprise the delivery plan, the sequence in which they are executed, and whether execution is sequential or concurrent.

Project Manager and the "Monitor" Task

Every delivery plan automatically has one overall task, which allows a designated project manager to monitor the delivery plan's progress.



Task	By	This	Subtasks	Subtotal
Service Delivery		0.00	152.52	152.52
Create CAPEX Request	MAH - IT Procurement Queue	8.00	0.00	8.00
Received CAPEX Approval	MAH - IT Procurement Queue	24.00	0.00	24.00
Order lightweight laptop	MAH - IT Procurement Queue	16.00	0.00	16.00
Confirm Received from Supplier	MAH - IT Procurement Queue	40.00	0.00	40.00
Install Additional Hardware and Software	MAH - Service Desk - WA Queue	16.00	0.00	16.00
QA Check on lightweight laptop	MAH - Service Desk - WA	0.50	0.00	0.50
Create Laptop SI Details	Service Item Manager	0.01	0.00	0.01
Update Laptop SI Details	Service Item Manager	0.01	0.00	0.01
Package lightweight for Transport and Hand off to driver	MAH - Service Desk - WA	8.00	0.00	8.00
Confirm receipt of lightweight laptop	MAH - Confirm Delivery	40.00	0.00	40.00
			Total project duration	152.52
			Approximate days (as per working hours per day)	19.06

The table below summarizes the fields on the Tasks subtab that pertain to the "Monitor" task, that is, the overall delivery plan.

Field	Description
Project Manager	Assigns a project manager from a position, a person/queue, or an expression. The project manager is the person, position, or queue that receives the Plan Monitor Task for a service in Service Manager. This task allows the project manager to oversee the entire delivery process including making adjustments to staffing, rescheduling and completing tasks, and even cancelling the entire delivery plan, if needed. Clicking on the "**...**" button on the right pops up a dialog box allowing you to search for and designate a position, a person/queue, or to type in the parameters of your expression. For an expression, click **Set Expression** to save. It is important to assign a person, queue, or position to be project manager for the delivery plan. This user is responsible for the plan monitor task.
Subject for plan monitoring task	Text describing the subject of the monitoring task for the plan. It is recommended that you place the word "Monitor" in the subject. The plan subject may include namespaces, as documented in Chapter 6, "Namespace". Clicking on the "**…**" button allows you to edit the subject. Click **Set Subject** to save changes.
Top level tasks execute	Indicates whether top-level tasks in the delivery plan execute concurrently (at the same time) or sequentially (in order, one after the next).
Start and complete plan automatically?	This check box is checked by default. This means that all tasks in the plan are automatically created, and their due dates computed, when the delivery plan begins. If you clear the check box, then the Project Manager assigned must take action before the delivery plan goes into effect. When the delivery moment begins only one task is created—the MONITOR task. That task presents a button labeled "Start Plan". This allows the Project Manager to "staff" the plan (using the Staffing page on the MONITOR task). On that page the Project Manager can reassign any performing position, person or queue in the delivery plan. When the Project Manager has finished Staffing the plan, they click **Start Plan**, which initiates the first tasks in the delivery plan. Until that button is clicked, tasks are in "Staffing" status.
Allow future delivery	Check this to allow end users to request a service in advance of when service delivery should begin. Once this option is checked, the end user sees a bar across the top of the ordering page which allows them to choose the date they want the service delivery to begin. End users cannot specify the due date/completion date for a service. If the user selects a future delivery date, Request Center holds the order in a waiting state until the date chosen by the user. Beginning on that date, the service delivery process begins, and proceeds as configured.

Field	Description
Notify when plan cancelled	Allows the service designer to configure an email to be sent when the Project Manager cancels delivery of a service.
Working hours per day	Used to convert hours to business days for the Standard Duration. See the "General Subtab" section on page 1-33 to set the Standard Duration and Display Units for the service.

Delivery Tasks

The table below summarizes the fields on the Tasks subtab.

Field	Description/Usage
New	Add a new task to the delivery plan.
Up Down	Change the order of tasks in the task list.
Indent Outdent	Group or ungroup tasks. For example, if you create a task and click **Indent**, the task becomes a subtask of the task above it.
Delete	Delete a task (and all subtasks beneath it).
General subtab	Create detailed plan activities for delivery of a task including: the task name, execution order, duration, priority, and conditions. If your company's installation is integrated with third-party software using Service Link, then you also choose the external action here. If you are using Service Item Manager, you can also choose Service Item tasks here.
Participants subtab	Define the details about the performer of the task, including who completes the task and how the work is supervised.
Email subtab	Define the optional email templates that the system sends when the task starts, completes, is cancelled, is rescheduled, is reassigned, or when an external task fails.
Task Instructions subtab	Enter instructions for the task or set links to any task instruction-related URLs.
Checklist subtab	Create a list of reminders or detail the procedural steps for completing a task. These appear in Service Manager as a checklist, detailing steps for completing the work.

For each task in the delivery plan, move through the General, Participants, Email, Task Instructions, and Checklist subtabs to define the task activities and delivery plan.

Designing Request Center Workflows

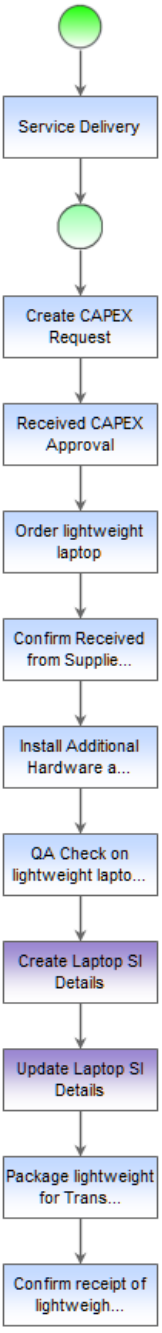
Request Center offers two approaches to designing the workflow of tasks that comprise the delivery plan:

- Use the Task area at the middle of **Plan** tab to enter tasks, in conjunction with the buttons available (Indent, Outdent, Up, Down) to configure the workflow so that tasks execute in the appropriate order and are grouped correctly.

- Use the Graphical Workflow Designer, available from the **Graphical Designer** tab, to draw the workflow by dragging and dropping tasks, connectors, and subtask groupings onto the drawing area.

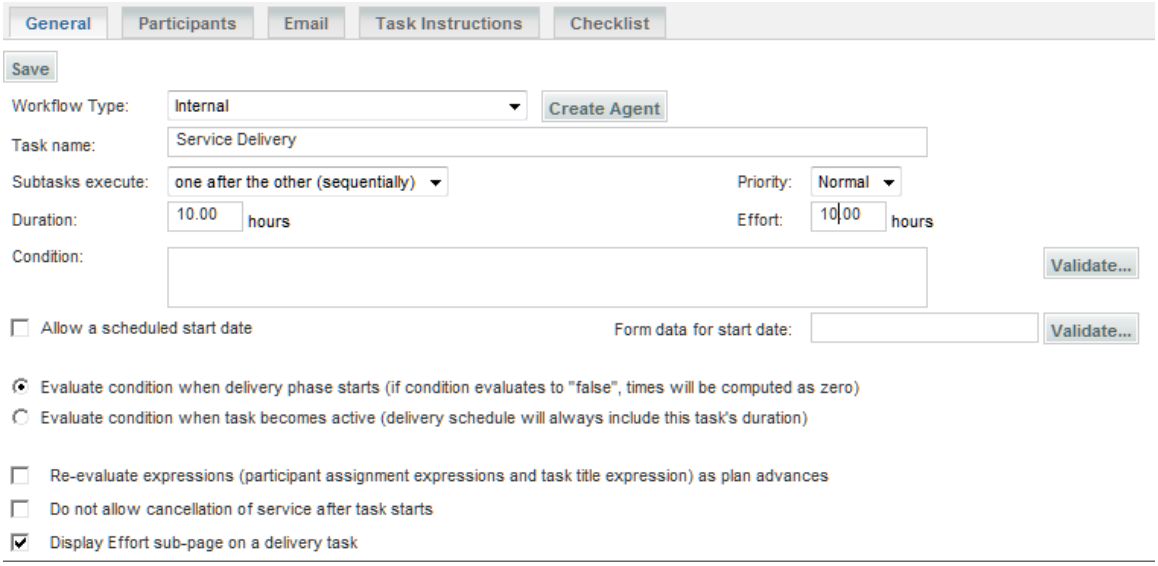
A workflow can be configured using either tool, and, in fact, service designers can switch back and forth between Graphical Designer and the dialog boxes provided on the Task tab. The Graphical Designer provides the workflow for the tasks, and its property sheet allows users to enter most general information about the task as well as the performer's role. Other individual task details, such as emails associated with task fulfillment, checklists, and task participants must be supplied by using the subtabs of the Plan tab.

Task
Service Delivery
Create CAPEX Request
Received CAPEX Approval
Order lightweight laptop
Confirm Received from Supplier
Install Additional Hardware and Software
QA Check on lightweight laptop
Create Laptop SI Details
Update Laptop SI Details
Package lightweight for Transport and Hand off to driver
Confirm receipt of lightweight laptop



General Subtab

The **General** subtab is where you define the detailed delivery activities for a task or series of tasks. Ideally, you should be able to move through the fields described below in order as you create detailed delivery activities. If multiple tasks are listed on the Plan tab, click on the task name you'd like to configure. When the task is highlighted in blue/gray, you are ready to begin; each task has its own set of subtabs: General, Participants, Email, Task Instructions, and Checklist.



Field	Description
Workflow Type	The default is Internal. This is the only option displayed for a site that has no Service Link integrations defined with external systems and no Service Item tasks. Choose **Internal** if the task is executed within Request Center. Choose the appropriate **External** option if the task is executed in an external application (not within Request Center). For external tasks only, an ellipsis (…) button appears next to the Workflow Type after you have saved the task. See the "External Tasks in the Workflow" section on page 1-36 for more information. Choose **Service Item Task** if you are using Lifecycle Center's Service Item Manager module to design service items and track these items. More information on service item tasks is given in the "Service Item Tasks in the Workflow" section on page 1-37 and in Chapter 3, "Lifecycle Center". Choose **Directory Task** if the operation is for adding or updating Organization Designer people or queues. See the"Directory Tasks in the Workflow" section on page 1-37 for more information.

Field	Description
Create Agent	After you have saved the task, a **Create Agent** button appears. Click this button to use the Integration Wizard. See the *Cisco Service Portal Integration Guide* for complete details. **Note** The Integration Wizard is available only to those service designers who have been granted a role that allows creation of Service Link agents and transformations. The Integration Wizard automates many of the steps involved in implementing an integration. It is available only for creating integrations between Request Center and web services. The Integration Wizard works by retrieving the wsdl and operation to be invoked by the web service integration. Based on that definition of the integration, the Integration Wizard creates: • The Service Link agent that can be used in an external task to perform the integration • A transformation to transform nsXML into the SOAP message required by the web service • Agent parameters for all data required both in the initial web service request and the response • A dictionary containing fields mapped to the agent parameters • An active form component containing the dictionary created to hold agent parameter values
Task name	A brief title for the task. This appears as the subject of the task in Service Manager.
Subtasks execute	Use the drop-down menu to choose the order in which any "child" tasks of this task execute: one after the other (sequentially) or at the same time (concurrently). Child tasks (subtasks) become active (ongoing) in the delivery plan, and must be completed before their parent task becomes active. If child tasks are executed sequentially, the service duration includes the durations of all such tasks. If tasks are executed concurrently, the service duration includes the maximum duration of any child task.
Duration	The expected length of time between when the system says the task is active (begins) and when the task is completed, estimated to the nearest hour. For example, if installing a software program only takes 1 hour, you might give the performer a duration of 16 hours to finish the task, as the performer's workload and priorities may vary throughout the course of 2 workdays. The Duration value is not visible to the customer or performer, but is used to calculate the due date for the task.
Effort	The actual length of time it should take the performer to complete the task, estimated to the nearest hour. For example, it might take 1 hour to install a software program. While Duration is "total elapsed time", Effort is "time on task". The Effort value is not displayed to any users or used in due date calculations; it is used instead as an Internal Productivity Target.

Field	Description
Priority	Set the task priority to low, normal, or high. No system behavior is tied to the priority for a task, other than a flag which is visible in Service Manager. • If the priority is set to **high**, the system sets a red exclamation point in the Service Manager view where the task is displayed. • If the priority is set to **low**, the system sets a blue down-arrow in the Service Manager view where the task is displayed. • If the priority is set to **normal**, no flag is displayed in Service Manager.
Condition	If the task should be performed only under certain circumstances, type an expression in the Condition field that specifies when to perform the task. See the "Conditions" section on page 1-41 for more information.
Allow a scheduled start date	To make the task a delayed task, check **Allow a scheduled start date**. Then, in the Form data for start date field, do one of the following: • Enter a date and time in the following format: "YYYY-MM-DD HH:MM" such as "2006-12-23 13:30" (The quotes are required.) • Or, enter the name of the dictionary field you want to use to hold the date and time of the starting-point for the task. Use the following syntax: – Data.*DictionaryName.FieldName* for a field from a dictionary in one of the service's active form components. – ParentData.*DictionaryName.FieldName* for a field from one of the parent service's dictionaries (in the case of a bundled service). For more information about delayed tasks, see "Creating a Scheduled Start Task" section on page 1-60.
Form data for start date	If you have checked the "Allow a scheduled start date" option, you may enter form data for the start date and click **Validate**.
Evaluate condition when delivery phase starts	If you have specified a condition, click the option button to indicate when the condition will be evaluated. Choose this if you want the condition to be evaluated when the delivery moment starts for this service. This means that the information required to correctly evaluate the expression must be available before the delivery moment begins.
Evaluate condition when task becomes active	If you have specified a condition, click the option button to indicate when the condition will be evaluated. Choose this if you want the condition to be evaluated only when the delivery process has reached this task in the delivery plan.
Re-evaluate expressions as plan advances	This feature enables you to dynamically change the performer assignment for a task, as well as the name of the task, based upon form data. This is useful if you have designated that a task name or a performer assignment should be derived from a namespace expression using form data, and when the form data will be changed during the delivery process. Setting this flag specifies that the expression should be evaluated when the task becomes active (not when the entire delivery plan is initiated). **Note** **IMPORTANT**: The re-evaluation of the performer for a task does not affect the due date for the task. Due dates are not recomputed after the delivery moment begins.

Field	Description
Do not allow cancellation of service after task starts	If you check this box, the customer will no longer be able to cancel the service after this task becomes active. For example, check this if the task will cause the delivery team to incur substantial costs or take an action that is not easily reversible. The "Cancel" option in My Services is deactivated once the delivery team begins this task.
Display Effort subpage on a delivery task	If you check this box, an "Effort" subtab for a delivery task is shown in Service Manager.

As you compile and arrange your service delivery plan, the system keeps track of the total number of hours required to complete all of the plan's activities. It also uses the "Working hours per day" setting (Plan tab > Tasks subtab; see the "Working hours per day" section on page 1-31) to compute how many working days are required to execute the plan.

Delivery Task Name

The delivery task name may include the service name, through the use of the #Name# namespace. Service data (#Service.Data….#) can also be used. Since all tasks are created when the form is submitted, the value of the specified namespace must be available in the ordering moment. This practice is not recommended.

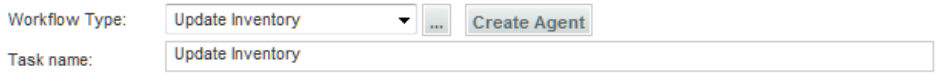
Using form data for a task name allows task performers to more easily differentiate tasks in Service Manager. However, it presents challenges in the reporting modules, since the task would no longer be automatically groupable by the task name; report designers would need to use "Custom Groups" to aggregate by such tasks. It may also require administrators to configure Service Manager to allow "contains" searches, which may adversely affect performance. Service designers should think carefully before including form data namespaces as part of a task name.

External Tasks in the Workflow

By default, the drop-down list for Workflow type contains only one entry, "Internal", indicating that the task will be performed within the Service Manager module. If integration specialists have used the Service Link module to define "agents" that integrate with external systems, the action associated with each agent also appears in the drop-down list. For example, the workflow type for the "SAP Integration" agent might be displayed as "Send data to SAP"; the workflow type for an agent that integrates with Remedy might look like "Send Ticket to Remedy".

Contact your site administrator to obtain details about the services and tasks integrated with Service Link.

For external tasks only, an ellipsis (…) appears next to the Workflow Type after you have saved the task.



If you have appropriate permissions to the Service Link module, you can click the "**...**" button to access the Agent Parameter Override dialog box. This dialog box includes settings that show how the data to be sent to the external system is mapped to fields on the service form or other data about the requisition. If you have permission to change these settings, the dialog box is editable; otherwise it is read-only.

Service Item Tasks in the Workflow

The Service Item Manager module (available only to customers who license Lifecycle Center) allows designers to designate certain items to be "service items", whose history can be tracked within Request Center. Typical service items might be a laptop; desktop; software license; or any corporate asset that can be uniquely identified and whose usage (and ownership) should be tracked.

A Service Item Task can be used only when the service includes a Service Item-Based Dictionary (SIBD). To configure the Service Item Task:

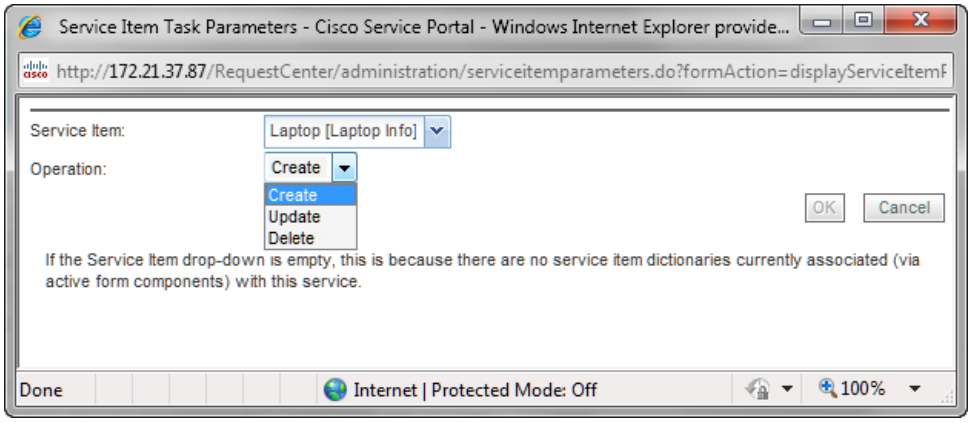
Step 1 Choose **Service Item Task** as the Workflow Type.

Step 2 Save the plan.

Step 3 Click the ellipsis that appears next to the Workflow Type.

The Service Item Task Parameter popup window appears.

Step 4 Choose the SIBD to use, and the operation (Create, Update or Delete) to apply. Then click **OK**.



For more information on service items and service item tasks, see Chapter 3, "Lifecycle Center".

Directory Tasks in the Workflow

Directory tasks can be used to invoke operations to add or update persons and queues using form data. If the operation fails, the task is marked as completed and an error message is written to the service form.

The following operations are supported:

Operation	Description
Create new person	Create a new person. If a person with the same login name already exists, an error is returned.

Update existing person	Update an existing person. If the person does not exist, an error is returned.
Create new person / Update existing person	Create a new person or perform an update if the person already exists.
Activate a person	Modify the status of a person to “Active”.
Inactivate a person	Modify the status of a person to “Inactive”.
Create new queue / Update existing queue	Create a new queue. If a queue with the same name already exists, an error is returned.
Update existing queue	Update an existing queue. If the queue does not exist, an error is returned.
Create/Update queue	Create a new queue or perform an update if the queue already exists.

Any free form dictionaries that contain the required fields can be used with the directory task. The dictionary should also include a text field named “ErrorDescription” for capturing any errors returned from the operation. Consider hiding this field during the ordering moment, and using the value of this field to conditionally trigger manual tasks for error handling.

To configure the Directory Task:

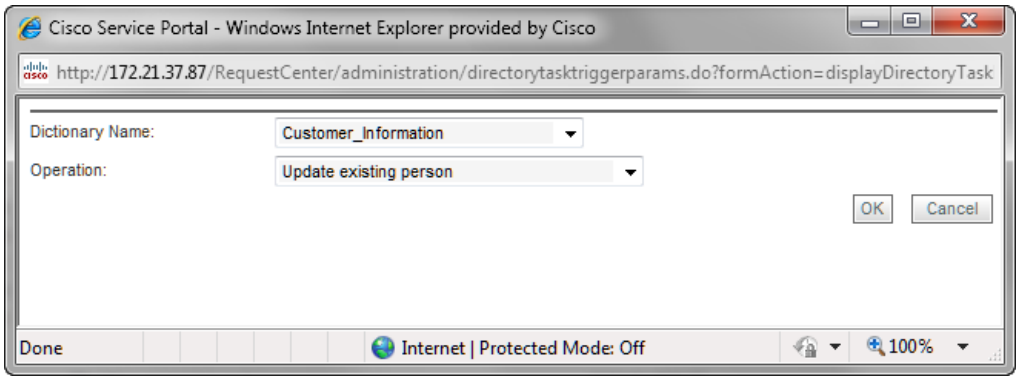
Step 1 Choose **Directory Task** as the Workflow Type.

Step 2 Save the plan.

Step 3 Click the ellipsis that appears next to the Workflow Type.

The Directory Task Parameter popup window appears.

Step 4 Choose the dictionary to use, and the operation to apply. Then click **OK**.



The required and optional fields for the directory operations are listed below:

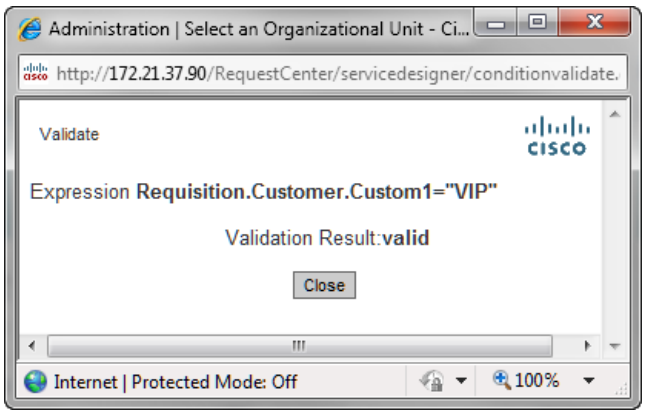
Operation	Dictionary Field	Remarks
Create or Update Person	**Input**: (* Mandatory for Create operation) First_Name* – Text(50) Last_Name* – Text(50) Login_ID* – Text(50) Password* – Text(50) Email_Address* – Text(100) Home_Organizational_Unit* – Text(200) Timezone* – Text(50) Organizations – Text(4000) Groups – Text(4000) Roles – Personal_Identification – Text(512) Title – Text(100) Social_Security_Number – Text(100) Notes – Text(4000) Company_Code – Text(200) Division – Text(200) Business_Unit – Text(200) Department_Number – Text(200) Cost_Center – Text(200) Management_Level – Text(200) Region – Text(200) Supervisor_ID – Text(200)	Organizational units or groups referenced are created automatically if they do not already exist. The values for the Person_ID and Organization_ID fields are returned to the service form when the person or Home OU are created during the operation. The value for the Timezone field should be in the form of the short name as seen in the Time Zone selections in Organization Designer; for example, “America/Los_Angeles”. The input type for the Organizations, Groups and Roles fields should be set to “select (multiple)” in the Active Form Component used.

Operation	Dictionary Field	Remarks
	Employee_Type – Text(200) Location_Code – Text(200) Company_Street_1 – Text(100) Company_Street_2 – Text(100) Company_City – Text(100) Company_State – Text(100) Company_Postal_Code – Text(100) Company_Country – Text(100) Building – Text(100) officecode – Text(100) Level – Text(100) Cubicle – Text(100) Personal_Street_1 – Text(100) Personal_Street_2 – Text(100) Personal_City – Text(100) Personal_State – Text(100) Personal_Postal_Code – Text(100) Personal_Country –Text(100) **Output**: Person_ID – Text(50) Organization_ID – Text(50) ErrorDescription – Text(80)	In an Update operation, the Login_ID field is used to look up the person record. Person attributes absent from the dictionary are excluded from the update.
Activate or Inactivate Person	**Input**: Login_ID – Text(50) **Output**: ErrorDescription – Text(80)	
Create or Update Queue	**Input**: (* Mandatory for Create operation) Name* – Text(100) Email_Address* – Text(100) Home_Organizational_Unit* – Text(200) Timezone* – Text (50) **Output**: Person_ID – Text(50) Organization_ID – Text(50) ErrorDescription – Text(80)	Organizational units or groups referenced are created automatically if they do not exist in the system. The value for the Timezone field should be in the form of the short name as seen in the Time Zone selections in Organization Designer, for example, “America/Los_Angeles”. The values for Person_ID and Organization_ID fields are updated back to the request form when the queue or Home OU are created.

Conditions

The use of expressions to configure a delivery plan is a versatile and powerful feature. A conditional statement allows tasks to be initiated or skipped based on the whether the expression used in the condition evaluates to true (include this task) or false (skip the task). The expressions are formulated using the namespaces and operators documented in Chapter 6, “Namespace”.

After you enter an expression, click **Validate** to make sure that the expression is correct. A Validate window is displayed, in which Service Designer indicates if the syntax of the expression is correct or has errors.



The message “unexpected token” indicates that the namespace used is not valid in this context or, perhaps, that you have forgotten to enclose an alphanumeric literal in quotes.

Validation checks that any namespace specified is valid for the current scope (the specific level of review/authorization or task), with the exception of dictionary fields (Data.*DictionaryName.FieldName*). However, it may cause a runtime error: if the specified namespace, for example, Data.EUIT_ACCESS.Access_Type, does not exist.

Validation also checks that correct relational and arithmetic operators are used.

If a condition has been entered, you must decide when that statement will be evaluated. The condition for each task (approval, review, or delivery) can be evaluated either

- At the beginning of a phase (Authorization or Delivery moment), or
- When the activity becomes active.

Evaluate condition when authorization/delivery phase starts

The designer can choose to evaluate a conditional statement when a **phase** starts by choosing the “**Evaluate condition when delivery phase starts (if condition evaluates to “false”, times will be computed as zero)**” option within a specific task, as shown above for delivery tasks. A **“**phase**”** corresponds to any of the system moments defined for processing a requisition. Each authorization or review has its own moment; all delivery tasks are performed within the Service Delivery moment.

Authorization tasks are always serial. You could put one conditional on one task saying if field= “some value” and a conditional on another saying field<> “some value”. That way, you know one authorization task will always be executed, and if you choose “when authorization phase starts” only one authorization task will appear in the process view. If you choose “when task becomes active” both tasks would be displayed, but one would be skipped.

The "**if conditions evaluate to "false", times will be computed as zero**" statement means that Request Center will evaluate the conditions at the beginning of the phase. If these conditions are false, then the corresponding tasks will not be executed, and the **Due Date** for the service will be calculated without including the duration of these tasks.

Evaluate condition when activity becomes active

Alternatively, the designer can choose to evaluate a conditional statement when the **task** starts by choosing the "**Evaluate condition when activity becomes active (times will not be affected, scheduling will be done by using these efforts)**" option.

Request Center evaluates the condition at the beginning of each task. If the condition is false, the corresponding task will not be executed. Durations for any task configured with this option will be used to calculate the due date upon submission.

Re-evaluate expressions as plan advances

The Re-evaluate Expressions feature is useful for designs in which there are multiple sequential authorizations or tasks. It enables the person performing a task to enter information in the service form that is used to recompute the expression used to assign the performer for a subsequent task. If this option is not checked, all information used in expressions in the authorization task must be present during the Ordering moment.

This feature allows dynamic assignment of a task to a user (person or queue) and dynamic adjustment of the task title. Once the task becomes active, the expression is evaluated and the task is assigned appropriately.

Tips and Techniques

Use a condition that always evaluates to false (for example, "1=2") in a conditional statement to specify a task that will automatically be skipped.

The most common use cases for this are:

- When a service needs to "auto-complete" without having any tasks completed. The skipped task is the only task in the delivery plan. When it is skipped, the requisition is marked as complete.
- When an email needs to be sent without having to complete a task or when multiple emails are needed during a given moment in the following ways:
  - Create a parent task. On the Email subtab, choose an email to be sent out at completion. (You may also configure a notification to go out when the activity becomes active.)
  - Create a child task with the condition 1=2. Set the condition to evaluate when the activity becomes active.

Graphical Workflow Designer

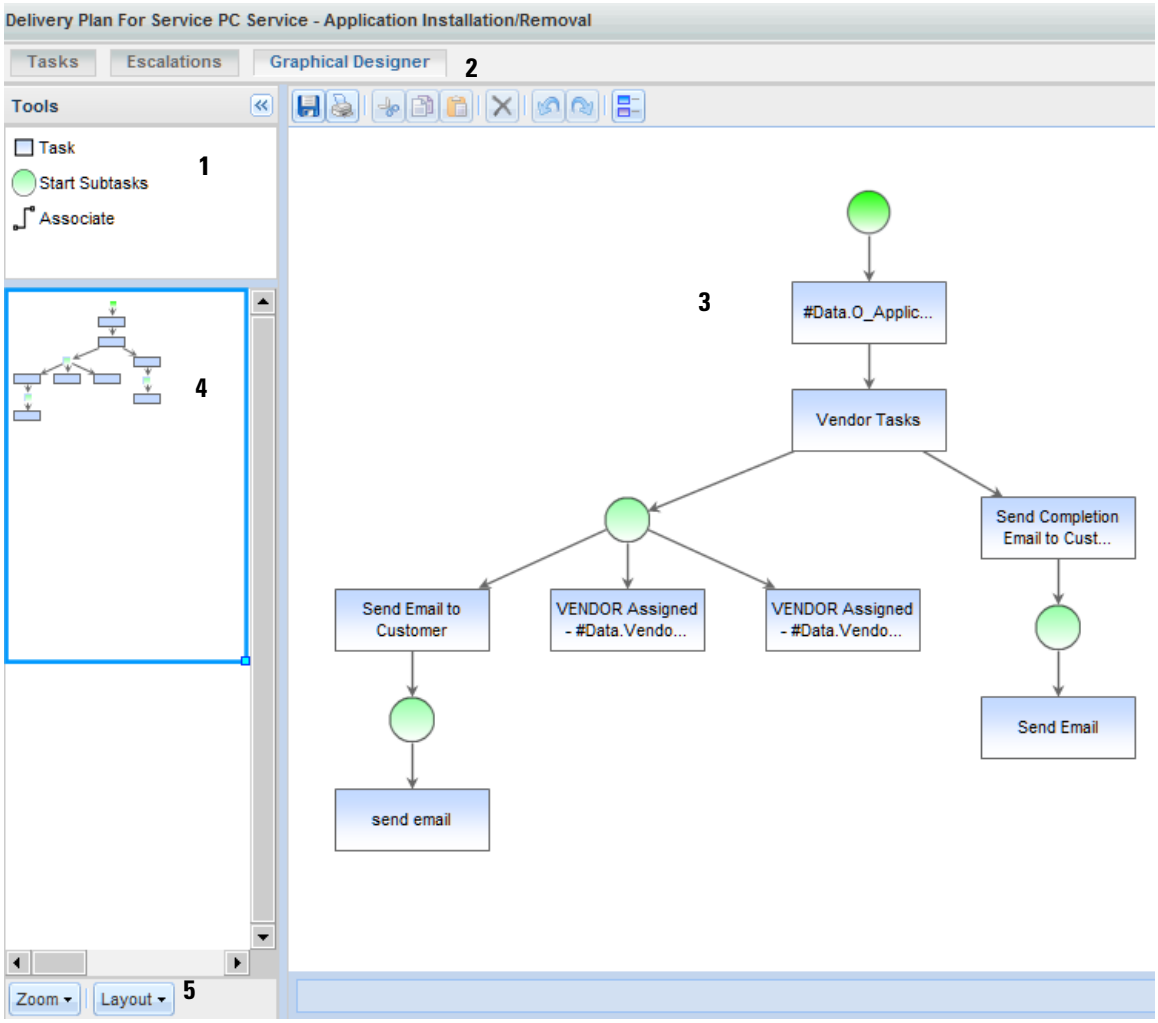
The Graphical Workflow Designer allows service designers to drag and drop tasks, name them, and connect them. The Graphical Designer also provides the ability to define parent/child relationships among tasks. An accompanying Task Properties sheet allows specification of additional details about the task definition and execution.

The Graphical Designer provides a visually oriented alternative to designing the service's delivery plan by using the dialog boxes provided in the General subtab for the delivery plan. Designers can define tasks and specify the workflow between them using either method, and freely switch back and forth between the Graphical Designer and the dialog boxes.

The Graphical Designer provides a simplified view of the delivery plan. To configure the more advanced details of a plan (such as those available on the Participants, Email, Task Instructions, and Checklist subtabs), switch to the Plan tab.

Overview

The Graphical Designer has the following sections:



1	Tools Pane	**4**	Outline Pane
2	Toolbar	**5**	View Tools
3	Work Area		

- **Tools Pane**, at the top left, contains building blocks of the workflow designer. These building blocks can be dragged and dropped into the work area and connected to create a complete workflow.

- **Toolbar** contains the buttons for saving the delivery plan; printing the plan; and manipulating the contents of the work area.
- **Work Area** holds the workflow diagram. When the Designer is invoked for a new delivery plan, the work area is blank except for the bright green circle which denotes the start of the plan.
- **Outline Pane**, in the middle of the left side of the page, gives a high-level overview of the diagram.
- **Zoom/Layout** buttons, the View Tools, at the bottom left, allow the designer to magnify or reduce the size of the workflow diagram or to alter the diagram's orientation.

Graphical Designer Toolbar

The menu bar at the top of the work area includes the options summarized below.

Option	Explanation
	Save the current workflow. In order to save a workflow, the design must be valid. A valid workflow has one starting point, and all tasks are connected, starting from the initial task.
	Print the current workflow. A printer-friendly version of the workflow opens in a new window. Use the standard browser Print command to print the workflow, closing the window when you are done.
	Cut the selected object or objects from the workflow diagram.
	Copy the selected object or objects. Only those attributes of a task which appear on the Graphical Designer's property dialog box are copied.
	Paste the previously copied or cut object or objects onto the work area.
	Delete the selected object or objects from the work area. If you delete tasks or associations, be sure to reconnect the remaining tasks in order to save the workflow.
	Undo the most recent change to the workflow.
	Redo the most recent change to the workflow.
	Display a legend explaining the icons used to represent different object types in the workflow.

The legend lists the color coding used to represent objects included in a workflow diagram:

Icon	Explanation
Start Plan	The diagram always has one "Start Plan" icon, which appears at the upper left of the diagram.
Start Subtasks	A group of subtasks must be children of a "Start Subtasks" icon.
Internal Task	A task performed within Service Manager has a workflow type of "Internal".

External Task	A task defined within the Service Link module for execution by an external system.
Service Item Task	A task defined within Service Link to create, delete, or update a service item.
Embedded Workflow	Within a bundled service, the workflow for each child service.

Tools Panel

The Tools Panel includes tools for adding the basic types of objects that comprise a workflow to the diagram.

Option	Explanation
Task	A workflow consists of a series of tasks, arranged in sequence. Use the Task tool to add a task to the workflow.
Start Subtasks	Tasks can be grouped as child tasks to a parent task.
Associate	Use the Associate tool to specify the sequence of tasks and to associate parent with child tasks.

In addition to the options in the Tools Panel, context-sensitive task cursors are available to help manipulate the content of the diagram. As you move the mouse over tasks previously placed on the diagram, the cursor changes shape, to indicate which actions are available:

Cursor	Explanation
(select)	Once an object has been selected, the select-cursor appears. Any objects marked with the select cursor are subject to the next Copy, Move, or Delete command.
(connect)	The connect cursor allows you to associate the highlighted task with another task. With the connect cursor displayed, drag the mouse to the task to be connected to the current task. When you release the mouse, a task sequence (association) is established.

View Tools

The Zoom and Layout buttons allow you to manipulate the size of the diagram and its orientation. The diagram can be oriented vertically with the first task at the top, and subsequent tasks below, or horizontally with the first task to the left, and subsequent tasks shown to the right.

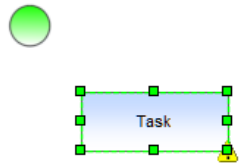
Creating a Delivery Plan via the Graphical Designer

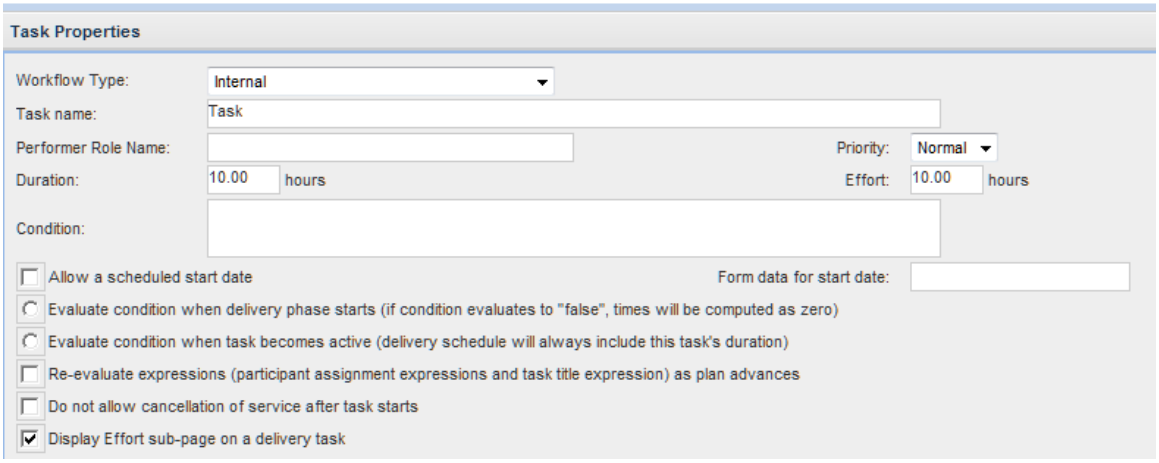
To create a delivery plan using the Graphical Designer, edit the service in Service Designer, click the **Plan** tab, then click the **Graphical Workflow** subtab. The work area appears empty except for the “Start Workflow” icon.

Defining a Task

The first step would typically be to place a task on the workflow. Click the **Task** tool (from the Tools panel) and drag it onto the work area. A task appears. Double-click the task to display the Task Properties dialog box at the bottom of the work area.

Figure 1-1 Task Properties – Yellow Alert Icon





The Task Properties window includes all data shown in the General subtab for the task as well as the Performer Role Name, which is especially useful in design sessions. Detailed explanations for the fields here are given in the "General Subtab" section on page 1-33. You can start defining the workflow for the task here, but will need to use the Plan tab subtabs for each task, to complete the task definition.

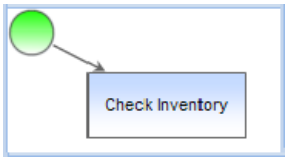
Critical aspects of defining the task are to assign a Task Name and choosing the Workflow Type. You can create a new "Internal" task (a task to be performed wholly within Service Manager); designate that a previously defined Service task (external task specified via a Service Link agent); or configure a Service Item Task (task to create, update, or delete a service item specified via Service Item Manager) to be included in the workflow.

As soon as you move from the Task Properties dialog box back to the diagram, the task name is displayed and the Task icon in the diagram may change, to reflect the workflow type. You may return to the Task Properties dialog box at any time by double-clicking the task.

Specifying Workflow

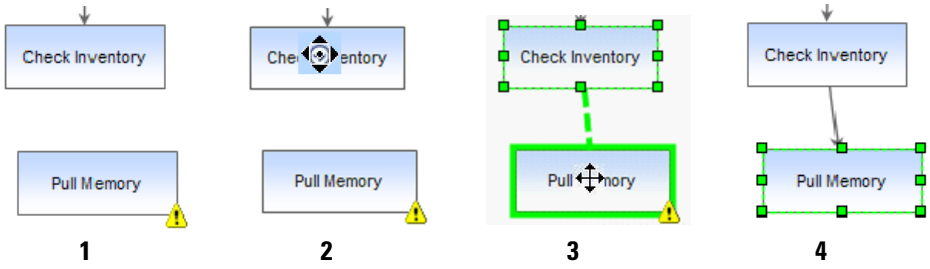
The yellow alert icon warns you that the diagram in its current state is not valid (Figure 1-1). You can hover over the alert icon to get a description of the problem, but in Figure 1-1 it is pretty obvious—the task needs to be associated with the "Start Workflow" icon, and assigned to its appropriate sequence within the workflow.

To place the task within the workflow, click the **Associate** tool. Then click the initial item in the workflow (in this case, the "Start Workflow" icon) and drag the mouse to the subsequent item (in this case, the first and only task on the diagram). When you release the mouse, an arrow is added to the diagram, showing the flow from "Start Workflow" to the first task in the workflow.



A more convenient method for connecting tasks is to use the Connect Cursor:

1. Add another task to the diagram and, if desired, give it a name, for example, "Pull Memory". This task is next in the workflow, after "Check Inventory".
2. To associate the tasks in this sequence, move the cursor to the middle of the first task (Check Inventory) until the connect cursor appears.
3. Drag the mouse to the second task. A thick dotted line appears between the tasks and the target task is highlighted in a thick solid line. The select cursor appears in the second task.
4. Release the mouse. An association is drawn and the diagram is valid.



Parent Tasks and Subtasks

Tasks may be grouped for several reasons. For example:

- Some of the tasks are conditional, and you want to send an email notification when the first or last task in the series completes, regardless of which specific task is executed.
- A task should be performed concurrently with a second task. In computing the Service Level Agreement (SLA) for the service, you do not want to count both tasks, just the one that would take the longest.

To account for these and other scenarios, Request Center supports grouping tasks. A parent task may have one or more children or subtasks. The subtasks may execute either consecutively or concurrently.

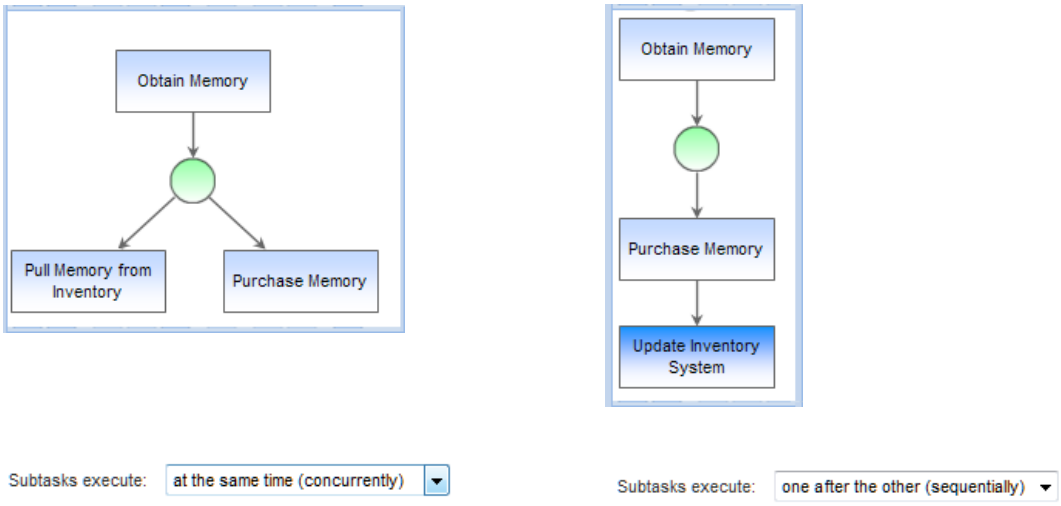
To create subtasks:

Step 1 Drag the Start Subtasks icon onto the work area in the appropriate sequence.

Step 2 Drag task icons onto the work area underneath the Start Subtask icon. Define the tasks as appropriate.

Step 3 To indicate that the tasks execute concurrently, move the cursor to the center of the Start Subtasks icon, until the connect cursor (with the cursor handles) appears, then drag the mouse to each subtask. Each are connected to the Start Subtasks, as shown in the left diagram, below.

Step 4 To indicate that the tasks execute consecutively, use the connect cursor to connect the Start Subtasks icon to the first task. Then connect the first subtask to the next. The resultant diagram will look like the example on the right, below.



Saving a Workflow

You can only save valid workflows. If any alert icons appear on the diagram, it indicates the workflow is not valid. You can hover the mouse over the alert icons to get a detailed description of the problem; it will usually be that the elements of the diagram are not properly connected to each other. Once the error has been corrected, click **Save** again to save the workflow.

Whether you use the Graphical Designer or the Plan tab dialog boxes, Request Center saves the workflow exactly the same way. This approach is good, in that you can use whichever tool is most convenient for you. But Request Center does not save the diagram itself—it reconstructs it from the delivery plan previously saved, and then adds to that delivery plan as you continue to work on the diagram. You can customize the appearance of the workflow for printing, for example, by moving tasks or increasing the size of a task icon, to show a longer description. However, any such changes are removed when the diagram is saved, and the workflow reverts to its default layout (horizontal or vertical).

Printing the Diagram

Click the **Print** option from the toolbar to print the diagram. The printer-friendly version will reflect the current view of the diagram, including its scale and orientation. If required, the diagram can span multiple pages. Use the standard browser **File > Print** command to print the diagram.

It is possible to make manual changes to the diagram, for example, to change the size of an individual task icon. If such changes have been made, they are reflected in the printer-friendly version. However, all such changes are temporary, and are not saved when the workflow is saved.

Graphical Designer vs. Plan Tab Dialog Boxes

There are certain things the Graphical Designer does superbly, much more efficiently, in fact, than using the dialog boxes in the Plan tab. For example, it is much easier to change the sequence of tasks, simply by reconnecting one task to another, in the Designer than it is by moving rows of data up and down.

The Graphical Designer includes an option to print the diagram. You will need to compare that printed version to the service preview available in Catalog Deployer, which includes the detailed workflow in textual format, to see which better suits your needs. Perhaps a user presentation will call for the printed diagram, while developer documentation is better served with the Catalog Deployer preview.

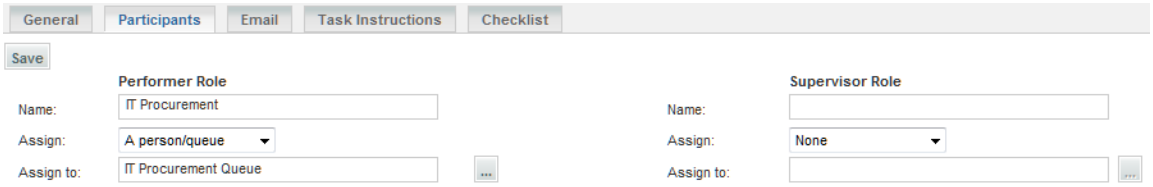
For specifying the general task properties, designers can use either the Plan tab or the Graphical Designer. Most of the same properties appear on both the Task Properties sheet in the Graphical Designer and the General subtab for the task. However, the General subtab dialog boxes have the following advantages:

- Although you can enter a conditional expression and form data for scheduled start date in both places, only the General subtab dialog boxes include the option to Validate the expression or namespace entered. Validating at design time is generally preferable to seeing the error for the first time when you test the form.
- The way you connect the subtasks to the Start Subtask icon determines whether the parent task is defined so that “subtasks execute sequentially” or “subtasks execute concurrently”. It may easier to simply change the “Top Level Tasks Execute” attribute of the parent task definition in the General subtab than having to reconnect all the subtasks in the diagrammer.

Of course, you will always use the Plan tab dialog boxes to complete the definition of the delivery plan, to specify the participants, task instructions, email notifications, and checklists for tasks as required.

Task Participants

On the Participants subtab, you specify who completes the task and, optionally, how the work is supervised. Click **Save** to save your changes on this tab.



Task Performer

The task performer is the person, queue or position to whom the task is assigned.

Field	Description
Name	Name of the Performer Role, meaning the entity who will perform the task. This name appears in the "By" column in the task list in the Plan tab.
Assign	Choose how the performer of the task is assigned: • **From a position:** The person currently filling a functional position defined in Organization Designer. • **A person/queue**: People or queues as defined in Organization Designer. • **From an expression**: A conditional expression shown in the "Assign to" field. See "Assigning Roles Based on an Expression" section on page 1-52. • **None**.
Assign to	• If you choose "From a position" or "A person/queue", click the "**…**" button to bring up a select dialog box to choose who the task is assigned to. Click **Add** or **OK** to save your selection. • If you choose "From an expression" click the "**…**" button to bring up the Edit Expression dialog box to enter the expression to be evaluated for assigning the task. Click **Set Expression** to save your selection.

Tasks are typically not assigned to a particular person, but rather to a queue or functional position. In this way, multiple people are available to work on the task, and it will not be delayed if one person is unavailable.

Task Supervisor

The task supervisor provides a measure of control over task performance. When combined with the Administration setting "Allow Task Supervisor to cancel task", it allows the designated person, queue, or functional position to cancel (skip) the task.

Field	Description
Name	Name of the person in the supervisor role, meaning the supervisor of the person who will perform the task/deliver the service.

Assign	Choose how the supervisor of the task is assigned: • **From a position:** The person currently filling a functional position defined in Organization Designer. • **A person/queue**: People or queues as defined in Organization Designer. • **From an expression**: A conditional expression shown in the "Assign to" field. See "Assigning Roles Based on an Expression" section on page 1-52. • **None**.
Assign to	If you choose "From a position" or "A person/queue", click the "**…**" button to bring up a select dialog box to choose who the task is assigned to. Click **Add** or **OK** to save your selection. If you choose "From an expression" click the "**…**" button to bring up the Edit Expression dialog box to enter the expression to be evaluated for assigning the task. Click **Set Expression** to save your selection.

Assigning Roles Based on an Expression

As with authorizations, the service team or person assigned to tasks in the delivery plan may depend on data on each requisition, such as a customer's location. Expressions can be used to intelligently route tasks, rather than creating different forms or workflows for each possible scenario.

Performer, Supervisor, and Project Manager roles can all be dynamically assigned.

To assign from an expression, choose that option from the Assign drop down and next to the "Assign to" field click the "**...**" button. The Edit Expression dialog box appears to allow you to enter the expression. The expression must reference a namespace (or field on the service form) that uniquely identifies a person, either via that person's ID (assigned in Organization Designer) or login name. An expression for a person's full name is also available but should typically not be used, in case two users of the same installation have the same name.

In the dialog box, click **Set Expression** to save the expression, or **Close** to close the Edit Expression window without setting the expression. Click **Save** to save your changes on this tab.

Email

Several events are associated with each task. By associating an email with each event, you can notify the service's customer, task performer, or other groups or individuals of the current status of the request.

Use the Email subtab to specify which notifications should be sent in response to which event. The notification must be defined using the Notifications component of the Administration module. Although Request Center is shipped with some default email templates, most sites prefer to design custom notifications.

For each task in the delivery plan (and for each authorization or review), the designer can choose the number of tiers in the escalation structure to be used by that task.

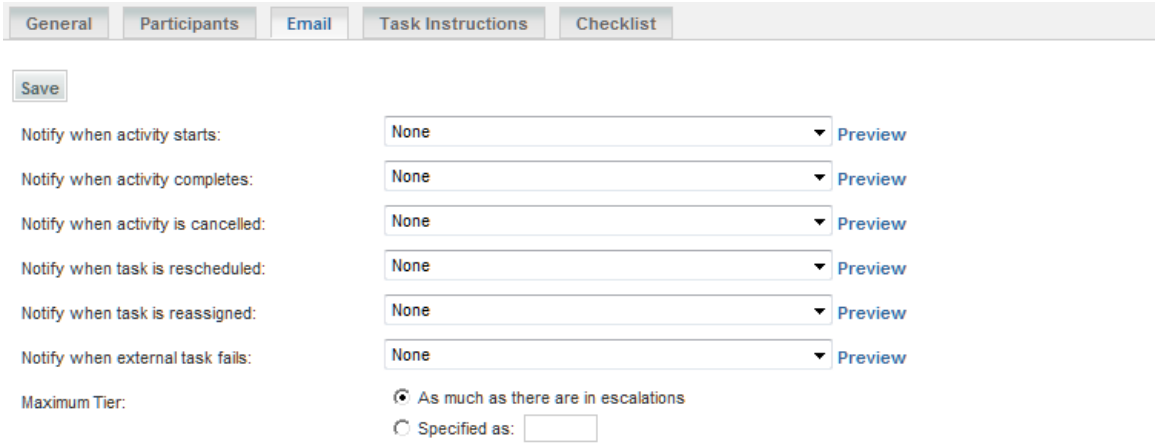
For example, you might configure three escalation tiers:

- Tier 1: 1 hour after the task becomes late
- Tier 2: 8 hours after Tier 1
- Tier 3: 16 hours after Tier 2

The Maximum Tier setting determines how many of the escalation tiers are to be used by that particular task, starting with the first tier.

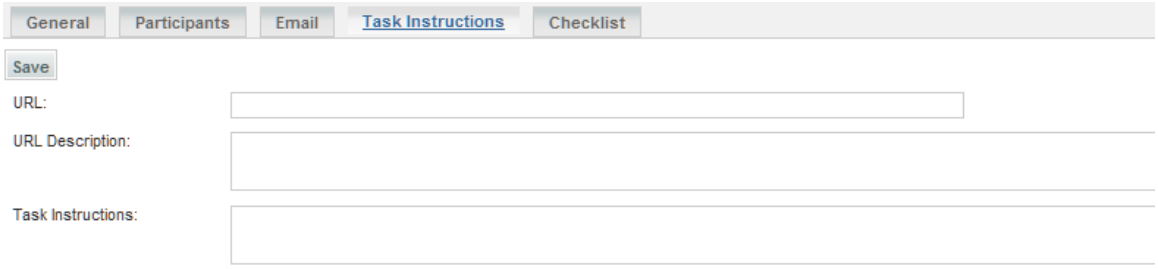
- As much as there are in escalations = all tiers
- Specified As = the number of tiers to use for this task; any number less than or equal to the number of tiers defined.

In this case, for example, if the maximum tier is Specified As 2, notifications would be sent 1 hour and 9 hours after the task becomes late; the third tier escalation would not be applied.



Task Instructions

On the Task Instructions subtab, you can give instructions for the task or set the link to any task instruction-related URLs.

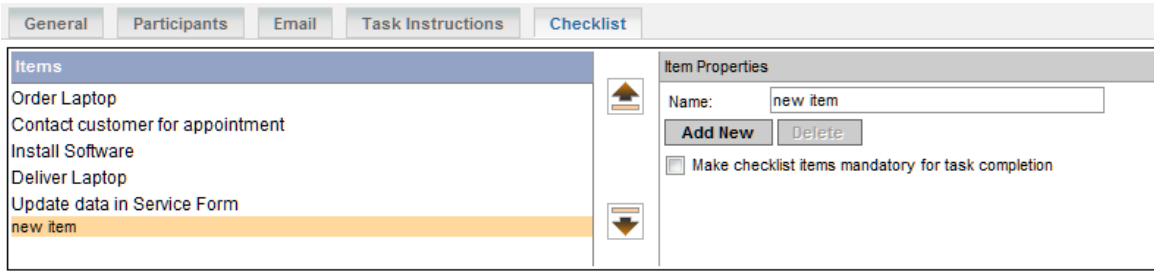


Field	Description
URL	Enter the full URL beginning with “http://” you wish to link.
URL Description	Enter a brief description to explain what the user will see when they link to the URL, or to include HTML in the checklist item. For example: Click Here for more information on how to complete your task.
Task Instructions	Use this text field to enter task instructions, if desired.

The HTML editor can be used to introduce HTML into the URL Description and Task Instructions fields.

Checklists

On the Checklist subtab, you can create a list of reminders or detail the specific procedural steps for completing a task. These appear in Service Manager as a checklist, detailing procedural steps for completing the work. Checklists do not affect due dates and are not reportable.



Field	Description
Items	The items in this list comprise the checklist the performer sees in Service Manager. Use the **Up** or **Down** Arrows to rearrange the item order. Items at the top should be performed first.
Name	Enter the name for the checklist item.
Make checklist items mandatory for task completion	Check this option to make all the items mandatory for completion of the task. This means the person performing the task will not be able to close out the work for the task until they've checked all of the items in the checklist in Service Manager.

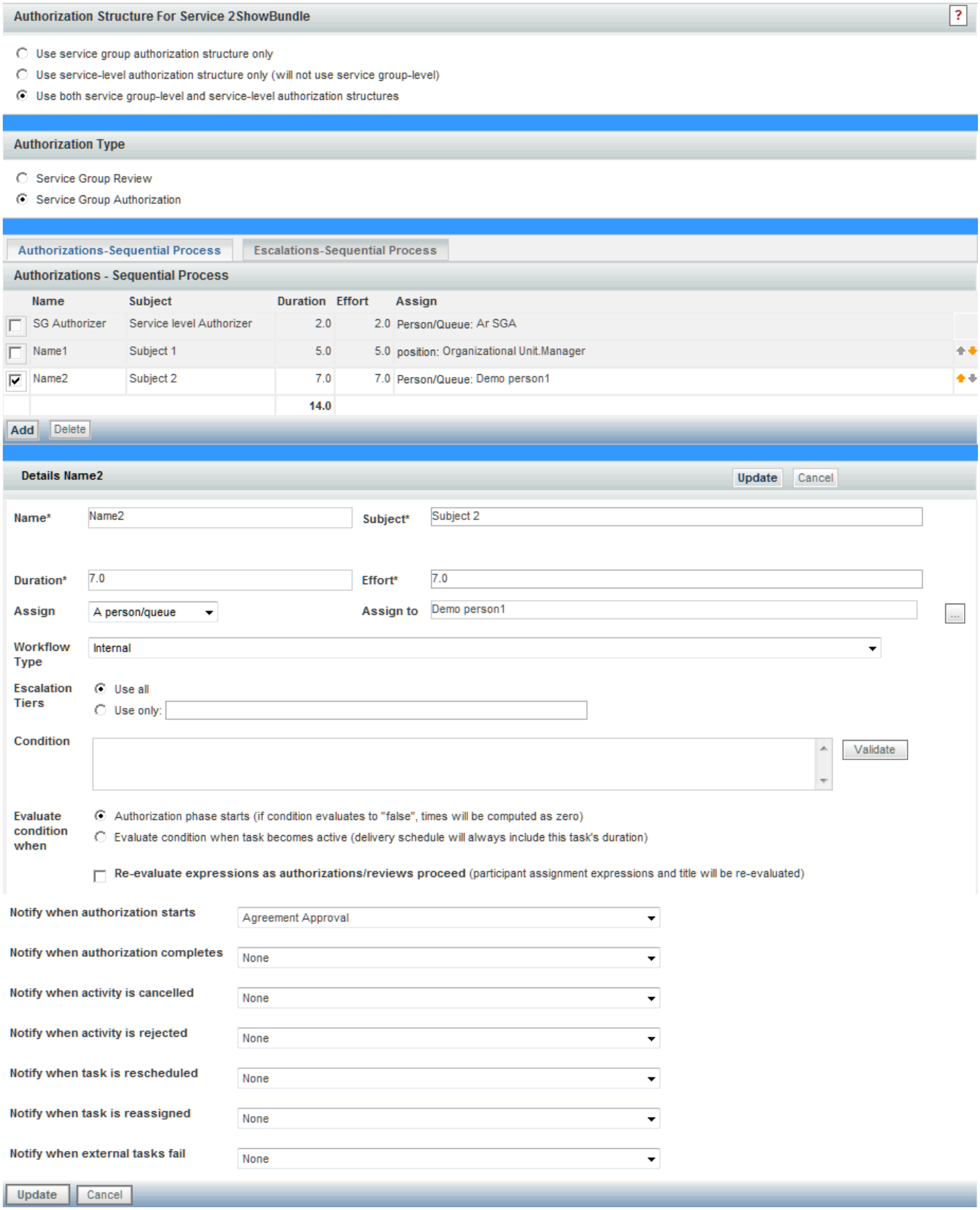
You can also:

- Add new items to the list by entering the checklist item in the Item Properties box to the right and clicking **Add New**.
- Remove an item from the list by highlighting the item and clicking **Delete**.

Authorizations

The Authorizations tab is used to configure authorization, review, and escalation tasks for the service and to specify the order in which these occur.

Using the Authorizations tab, you can also customize the escalation tiers, recipients, and email messages for late authorization/review tasks for a service.


Authorization Structure For Service 2ShowBundle
Use service group authorization structure only
Use service-level authorization structure only (will not use service group-level)
Use both service group-level and service-level authorization structures
Authorization Type
Service Group Review
Service Group Authorization
Authorizations-Sequential Process
Escalations-Sequential Process
Authorizations - Sequential Process
Name Subject Duration Effort Assign
SG Authorizer Service level Authorizer 2.0 2.0 Person/Queue: Ar SGA
Name1 Subject 1 5.0 5.0 position: Organizational Unit.Manager
Name2 Subject 2 7.0 7.0 Person/Queue: Demo person1
14.0
Add
Delete
Details Name2
Update
Cancel
Name*
Name2
Subject*
Subject 2
Duration*
7.0
Effort*
7.0
Assign
A person/queue
Assign to
Demo person1
Workflow Type
Internal
Escalation Tiers
Use all
Use only:
Condition
Validate
Evaluate condition when
Authorization phase starts (if condition evaluates to "false", times will be computed as zero)
Evaluate condition when task becomes active (delivery schedule will always include this task's duration)
Re-evaluate expressions as authorizations/reviews proceed (participant assignment expressions and title will be re-evaluated)
Notify when authorization starts
Agreement Approval
Notify when authorization completes
None
Notify when activity is cancelled
None
Notify when activity is rejected
None
Notify when task is rescheduled
None
Notify when task is reassigned
None
Notify when external tasks fail
None
Update
Cancel

Authorization Structure

For each service you can choose from three options:

- **Use service group authorization structure only**: The service inherits the authorization structure defined for the service group (specified in **Administration > Authorizations**); no additional configuration is expected. See the Site Administration chapter of the *Cisco Service Portal Configuration Guide* for more information.
- **Use service-level authorization structure only (will not use service group-level)**: Any authorizations defined for the service group are ignored. Only authorizations configured for the service are applied.
- **Use both service group-level and service -level authorization structures**: Accepts the authorization structure defined for the service group, and allows you to supplement it with a customized scheme for each service.

Authorization Types

There are two service group authorization types:

- **Authorizations** give the approver the opportunity to determine if the person requesting the service is eligible to receive it. If an authorization is rejected, the process stops and the service is not delivered. If multiple authorizations are defined, they are performed sequentially in the order in which they are specified—one authorization cannot be started until the previous is approved.
- **Reviews** are for information only. A reviewer cannot reject a service or cancel the delivery process, only indicate that he/she has reviewed the service. If multiple reviews are specified, they are performed concurrently. However, the delivery process will not proceed until all reviews have been completed.

If you see the message "not currently enabled via the Administration module," then the particular service authorization/review is not enabled for your site. This can be changed in the Administration module.

Authorizations and Reviews Subtabs

The Authorizations and Reviews subtabs are where you determine who should authorize or review the service. You can set up unique roles and define the order in which these roles review and approve requests. Based on the authorization type you choose, either the Authorizations – Sequential Process or Reviews – Concurrent Process subtab appears.

Note The Up and Down Arrow buttons ( ) to the right of each role allow you to move the role up or down in the approval process.

Note Click **Update** to save changes.

The table below defines the fields on the Details screen which appears after you click **Add**, or choose a previously defined authorization/review role by checking the check box to the left of the Name field.

Field	Definition
Name	Name given to the authorization role. This role name does not appear anywhere else in the application.
Subject	The title for the authorization/review task that this role performs. The entry here appears in the My Authorizations portlet that authorizers and reviewers see in My Services.
Duration	The published amount of time (in hours) to perform the review or authorization. This field exists for the purpose of letting you add some time to the amount of time that the authorization really takes. For example, you estimate that an authorization in the service group that you are configuring takes 0.5 hours. However, you want to add some time to the "official" estimate displayed in My Services. Therefore, you enter 1.0 hours in this field, and 0.5 hours in the Effort field.
Effort	The actual amount of time expected to be expended on this authorization.
Assign	Options to assign the default reviewer/approver from: • From a position – A person in a functional position. • A person/queue – A specific person or queue. • From an expression – A conditional expression shown in the Assign to: field.
Assign to	Specific person, queue, title of functional position, or conditional expression, depending on what you chose in the "Assign" field. Use the "…" button to choose the specific person or functional position, or enter an expression directly into this field.
Workflow Type	The default setting for the Workflow Type drop-down is "internal," which means that the task is performed manually by a user. To set the task so that it is performed as an external task in a third party system, use the drop-down list to choose the appropriate action from this list (see the "Configuring Authorizations for Use with Service Link" section on page 1-59.)
Escalation Tiers (option button)	Choice for tiers on the Escalations subtab (see the "Escalations" section on page 1-10): • Use all Indicates that all escalations set up on the Escalations subtab are to be used. • Use only (where a number (X) is entered from 0 to 99) Indicates that X tiers of escalations set up on the Escalations subtab are to be used. For example, if you have set up the Escalations tab with three tiers of escalations and choose the first option, all three tiers are implemented if the activity is later. If you choose the second option, specifying the number 1, only the first tier of three is implemented if the activity is late.

Field	Definition
Condition	Type in any expressions containing conditions that need to be met to fulfill the review. For example, if you enter the following: ActualCost<=1000, then anything that has an actual cost of less than or equal to $1000 is automatically approved by the chosen authorization role. See the "Syntax for Conditional Statements" section on page 6-20 for more details on creating conditional expressions. Click **Validate** to make sure that your expression is correct. Service Designer indicates if the syntax of the expression is correct or has errors. Note that this is a syntactical check only; the validate function does not check to see if the data you have referenced is actually in the system database.
Evaluate condition when authorization phase starts (radio button)	The condition set up in the Condition field becomes active as soon as the review phase starts.
Evaluate condition when task becomes active (radio button)	The condition set up in the Condition field becomes active after the review phase completes and the activity phase begins.
Re-evaluate expressions as authorizations/reviews proceed (check box)	Configure the system to re-evaluate expressions for the assignment of participants and title of task following each authorization/review task. This is useful if the participant or task title expressions change after initiating the task.
Notify when authorization/review starts	The email template that is automatically sent out when the review process begins; or choose "none".
Notify when authorization/review completes	The email template that is automatically sent out when review is completed; or choose "none".
Notify when activity is cancelled	The email template that is automatically sent out when the requester cancels the activity; or choose "none".
Notify when activity is rejected (authorization only)	The email template that is automatically sent out when the authorizer rejects the activity; or choose "none".
Notify when task is rescheduled	The email template that is automatically sent out when the task is rescheduled; or choose "none".
Notify when task is reassigned	The email template that is automatically sent out when a task is reassigned.
Notify when external tasks fail	The email template that is automatically sent out when external tasks fail.

Using the Site Authorization Scheme

You can use the site authorization scheme as set up in the Authorizations tab of the Administration module. This is the default setting, and often is the easiest to implement. See the Site Administration chapter of the *Cisco Service Portal Configuration Guide* for more information.

To use the site authorization roles and order scheme:

1. Click the radio button next to **Use service group authorization structure only** if it is not already chosen.
2. Click an item in the Area Authorizations and Reviews list to view the details of the site-defined authorization roles and escalations.

Configuring Authorizations for Use with Service Link

If your business process requires authorizations to be processed through a system other than this product, you can use Service Link to have an authorization task performed by that external system. The settings required to define an external task and how it should be handled appear on the Authorization tab—in the Details screen of a role.

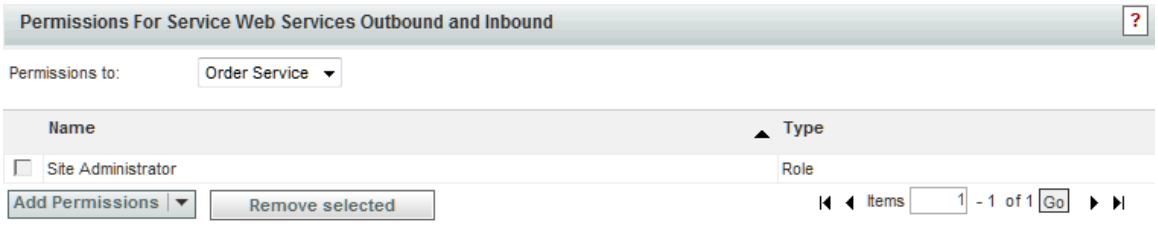
1. The default setting for the Workflow Type drop-down menu is "internal," which means that the task is performed manually by a user. To set the task so that it is performed as an external task in a third party system, use the drop-down list to choose the appropriate action from this list.
2. If for any reason the third party application does not successfully complete the authorization task, you might want to notify an administrator that a problem occurred in the Service Link integration for this task. In the "Notify when external tasks fail" menu, you can choose a template that will send such a notification.
3. If you have appropriate permissions to the Service Link module, you will see an ellipsis (…) button to the right of the Workflow Type drop-down menu. When you click this button, the Task Data Mapping dialog box appears.

This dialog box includes settings that show how the data to be sent to the external system is mapped to data on the service order form, or elsewhere in the system. If you have permission to change these settings, the dialog box is editable. Otherwise, it is read-only.

Now you are ready to set up escalation emails. See the "Escalations" section on page 1-10 for details.

Permission to Order a Service

The **Permissions** tab shows which participants are allowed to order this service.



To grant permission to order a service:

Step 1 Open the **Permissions** tab of a service you wish to allow participants to order.

"Order service" is the only permission that can be assigned at the service level; it is already displayed.

Step 2 Click **Add Participants** to add People, Organizational Units, Functional Positions, Groups, Roles, or to allow access to Anyone.

Step 3 Search and choose the entity or entities you wish to add.

People, organizational units, functional positions, groups or roles previously created/defined in the Organization Designer module (or via Directory Integration if this is implemented) can be searched and chosen.

Step 4 Click **Add** to grant the chosen entity permission to order the service.

Granting access to "Anyone" allows any Request Center user to order this service. Only services that are truly universal in nature, and open to the entire customer base, should have Anyone as a grantee. The "Site Administrator" role, by definition, has permission to order any service.

Creating a Scheduled Start Task

By default, a delivery plan becomes active (and the clock starts ticking on whether the service performers are completing tasks on time) as soon as all authorizations for the service request are completed, or, if there are no authorizations, as soon as the service request is submitted. Sometimes, this is not the desired behavior. For example, you may enter a request on behalf of a new employee scheduled to start work in a few weeks; or a development team may request a new server for a project scheduled to ramp up in the future. For such cases, Request Center allows you to create a Scheduled Start task, which does not become ongoing (active) until the specified start date is reached.

You can specify that a scheduled start task is executed on:

- A date and time specified on the service form by the customer or by someone performing an approval or review.
- A fixed date and time that you specify in the service's design, on the Plan tab in Service Designer

To create a scheduled start task by using a field on the service form:

Step 1 In the service definition, go to the **Form** tab for active form components used in the service, and identify the dictionary field that will hold the data (for example, NewHire.StartDate). The field's data type can be either Date or Date and Time.

Step 2 If required, go to the Active Form Component that contains the dictionary field identified in the previous step. Click the Access Control and Display Properties tabs, and set the appropriate dictionary permissions and the HTML representation for the field to be used.

Step 3 Click the **Plan** tab for the service, and then identify or create the task that you want to delay.

Step 4 On the General subtab for the task, check **Allow a scheduled start date**.

Step 5 In the "Form data for start date" field, enter the name of the dictionary field you want to use to hold the date and time of the starting-point for the task. This is the same field you identified in Step 1. Use the following syntax:

- Data.*DictionaryName.FieldName* for a field from a dictionary in one of the service's form components.
- ParentData.*DictionaryName.FieldName* for a field from one of the parent service's dictionaries (in the case of a bundled service).

To create a scheduled start task by using a fixed date and time:

Step 1 Click the **Plan** tab for the service, and then identify or create the task that you want to delay.

Step 2 On the General subtab for the task, check **Allow a scheduled start date**.

Step 3 In the "Form data for start date" field, enter a date and time in the following format:

"YYYY-MM-DD HH:MI" such as "2006-12-23 13:30" (The quotes are required.) HH is in 24-hour time, GMT. For example, this task will begin on December 23 at 1:30 PM GMT.

If the start date specified for the scheduled start task is earlier than the earliest possible starting date (with respect to the rest of the plan), the system simply ignores the specified start date and treats the task as if it were not a delayed task. This is logged in the System History.

Alternatively, if the start date specified is not mandatory and the customer/approver does not specify a start date, Request Center treats the task as if it were not a delayed task.

System Behavior for Scheduled Start Tasks

The system maintains both a Scheduled Start Date and an Actual Start Date for each delivery-plan task. You can see these dates on a task form in Service Manager.

The system does not automatically start a scheduled start task after the preceding task is completed. The scheduled start task has a *Scheduled* status until the specified starting-point is reached. At that point, the system changes the task status to *Ongoing*, and the task is then available for processing by the assigned performer. The Scheduled Start and Started on fields are the same for a scheduled start task.

If the starting-point for a scheduled start task is defined from form data, and you use a Date field to define this point, the system sets the time to the working hours of the task's performer. For example, if the task is to be performed by the HR Group queue and that queue's working hours are from 8:00 to 16:00, the system sets the time to 8:00 on the date entered by the user.

The system schedules all delivery-plan tasks after the Service Delivery phase of the workflow process begins. (The Service Delivery phase begins after the Authorization phase completes.)

The system does not evaluate the starting-point specified for a scheduled start task until it is scheduling all the delivery-plan tasks at the start of the Service Delivery phase. You may want to take advantage of this when designing services, as it means that the starting-point may also be specified by the performer of an authorization or review step (as opposed to the customer who orders the service).

Entering an Invalid Date

As a service designer, you cannot control what the customer enters as the starting-point for the scheduled start task. For example, the customer might enter a future date that ends up occurring before preceding tasks in the service are completed.

The system evaluates the date for the starting-point when it schedules the delivery-plan tasks. If it determines that the starting-point specified for the task is earlier than the earliest possible starting date (with respect to the rest of the plan), it simply ignores the specified starting-point and treats the task as if it were not a delayed task. The system enters a comment in the System History field, indicating that the start date is ignored.

Similarly, if the starting-point is not a mandatory field on the service form, the customer might not specify a starting point for the scheduled start task. The system ignores the blank starting-point field, and treats the task as if it were not a scheduled start task.

Bundling Services

A bundled service is a service that contains one or more related services that are automatically ordered when a customer orders the bundle. Bundling is a convenient way to group services that are commonly ordered at the same time. For example, all new employees might require the following services:

- LAN ID
- Desktop or Laptop PC
- New Phone with Voicemail

Rather than expecting users to remember to order all three services for a new employee, you could create a bundle that contains each of these services.

In a bundle, the new service containing two or more related/existing services is referred to as the *parent service*. The services contained by a parent service are referred to as *child services*.

Creating and processing a bundle involves several different modules and tasks within Request Center. Here's a brief overview.

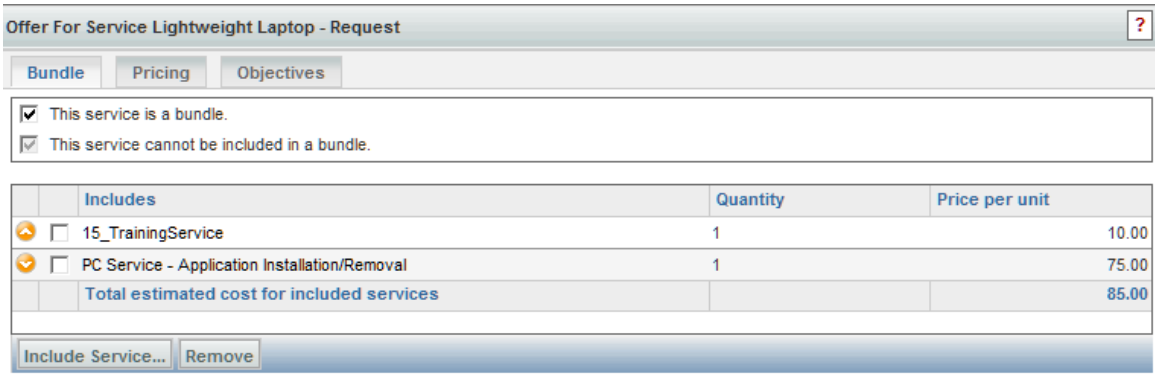
- **Create**. After creating service groups and services, service designers use the Service Designer module to create a bundle.
- **Order**. Customers use My Services to order a bundle. My Services users may not realize that a service is a bundle. For example, if they click the Order link when choosing the service from the services catalog, the composite order form for the bundle appears, and if they review the order form, only the parent service appears. However, if a user clicks the service-name link as it appears in the service catalog, they can then click links for each of the child services and see detailed information. They cannot however, order any of the child services this way. After a customer submits an order, the Service Order Confirmation page lists all services included in the bundle. If customers want to cancel a bundle, they must cancel the complete order; they cannot cancel individual child services included with the bundle.
- **Schedule**. Request Center treats the order like any other set of requisition entries on a requisition: It schedules
  - Authorization and review steps defined for the parent service. (The authorizations defined for the child services are not inherited by the parent.)
  - Delivery plan tasks for each of the services.
  - Plan-monitoring tasks for each of the services.
- **Deliver**. The delivery plan of each child service executes as if the child service had been ordered on its own. The scheduling of the delivery-plan tasks for the child service depends, however, on the position of the included task for the child service within the delivery plan of the parent service. For example, if the parent service lists the child services of LAN ID, Desktop, and New Phone, then the

tasks would be completed in that order. (This assumes that you chose to complete top-level tasks sequentially in the Delivery Plan. If you want the delivery plans for the child services to execute concurrently, you could change this setting.)

- **Monitor**. If Service Manager users monitor a plan, they can see the plan-monitoring task for any service on the Plan tab, in Service Manager. The plan-monitoring task for a parent service shows the included tasks for each of the child services. If the site configuration parameter "Show Task Link" (available in the "Personalize Your Site" folder in the Administration module) is set to On (its default), the tasks shown on the Plan tab are links to the actual task forms. For included tasks, these links take you to the plan-monitoring tasks of the included services.

Creating a Bundle

In Service Designer, begin by choosing the service you wish to use as the parent service or create a new service to act as the parent. Once you have chosen the service, click the **Offer** tab to get started. The assumption is that you have already created the child services that you wish to include in the bundle.



To create a bundle:

Step 1 From the service's Offer tab, click the **Bundle** subtab.

Step 2 Check **This service is a bundle**. The system automatically checks the "This service cannot be included in a bundle" option because Service Designer does not support bundled services within bundles.

The "Include Service" button becomes enabled.

After you add a service to a bundle (making it a child service), Service Designer disables the two check boxes and instead displays a list of the bundles in which the service is included.

Step 3 Click **Include Service**. The "Select a service" dialog box appears. This dialog box lists only services for which the "This service cannot be included in a bundle" option is not checked.

Step 4 Choose the service that you wish to include in the bundle, and click **Add Selected Services**. The child service appears in the Includes section on the Offer/Bundle subtab. The child services appear in My Services as part of the detailed information for the parent service (on the Included Services subtab).

Step 5 Repeat Steps 3 and 4 to add other child services to the bundle.

To change the order of child services:

Step 1 In the “Includes” section, check the check box of the service that you want to move.

Step 2 Click the Up- or Down-Arrow icon to move the service.

Preventing Bundling

If you want to prevent other service designers from including a service in a bundle, complete the following procedure. Services marked in this way do not appear in the “Select a service” dialog box when choosing child services.

To prevent bundling a service:

Step 1 Choose the service that you want to prevent from being bundled.

Step 2 Click the **Offer** tab, and then the **Bundle** subtab.

Step 3 Choose the **This service cannot be included in a bundle** option.

Reviewing Included Tasks

For each service you add to a bundle, Service Designer automatically inserts a task into the delivery plan of the parent service. This task is referred to as an included task.

New | Indent | Outdent | Up | Down | Delete

Task	By	This	Subtasks	Subtotal
Service Delivery		10.00	152.52	162.52
Create CAPEX Request	MAH - IT Procurement Queue	8.00	0.00	8.00
Received CAPEX Approval	MAH - IT Procurement Queue	24.00	0.00	24.00
Order lightweight laptop	MAH - IT Procurement Queue	16.00	0.00	16.00
Confirm Received from Supplier	MAH - IT Procurement Queue	40.00	0.00	40.00
Install Additional Hardware and Software	MAH - Service Desk - WA Queue	16.00	0.00	16.00
QA Check on lightweight laptop	MAH - Service Desk - WA	0.50	0.00	0.50
Create Laptop SI Details	Service Item Manager	0.01	0.00	0.01
Update Laptop SI Details	Service Item Manager	0.01	0.00	0.01
Package lightweight for Transport and Hand off to driver	MAH - Service Desk - WA	8.00	0.00	8.00
Confirm receipt of lightweight laptop	MAH - Confirm Delivery	40.00	0.00	40.00
Deliver Included Service 15_TrainingService	Subplan Manager	0.00	0.00	0.00
Deliver Included Service PC Service - Application Installation/Removal	Subplan Manager	80.00	0.00	80.00
			Total project duration	242.52
			Approximate days (as per working hours per day)	30.31

To review included tasks in a bundle:

Step 1 From the Service panel, choose the parent service.

Step 2 Click the **Plan** tab.

The child services are automatically named following this convention: "Deliver Included Service *service-name*." For example, a service might appear as "Deliver Included Service Phones".

Included service tasks behave like most delivery-plan tasks, with the following exceptions:

- You cannot delete an Included Service task. The Delete button on the Plan tab is disabled. To delete this task, remove the corresponding child service from the bundle.
- You cannot set Duration values. These are calculated as the total Duration from the child service's delivery plan.
- You cannot edit the Task Type. This is set to "Included Service" by default.
- You cannot create or associate checklists. There is no Checklist tab for the task.
- You cannot edit the information on the Participants tab.

Although the task is automatically named "Deliver Included Service *service-nam*e," you *can* change the task name if desired.

You can also arrange included tasks in a task/subtask relationship by indenting one under another. However, the delivery of both services will start at the same time.

If desired, you can make one of the subtasks conditional to effect an "opt-out" scenario. In this case, the child service still appears to have been ordered (with all tasks skipped).

To change an included task name:

Step 1 Click the task name. The General tab appears with information about the task. The "Task Type" indicates the task is an Included Service. The "Service Name" identifies the service that the included task represents. (This is important information if you change the name of the included task.)

Step 2 Enter a new task name in the "Task Name" field, and click **Update**.

Reviewing Included Participants

For each child service you add to a bundle, Service Designer automatically assigns participant information into the delivery plan of the parent service.

To review included participants in a bundle:

Step 1 Choose the parent service.

Step 2 Click the **Plan** tab.

Step 3 Click the **Participants** subtab.

The information on the Participants tab is taken from the Project Manager of the child service. Service Designer automatically assigns both the Performer and the Supervisor of the task using whatever is currently defined as the Project Manager of the child. If you change the Project Manager of the child service, the change is reflected dynamically for the included task.

Service Designer names the performer of the included task the Subplan Manager. It names the supervisor the Plan Manager.

Pricing Bundles

The price of a Bundle includes the price of each included service, as defined in each child service. You set the price of any service on the Pricing subtab of the service's Offer tab. The price could potentially be adjusted by using the Set Price action in an active form component rule, applied either to the bundle as a whole or to its component services.

For your convenience as you construct a bundled service, the price of each child service is listed on the parent service's Offer tab, Bundle subtab. If a child service's price is defined as Pricing required, the system schedules a pricing step for that service at the time a My Services user orders the bundle.

Discounting the Price of a Bundle

The price of a bundle can be discounted in either of two ways:

- Use dynamic pricing, via the Set Price action available in active form component rules, to reduce the price of a service when it is used in a bundle.
- Use the Pricing subtab of the parent service. There you can set a negative price for the parent service, which is subtracted against the total cost of the included services.

For example, if the cost of all child services adds up to $5,000 and you enter a negative price of $1000 for the parent service, the net cost for the bundle becomes $4000. By doing this, you encourage the end user to order the bundled service rather than each service individually.

To discount a bundle:

Step 1 From Services panel, click the parent service.

Step 2 Click the **Offer** tab, and then the **Pricing** subtab.

Step 3 Enter a negative value for the price. For example, to discount the service bundle by $1000.00, enter –1000.00.

Step 4 Click **Save All Settings**.

Customers can see that the price has been discounted only after they've clicked Order for the bundle. To let them know about the discount up front, you can use the service's Description field on the General tab, or the Description field on the Offer tab/Pricing subtab, to briefly highlight the financial advantage of choosing the bundled service.

Customer View of Bundles

When a customer orders a bundle in My Services, they click the service name to view the service details. On the Details page, they can click “Included Services” to see all services included in the bundle.

From there the customer can click on each service name to view the specific details related to that service. When the customer drills down to this level, a Return to Service Bundle button displays *in place of* the Order Service button. This means that a customer cannot order a child service from within a bundle unless they are ordering the entire bundle. To order services separately, the customer must return to the service catalog.

When the customer orders a bundle, the Order Confirmation page displays the names of all services in the bundle including the Due Date for each child service.

The Track Resolution page displays a list of all child services in the bundle. The customer can click a link to view the order form and to view the Delivery Process for each service as it is performed.

Can a customer cancel a bundle once it has been ordered?

A customer can cancel a bundle once it has been ordered. Canceling a bundle cancels all included services. The option to cancel individual child services is disabled for the end user on the Edit Requisition page.

The “Do not allow cancellation of service after task starts” check box on the service's Plan tab, General subtab in Service Designer essentially defines “the point of no return” for the customer to cancel a service. This option can be checked for the entire bundle or for individual child services within, but once this point has been reached for *any* of the services within the bundle, the customer cannot cancel the bundle.

Additionally, when “the point of no return” has been reached for any service within the bundle, the Cancel button is removed for the service order.

Exporting and Importing Bundles

You can export and import bundled services through Service Designer just as you can with any other service. When Service Designer constructs the XML for a bundled service, it generates the XML for the parent as well as for each of the child services within the single XML file. This means that you can export a bundle and import it into another catalog that does not already contain the child services. The export utility creates each service in the bundle as a separate service in the catalog, and will link the children to the parent service just as they were linked in the source catalog.

Data for Bundled Services

You can control what dictionary fields service performers see on the order form of a bundle via the “Show Bundle Data” parameter located in the “Personalize Your Site” folder in the Administration module.

When the system creates the composite order form for a bundled service, any duplicate dictionaries are eliminated. This means that if a dictionary is associated with a parent service and one of the parent's child services (or multiple child services), the system will display only the first instance of that dictionary on the bundled service order form.

There are two options to display service form data:

- **ShowBundleData = On** (default) shows service performers the composite order form for the bundle when processing work items within any of the child services.
- **ShowBundleData = Off** shows service performers only the dictionaries from the child service they are working on. The system ensures that the data for the bundled service duplicates the data entered for one dictionary in all instances of that dictionary. So if a dictionary from a child service was suppressed on the original order form (because it duplicated a dictionary in the parent service), any service performer processing just the child service will still see the data entered by the customer.

Namespace Variables for Bundled Services

The system provides a number of namespace variables that you can use when working with data in bundled services.

Namespace Variable	Data Type	Purpose/Usage
Service.Bundled	Boolean	True for a requisition entry if the service is a child on a bundle.
Service.IsBundle	Boolean	True for a requisition entry if the service is itself a bundle.
Service.BundledServices	Numeric	Indicates the number of child services belonging to a parent.
Requisition.Services	Numeric	Indicates the number of services in a requisition.
ParentService.Data.DictionaryName.FieldName		Syntax for referring to a data element within a bundle.

- If the value of the site configuration parameter **ShowBundleData** is **On**, ParentService.Data.*DictionaryName.FieldName* is equivalent to: Data.*DictionaryName.FieldName*.
- If the value of **ShowBundleData** is **Off**, ParentService.*Data.DictionaryName.FieldName* is the required syntax for referring to data elements in the parent service.

Managing Dictionaries

Commonly Asked Questions about Dictionaries

Peruse the questions below to gain a better understanding of the types of dictionaries available, the differences between them, and for examples of how they can be used.

What is the Reserved folder?

Every Service Portal instance automatically includes two dictionaries, provided as seed data for customer and initiator information. These dictionaries can be found in the “Reserved” dictionary group in the Dictionaries component of Service Designer.

There is also a “Reserved” form group in Active Form Components, which contains the Customer-Initiator form. This form includes the Customer_Information and Initiator_Information dictionaries.

Where does the personnel information come from when I use a person-based dictionary?

Request Center maintains a repository of all persons in the user organization who may need to access Service Portal applications. The personnel information in this repository is generally populated via directory integration. Any time a user logs in to Request Center, or has a service ordered on his/her behalf, their personnel information in the repository can be refreshed with their information from an enterprise-wide directory. Person information in Request Center is viewable via Organization Designer.

Why would I need to create a person-based dictionary?

If you are only referring to the request's customer or initiator, you don't have to create a person-based dictionary. Instead, use the Customer_Information and Initiator_Information dictionaries, already combined in the Customer-Initiator form.

However, requestors frequently need to designate another person who may be involved in the fulfillment of the request or needed as a reference. For example, a frequent use case for person-based dictionaries is when an initiator must designate one or more approvers for a request, if the approvers cannot be inferred from the supervisory structure available from the directory integration. In this scenario, the requestor selects the approver by searching through a drop-down list of available personnel and the approver's contact information is then included in the service form data for easy reference.

What is the difference between creating a person-based dictionary and a free-form dictionary containing a field of data type = Person?

Using the person-type field in an internal free-format dictionary is useful if all you want to do is choose a person and display **only his/her name** in a dictionary that includes other fields not related to that person.

A person-based dictionary automatically provides you with a field to display a person's name, a Select Person button and an invoked Person Search window, and the ability to do a person lookup, either into the integrated directory or into the Service Portal repository, depending on your system configuration. The form automatically displays any information on the chosen person that corresponds with fields in the person-based dictionary.

By contrast, a person-type field in an internal free-form dictionary displays a person's name and provides the same Search Person capability. However, once a person is chosen, only the person’s Name can be automatically displayed on the service form. Any additional data needs to be retrieved via a data retrieval rule, specifying a SQL statement, and mappings for each of the fields.

The “Select_Person” attribute provides the capability to search for people, either within the Service Portal repository or, if directory integration has been enabled for service forms, in an external directory. For example, the “Person” field on the service form below is the HTML representation of a “Select_Person” attribute.

Clicking the “Select” button brings up a “Person Search” dialog box which allows the user to specify search criteria and choose a single person. Once a person has been chosen, the search dialog box is dismissed, and any fields used in the dictionary are filled in with the current values of the corresponding fields in the chosen person’s profile.

What is the difference between using the Customer-Initiator form and a standard internal person-based dictionary?

Person-based dictionaries provide the mechanism for retrieving data on a particular person from the Service Portal repository and designating which aspects of the personnel data should be displayed on the service form.

The Customer-Initiator form is comprised of two internal person-based dictionaries, one each for customer and initiator information, respectively. The Customer and Initiator dictionaries supplement the standard behavior of automatically recording the customer and initiator for all requests. All information on the customer and initiator is available throughout the requisition life cycle via "lightweight namespaces," and is displayed as part of the Requisition Summary page for the request in My Services and Service Manager.

These dictionaries are automatically filled in on the service form based on values known to the system whereas in a standard person-based dictionary, you allow the requestor to choose a person, as shown above.

Designers can use the "lightweight namespaces" from the Customer-Initiator form in other dictionaries if desired. Thus, the Customer-Initiator form is an easy, partially preconfigured way to display person information, but not the only way.

What if I don't want to use all the fields in the Customer or Initiator dictionaries?

For each of the Customer and Initiator dictionaries, you can choose which dictionary fields appear on a service form within the Dictionaries component of Service Designer. Simply navigate to the Customer or Initiator dictionary in the Reserved dictionary group, and check or uncheck the check boxes in the "Use" column to determine which fields are used, or included, in the dictionary and on the Customer-Initiator form.

In addition to a customer or initiator's first and last name, and login ID, it is common to add fields for the person's email address, Home OU, and office location.

Once you have chosen fields for inclusion, you also have the option to **set dictionary fields to be read-only** on the service form.

What if I delete the Default Value for a field in one of the Reserved dictionaries?

The Default Value in a field in a Customer or Initiator dictionary allows the system to prefill a value on the service form.

If you accidentally delete the Default Value for a field within the Customer-Initiator form in Active Form Components, you must manually readd the default value. There is no automatic setting to restore default values. The value refers to a "lightweight namespace" that is tied to either Customer or Initiator information. For a complete list of such namespaces, see Chapter 6, "Namespace".

Fields must be chosen for use in the Customer or Initiator dictionary before their appearance and behavior can be modified. Go to the Display Properties tab of the Customer-Initiator form to manually re-enter the Default Value for a Customer or Initiator dictionary field.

How can I use the Extended Person fields in the Reserved dictionaries?

The short answer is: Any way you want to, but for a more detailed explanation, see the topic "Adding Extensions" in the *Organization Designer Online Help* and the discussion of Attribute Mapping in the *Cisco Service Portal Integration Guide*.

Essentially, Service Portal includes fields that provide an extension to the standard personnel data. Some of the most frequently required extended fields have been assigned meaningful names (such as Company Code and Division), but others have the names Custom 1 through Custom 10, and are intended to be

freely used, with no preconceived semantics. If you have additional personnel information in the LDAP directory that needs to be exposed in the application, map the attributes containing that information to one of the custom fields and check that the corresponding Custom field is included in the Customer or Initiator dictionary. If you have not mapped any directory attribute to one of the extended fields, you can include it in one of the Reserved dictionaries and then use a conditional or data retrieval rule to provide a value.

What is a Service Item-Based Dictionary?

A Service Item-Based Dictionary is a dictionary whose structure (fields) are based on a service item defined in the Service Item Manager module. For more information please see Chapter 3, "Lifecycle Center".

What is the Integration folder?

The Integration dictionary group is automatically created in all Service Portal instances. Any dictionary that is created through the Integration Wizard (Service Designer's wizard for creating web services integrations between Service Portal and external systems) is automatically placed in this group. Once created, integration dictionaries can be moved to any dictionary group. Designers cannot manually place dictionaries in this group.

Using Active Form Components in a Service

Active Form components specify the appearance and the behavior of the service form, the web page presented to users of Request Center when they request a service; authorize or review a previously entered service request; perform tasks to fulfill a service request; or review a completed service request.

Configuring an active form component is explained in detail in Chapter 2, "Active Form Components". This configuration consists of:

- Specifying the dictionaries that are included in the form component, and the order in which they appear
- Configuring the appearance of each dictionary and fields within the dictionary
- Configuring access control (view or edit permissions) to the dictionary in all moments of the service fulfillment life cycle
- Specifying conditional rules that affect the behavior or appearance of a service form
- Specifying data retrieval rules that allow data stored in an external relational database to be used within a service form

The current chapter supplements that information, adding configuration information relevant to incorporating an active form component into a service definition.

Adding Forms to a Service

Use the **Form** tab of the Services component to add active form components to a service and to specify the order in which forms appear on the service form. You can also review some aspects of the form component's usage and add to the dictionary permissions that have been defined for the form component.

The application prevents you from using the same dictionary twice on a single service form by presenting an error message if you attempt to add two forms that each contain the same dictionary. Once a form component is added to a service, it is removed from the search results so that the same form cannot be chosen twice.

It is best practice to include the Customer-Initiator form (containing the Reserved Customer_Information and Initiator_Information dictionaries) in every service, as documented in Chapter 2, "Active Form Components". The order in which this form appears in each service is up to the service designer.

The list of form components that is available to be incorporated into the service is restricted to forms in form groups for which the service designer has the permission to "View Forms" or "Design Forms".

Adding Form Components to a Service Definition

To add a form to a service:

Step 1 Navigate to the desired service in the Services component of Service Designer.

Step 2 Click the **Form** tab.

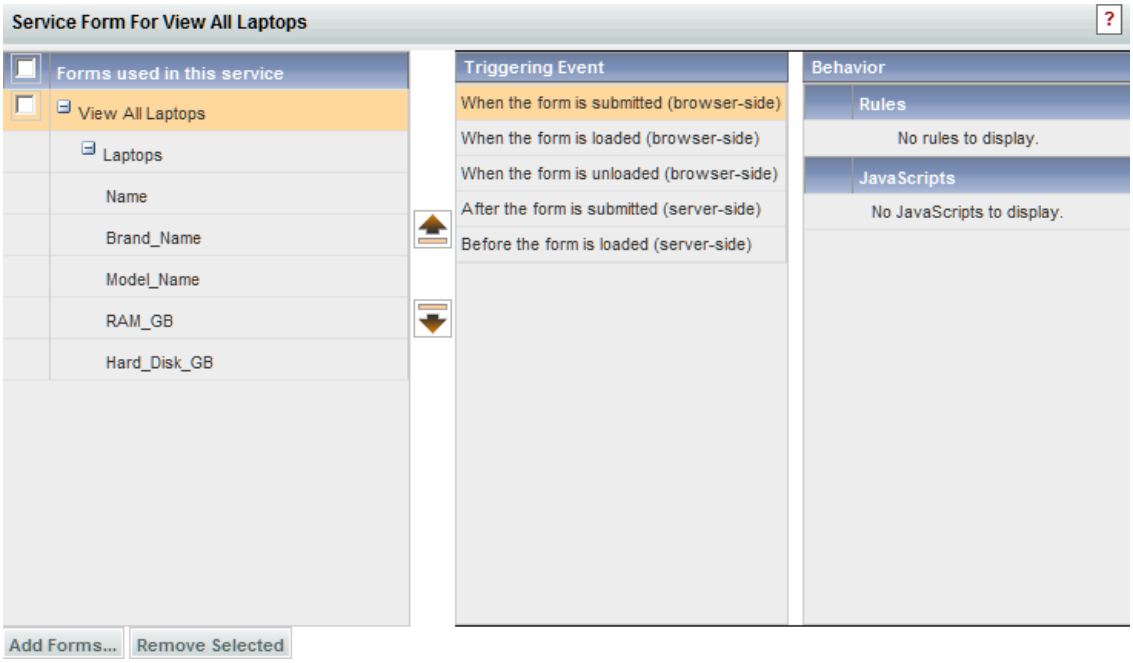
Step 3 Click **Add Forms**.

Step 4 Search and choose one or more form components to add to the service.

Step 5 Click **Add**.

Step 6 Any chosen forms are added to the service.

The application prevents you from using the same dictionary twice on a single service form by presenting an error message if you attempt to add two forms that each contain the same dictionary. Once a form component is added to a service, it is removed from the search results so that the same form cannot be chosen twice.



Changing the Order of Form Components in a Service Definition

To change the order that forms components appear on a service form:

Step 1 On the service's **Form** tab, check the check box to the left of the form you wish to move.

Step 2 Click the Up- or Down-Arrow icon to the right of the “Forms used in this service” pane.

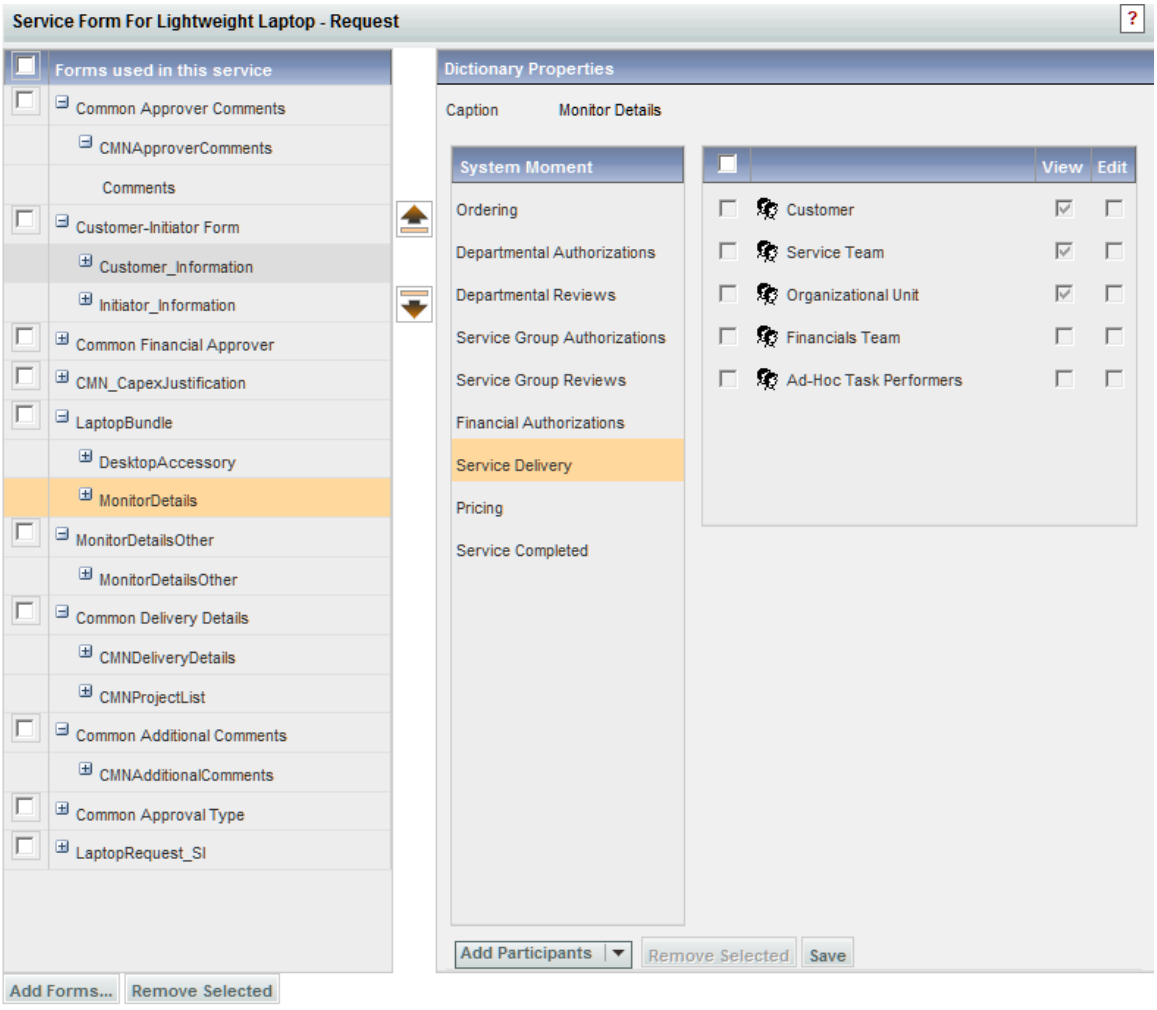
By changing the order of forms on a service, you change the order of form fields on the Form Content tab for the active form component. The fields on the service form are displayed in order by form component and, within each form component, in the Display Order specified for the field.

Similarly, the rules defined in a form component fire in the order they are specified within the form component, in the order in which the form components are sequenced in the service. In the event that you have created multiple rules which fire on the same dictionary or dictionary field, the last rule in the sequence of rules will take precedence.

Reviewing/Modifying Form Component Configuration

Once you have added a form component to a service, you can review portions of its configuration on the Form tab.

- When you highlight the form component name, you can view form component behavior at the form-level triggering events for conditional and data retrieval rules (for example, when a form is submitted).
- If you expand a form component node, the dictionaries in the form will appear. By highlighting a dictionary name, you can view rights for various performers (for example, the customer) at each system moment in the request fulfillment life cycle. You can also add participants to this access control; no other changes are allowed.



- If you expand a dictionary node, the fields that comprise the dictionary appear. By highlighting a field name you can view:
  - Input types (for example, text field, radio button) and display characteristics for each included dictionary field, on the General tab
  - Conditional rules, data retrieval rules or JavaScripts associated with the particular field, on the Behavior tab

The only modification that can be made to the form component's configuration on this page is to specify Additional Participants in the access control for a dictionary. This allows the form component's usage to be customized to the authorizations and tasks specified in this service.

To make any other modifications to what you see on a service's Form tab, you must navigate to the element to be changed. Use the mouse to hover over the element, then Ctrl-Click a dictionary, field, or form, as appropriate. When you are done, you can click the links at the bottom of the page ("Active Form Components using this Dictionary", "Services using this Active Form Component") to return to the service definition.

If an active form component is truly reusable, service designers should be able to add preconfigured form components and perform only small modifications on each service form. Remember, any change made to the elements used within a form component is inherited by all services that use that form component.

Removing Forms from a Service

You should be careful when removing a form component from a service. Remember, the form component's rules may refer to both dictionaries and fields in that same component and in other forms. It is the responsibility of the service designer to ensure that all dictionaries referred to by rules in a form components are actually present in the form (by virtue of being in an included form component.) In most cases, if a rule refers to a field that is not present in the current service, the rule fails silently, and any additional rules are unaffected. However, this may not be the case for hand-coded ISF, unless it has been coded very carefully. In that case, the ISF function's failure causes a JavaScript error, which stops execution of any further functions in the same event and may have other consequences.

To remove an active form component from a service, simply check the check box to the left of the component name and click **Remove Selected**.

Removing a form from a service does not delete the form from the system. All forms are managed in the Active Form Components area of Service Designer, and remain accessible to other services.

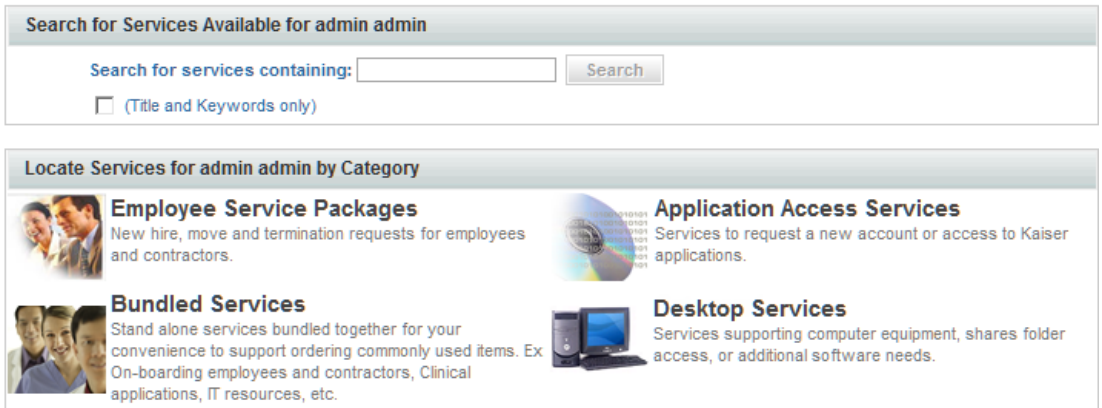
The Effect of Changing Form Components on Services

Any time you change a form component, those changes are automatically propagated to all the services that use that form component. Internally, Request Center maintains a version number for the service, and increments that number. This has no effect on users of the application modules, except that Service Designer users may notice a minimal delay if a form component is used in many (hundreds) of service definitions.

Similarly, changes to a dictionary are automatically propagated to all form components that use that dictionary. The changes to the form component are, in turn, propagated to all services that use the component, as above.

Managing Categories

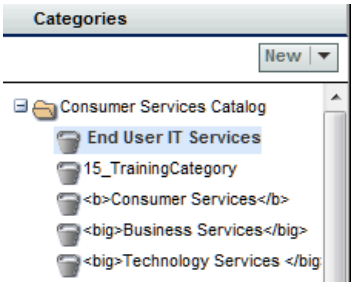
A category is a heading that exists on the catalog landing page or main menu to help customers find a service that meets their needs. In the Request Center sample below, from the My Services module, each of the cells in the section of the page labeled "Locate services for admin admin by Category" are, in fact, categories.



As the requestor clicks on a category, Request Center drills down to display subcategories or the services associated with the current category. The category hierarchy is completely user-configurable. Services can be associated with multiple categories. Categories are also use to organize service offerings in Demand Center.

Defining Categories

The Categories module is used to create and manage categories and category structure in your catalog. You specify a parent category and then the hierarchy within that parent. The Request Center catalog is named the "Consumer Services Catalog"; all categories are organized beneath that root category. A similar hierarchy can be constructed for the "Service Offering Catalog" for customers using the Demand Center module.



When you create a new category it is automatically added to the home page for the catalog (Consumer Services Catalog or Service Offering Catalog) in which it is created. Be sure to remove the category from the home page, if needed, and associate it with the appropriate categories and subcategories for your catalog.

Service Designer makes it easy for you to add new categories to the catalog.

The following table summarizes the fields on the New Category window.

Field	Definition
Name	The name of the category; up to 60 characters. If the "Use Categories in Search" setting is turned on in the Administration module, then any word that is in the category Name will return services that are in this category.
Description	A text field for entering a brief summary of the category; up to 255 characters. Use this if you are not specifying the URL.
or from URL:	Do not use this field. You cannot add a category description from a URL with the "from a URL" field. To add descriptive information from a URL for a category use the category's Presentation tab.

To add a new category:

Step 1 Within the Categories component, choose **New > New Category**.

The New Category window displays.

Step 2 Type a name for the new category in the Name field. There is a limit of 60 characters for a category name.

Step 3 Type a brief summary of the category in the Description field. There is a limit of 255 characters for a category description.

Step 4 Click **Add This Category**.

Step 5 Service Designer adds the category. You are now ready to configure it.

Configuring a Category

After you create a new category, you configure it by:

- Setting display options on the General Tab
- Determining the visual presentation on the Presentation Tab

Selections made on these category tabs are visible within My Services.

Use the General tab of the Categories component to determine how categories and subcategories display in the service catalog.

Field	Definition
Name	The name of the category.
Description	Descriptive text to describe what kind of products or services can be found within this category. This description displays with the category link in the catalog.
Display Style	Use the drop-down menu to choose whether to: • Display categories only. • Display categories and subcategories under them. If you display subcategories, the category appears in the portal with all of its subcategories as bullet points with active navigation links directly beneath it in the display.
Appears in categories	A list of the “parent” categories in which this category appears. To add this category to more categories, click **Include in More**. To remove this category from other categories, check the check boxes and click **Remove**.

Field	Definition
Subcategories in this category	A list of the subcategories or "child" categories which appear in this category. To display more subcategories in this category, click **Display More Categories**. To remove subcategories in this category, check the check boxes and click **Hide**.
Services in this category	A list of the services or service offerings which appear in this category. To display more services in this category, click **Display More Services**. To remove services from this category, check the check boxes and click **Hide**. You can also add a service to a category from the General tab for that service in Service Designer. You can only add service offerings to categories from within Categories.

Defining the Appearance of Categories

After you have set up the display options for a category, use the HTML tools on the Presentation tab to format how your category displays in My Services. See the "HTML Editor Tools Summary" section on page 1-84 for instructions on how to use the HTML editor.

To format the appearance of a category:

Step 1 Click the **Presentation** tab for a category within the Categories component.

Step 2 Click **Select an image to insert** to add a catalog icon.

- To add an image from the list of images currently available in the system, locate the image in the list and click **Add Selected Files**.
- To add a new image, click **Browse** to locate the image, then click **Attach** to upload the image to the system.

Images which represent categories can be in JPG,TIF, PNG, or GIF format. Category and service images are displayed with a width of 64 pixels and a height of 64 pixels, so custom images should be created to maintain this aspect ratio. Depending upon how you have customized Service Portal using the custom CSS capabilities, you might also want to consider developing custom icons with a transparent background.

Step 3 In the Category Details field, choose whether you want to show (and format) content in the top, middle or bottom of the category display area in the catalog. When you choose the "show" button, the bottom portion of the window populates with an HTML editor.

Step 4 If you choose to show multiple content panes, click the section of the schematic you want to format.

Step 5 If desired, enter a URL to display within the left, right or bottom content sections. The URL that you enter here must be fully qualified, beginning with "http://".

Step 6 Use the included HTML editor to enter and format text or graphics (this is chosen by default).

Or, to insert HTML code, click **Source** to disable the HTML editing tools and enter your coding directly. Click **Source** again to return to the HTML editor to display the rendered HTML code.

Step 7 Click **Save** to save changes.

Previewing the Category Display

Designing the category hierarchy is typically an iteratively process, so you will want to review your work as you proceed. The easiest way to do this is to have two browser sessions opened simultaneously—with Service Designer in one and the My Services (or My Services Executive) home page in another. Make changes to the categories and associated services and be sure to Update the page. Then, in the other browser session, refresh the page display; the new category hierarchy appears. The critical step here is to always save your changes in Service Designer before attempting to refresh the My Services or My Services Executive page.

Deleting a Category

You can delete a category if you no longer need it by choosing the category and clicking **Delete**.

Removing Categories, Subcategories, and Services

You may need to remove the categories in which a category displays, the subcategories that display in a category, or services from a category.

To remove, categories, subcategories, and services:

Step 1 Click the **Categories** component of Service Designer.

Step 2 Click the name of the category for which you want to remove categories, subcategories, or services.

The details for that category display on the right in the Category window.

Step 3 To remove a category in which this category displays:

- Check the check boxes next to the categories you want removed.
- Click **Remove**.

Step 4 To remove a subcategory from this category:

- Check the check boxes next to the subcategories you want removed.
- Click **Hide**.

Step 5 To remove a service from this category:

- Check the check boxes next to the services you want removed.
- Click **Hide**.

Step 6 Click **Update** to save your changes.

Managing Keywords

Keywords are words associated with a service that are used to support searching for a service within the service catalog. Keywords supplement words in the service name and description, and in categories (based on system settings), already used in keyword search. You can configure additional keywords, and associate them with the services you configure. All keywords in the system are available to all services in the system.

When adding a keyword, put yourself in the place of the customer and try to anticipate the words that make the most sense.

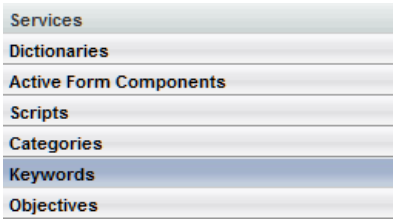
For example, if you have a "Computer Memory Upgrade" service, possible search keywords might be: RAM, memory, expansion, or upgrade.

You can create each of these keywords in Keyword Manager and then associate them to the applicable services or service offerings.

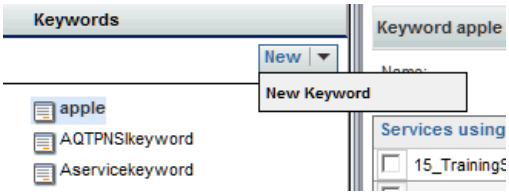
Adding a New Keyword

To add a new keyword:

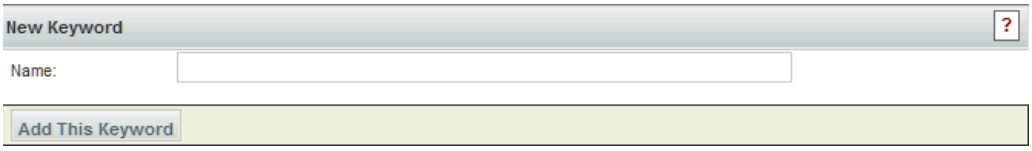
Step 1 Click the **Keywords** component of Service Designer.



Step 2 Choose **New > New Keyword**.



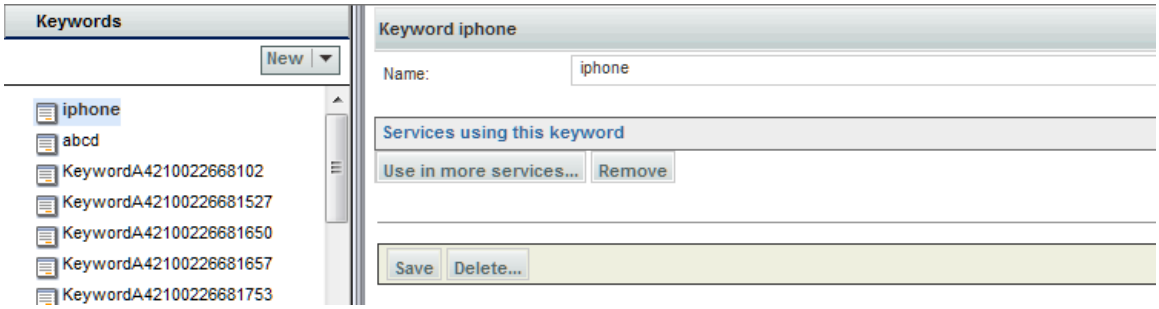
The New Keyword window appears.



Step 3 Type a name for the new keyword.

Step 4 Click **Add This Keyword**.

Service Designer adds the new keyword to the list of available keywords, and allows you to associate it with services or service offerings.



Associating Keywords with Services or Service Offerings

When you associate a keyword with a service, a user can employ it in My Services and My Services Executive to search for a service.

To associate a keyword association:

Step 1 Click the **Keywords** component of Service Designer.

Step 2 Click the name of the keyword you want to associate with a service or service offering.

The details for that keyword display to the right.

Step 3 Click **Use in more services**.

Step 4 Check the check boxes next to the services or service offerings that you want to associate with the keyword.

Step 5 Click **Add**.

Step 6 If you modify the keyword name, or edit the spelling, click **Save** to save changes.

Removing Keywords from Services or Service Offerings

You may find that you want to remove a keyword from its association with a service or a service offering. By removing an association, you are *not* deleting the keyword from Service Designer.

To remove a keyword association:

Step 1 Click the **Keywords** component of Service Designer.

Step 2 Click the name of the keyword you want to remove.

Step 3 Check the check box next to each service or service offering you want removed from association with the keyword.

Step 4 Click **Remove**.

Deleting Keywords

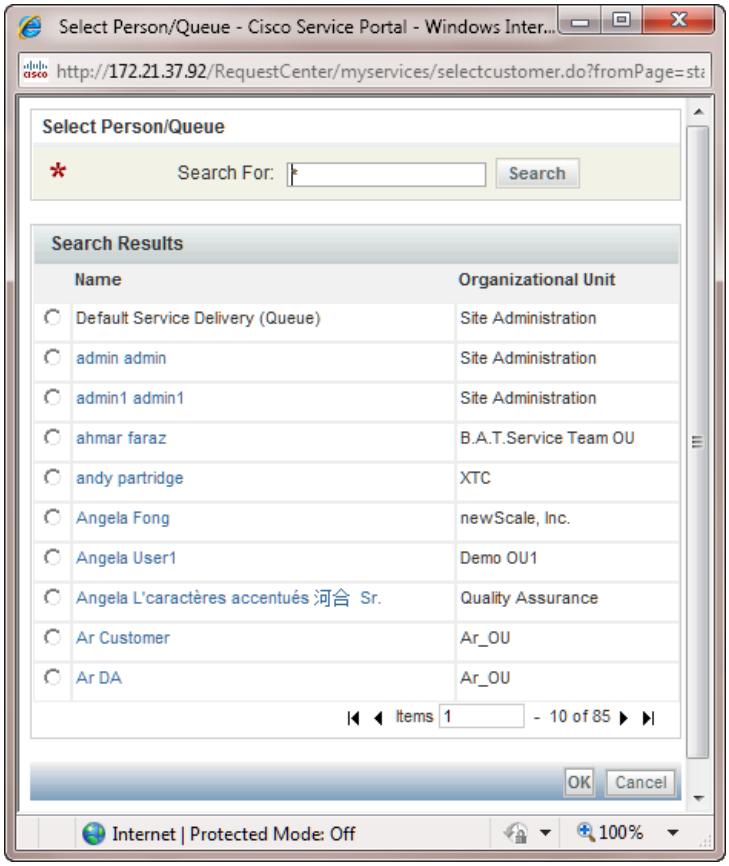
You can also completely remove a keyword by deleting it in the Keywords component of Service Designer. Choose **Delete** and Service Designer deletes the keyword and removes its associations to services or service offerings.

Select Person/Queue Dialog Box

The following table summarizes valid search criteria in the dialog box that appears if you assign a person/queue:

- to a role in the Authorizations tab, whether it is on the service-group or service level, after you choose "A person/queue" from the Assign drop-down menu and click the "**…**" button to the right of the "Assign to" field
- to dictionary rights in the Form tab or add people on the Permissions tab, after you click **Add People**

The Select Person/Queue dialog box appears, as shown below.



To search using any of the criteria, enter the appropriate data or wildcards in the fields at the top of the dialog box and then click **Search**.

Valid Search Criteria	Search Results
the wildcard (the * symbol) used alone	All people and queues in the system. You must enter the wildcard alone to see all possible selections (the full list of options does not necessarily appear by default).
first name or last name (except for “admin”)	All people with either matching first name or matching last name; use * alone to search for admin.
* before last name or * after first name (with no spaces)	All names with something either in front of the name or after the name that matches the search entry.
*queue (no space between * and the word “queue”)	All queues except for Default Service Delivery Queue; use * alone to search for the Default Service Delivery Queue.
* before or after a word in queue or organizational unit name	All queues that have something before or after the word in the queue that meets the search entry and all people names that belong to an organizational unit (OU) containing a matching text string. For example, typing Financial* yields both “Financial Queue” and the people in an OU called “Financials Team”.

Valid Search Criteria	Search Results
organizational unit name	All queues or people that belong to the organizational unit meeting the search entry.
name (replace “name” with a real name at your site)	Both all people and queues that have something in front of the name and after the name that matches the search entry and all people and queues that belong to an organizational unit that matches the search criteria. For example, typing Fin* yields results like “Financial Authorizations Queue” and “Alan Ashcraft” who belongs to the Financials Service Team. The *name* method does not function when searching for “admin”.

HTML Editor Tools Summary

In Service Designer, an HTML editor is used:

- On the Presentation tabs for a service in the Services component.
- For categories within Categories component.

In the Administration module, the HTML editor is used to format the body of notification emails.

This figure and table below show the tools available in the HTML editor.

By holding your mouse over a button in the editor, you can learn the name and function of the icon. This table describes the buttons as they are viewed from top to bottom, left to right, in the above figure.

Icon/Button	Use to:
Source	Clicking **Source** allows you to enter HTML source code directly in place of the standard editor (default). Click **Source** again to return to the standard editor to display the rendered HTML code.
Cut	Cuts the selected text.
Copy	Copies the selected text to the clipboard.
Paste	Pastes text from the clipboard at the insertion point.
Undo	Undoes recent changes.
Redo	Redoes recent changes.
Select All	Selects all text.
Remove Format	Removes formatting.
Find	Finds a word or text string.
Replace	Replaces a word or text string with a word or text string you enter.

Icon/Button	Use to:
Insert/Remove Numbered List	Numbers the selected paragraphs or the paragraph that contains the cursor.
Insert/Remove Bulleted List	Creates a bulleted list from the selected paragraphs or the paragraph that contains the cursor.
Decrease indent	Decreases the indentation of the selected paragraphs of the paragraph that contains the cursor.
Increase indent	Increases the indentation of the selected paragraphs of the paragraph that contains the cursor.
Left Justify	Aligns the selected paragraphs or the paragraph that contains the cursor to the left.
Center Justify	Centers the selected paragraphs or the paragraph that contains the cursor.
Right Justify	Aligns the selected paragraphs or the paragraph that contains the cursor to the right.
Insert/Pick an Image	Allows you to insert an image previously uploaded or to upload an image from the local workstation and then insert it in the current document. Images must be of the type JPG, GIF, TIF, or PNG format.
Table	Inserts a table and sets its properties.
Insert Horizontal Line	Inserts an horizontal line.
Insert Page Break for Printing	Inserts a page break.
Link	Creates a hyperlink to a URL.
Unlink	Removes a hyperlink to a URL.
Anchor	Prompts you to enter a name for the anchor inserted at the insertion point.
Bold	Bolds selected text.
Italic	Italicizes selected text.
Underline	Underlines selected text.
Paragraph Format	Allows you to apply predefined style formatting.
Font	Selects the font.
Size	Selects the font size.
Text Color	Selects the font color.
Background Color	Selects the background color behind text.
Maximize	Maximize the screen display. Clicking this button once the display has been maximized will restore the page.

CHAPTER 2

Active Form Components

- Overview, page 2-1
- Service Form Framework, page 2-4
- Service Form Performance and Security Considerations, page 2-77
- ISF Application Programming Interface (API), page 2-82
- ISF Coding and Best Practices, page 2-104
- Best Practices for Using Active Form Components, page 2-111
- Putting It Together, page 2-116
- Server-Side Associated Controls, page 2-125

Overview

Overview of a Service Form

A **service form** is the web page presented to Request Center users which displays the information required to fulfill a service request. The service form allows catalog customers to enter detailed information on the service they are requesting, and allows request authorizers, reviewers, and task performers to review, and potentially modify, information previously entered as well as supply additional details required for request fulfillment.

The service form consists of many components that are configured by the service designer. These components define the appearance of the service as well as its behavior in response to a particular event. The event may be user-initiated (for example, the user selects “laptop” as the type of computer to order) or part of the requisition life cycle (the request has been submitted, and a manager must now approve it). The components which are discussed in this chapter include:

- **Dictionary** – A group of fields which hold the data required to request a service and fulfill that service request.
- **Conditional rule** – A rule which governs how a form should look and behave in response to events that occur during request initiation and fulfillment.
- **Data retrieval rule** – A rule that allows data to be retrieved from external datasources, for the purpose of prefilling information on a service form, retrieving additional information based on the user's data entry, or validating the user’s data entry.

- **Form** – A set of one or more dictionaries, any rules that apply to those dictionaries or fields in those dictionaries, and details on how to display the dictionary and fields on the web page during the requisition life cycle. These **active form components** are the building blocks from which service forms are constructed.
- **ISF** (Interactive Service Forms) – a JavaScript API (application programming interface) through which programmers can provide additional interactivity to the service form by writing JavaScript functions and libraries.

Common Uses of Active Form Components

Form rules allow service designers to make service forms a Rich Internet Application (RIA) without having to write code! (RIA is a buzzword which means that the application responds immediately to what the user types or what appears on the screen, without the user having to click a “Submit” button at the end of a screen of input.) The rules allow designers to declaratively—by filling in an interactive dialog box and following the steps in a set of wizards—specify how the service form's appearance and behavior should change in response to user-initiated events.

Some common uses of conditional rules include:

- Enable/disable or show/hide fields based on a radio button (for example, Yes/No selections)
- Mark fields as mandatory based on the value of another field, or during a particular task or moment in the delivery (fulfillment) cycle
- Show/hide entire dictionaries to customize the user experience for different tasks within the service delivery moment
- Set focus on a field to attract the user’s attention
- Validate data for correctness

Data retrieval rules provide an online, real-time interface between the service form and information stored in a relational database. These rules allow such data to be displayed in dictionary fields, or interrogated to determine the correctness of items entered by service requestors or fulfillers. Some common uses of data retrieval rules include:

- Prefill form data with information maintained via other applications such as a Configuration Management Database (CMDB) or ERP or HR system
- Provide dynamic drill-downs so that the items listed in a drop-down list vary dynamically, based on an item previously entered or chosen from another list

Lightweight Namespaces

Designers use namespaces to send email to the people when a particular task starts or is completed, or to dynamically determine whether a particular task or authorization should be executed. Namespaces are documented in detail in Chapter 6, “Namespace”.

Active form rules need the equivalent of namespaces in order to dynamically access field values to be used or evaluated. For example, the service may need to display an additional dictionary or field if the user entered “Other” in a previous field; the current customer's organization may need to be used as the criteria for building a drop-down list to display valid locations for a service delivery; default values need to be provided for customer and initiator data.

Lightweight namespaces provide these capabilities. They are “lightweight” since only the information accessible to the service form (not, for example, details about the service's delivery plan or task performers) can be used within the rules.

Who is the Audience for this Chapter

This chapter is intended primarily for service designers who are responsible for designing the appearance and behavior of a service form for request fulfillment. All specifications governing the form's appearance and approval/review cycle, as well as most rules determining the form's behavior are defined “declaratively”—that is, there is no need to write commands in a programming language; the designer simply fills in a series of dialog boxes and follow steps in a wizard to specify the desired result.

Some aspects of defining active form components may require some programming expertise. In particular:

- Designers may write SQL to specify complex queries to be performed to validate or retrieve data in a data retrieval rule. This supplements the capability to have Request Center generate simpler queries automatically.
- JavaScript coding may be required to supplement the declarative capabilities provided by conditional rules. For example, complex computations, interdependencies or other procedural code, or manipulation of data in grid dictionaries may be inserted via ISF. In addition, users may include additional controls (buttons) on the form which execute JavaScript functions.

What is Covered in this Chapter

This chapter includes guidelines and instructions for using active form components to enrich the out-of-the-box functionality of service forms. After reading this chapter, you should know how to customize service forms to your requirements, as well as some recommended “Best Practices” for performing this customization.

- The “Service Form Framework”section gives an overview of service design using Active Form Components as well as guidelines for using the components to enhance the service form usability and reduce maintenance time and cost.
- The “ISF Application Programming Interface (API)” section gives detailed instructions on how to use the screens and options available in Service Designer to work with ISF. ISF capabilities, including all available functions and how to integrate these functions into a service form.
- The “ISF Coding and Best Practices”section reviews a typical development cycle for ISF and considers some both practices for coding ISF.
- The “Best Practices for Using Active Form Components”section does just that, showing sample rules and ISF code and discussing when and how to use each in some frequently encountered use cases.
- The “Putting It Together” section discusses design principles for implementing forms and services with the lowest cost to maintain, yet provide the desired functionality.
- The “Server-Side Associated Controls” section considers options for implementing server-side code to interact with form-based data.

What is not Covered in this Chapter

- This chapter does not cover a detailed operation of Service Designer. It assumes the reader is familiar with Service Designer and can navigate through the different screens, tabs and dialog boxes. This information is available via online help.
- This is not a JavaScript guide. The primer mentions some JavaScript techniques, but it assumes the reader is already familiar with JavaScript and dynamic HTML (DHTML) to control web pages.

Service Form Framework

Overview

All active form components execute within the context of a service form. A service form is an HTML page consisting mainly of elements generated dynamically, based on form, dictionary and field specifications defined via Service Designer. The service form also supports the ISF API, which includes global variables, a JavaScript object model, and a set of JavaScript functions which programmers can invoke to achieve the customization desired for their service forms.

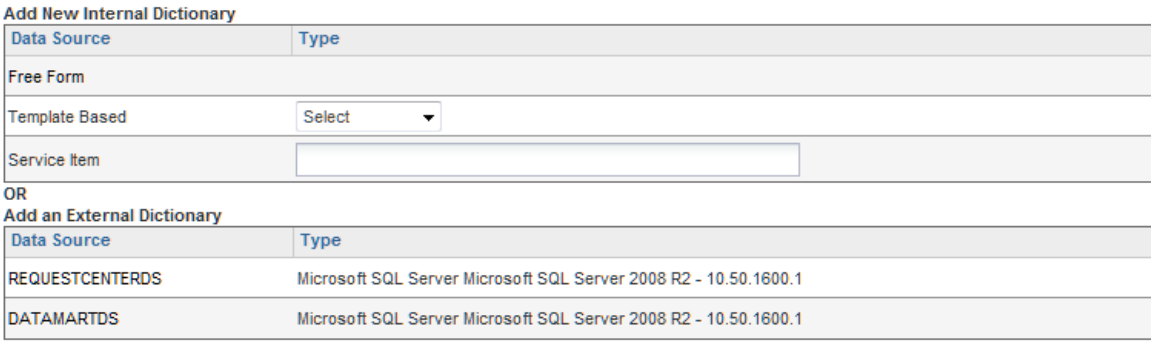
A service form is the interactive web page through which service requisitions are entered and tracked in Service Portal. Service designers configure service forms by specifying their components and behavior using Service Designer. The basic appearance and behavior of a service form is determined by the dictionaries and fields specified as part of the “active forms” that are used in the service definition, and by the permissions granted to identified users or groups of users to view or edit dictionaries in the sequential “moments” of a requisition's life cycle.

Understanding active form components and other declarative design capabilities provided by Service Designer is critical to understanding the architecture and capabilities of a service form. Therefore, it is useful to examine the design components that comprise a service.

Dictionaries

A dictionary is the building block for a service form. A dictionary is a grouping of fields, the input elements through which users enter data for their service request and by which data previously entered or supplied automatically may be displayed.

A dictionary is defined and maintained via the Dictionaries component of Service Designer. Choose a previously defined dictionary on the left, or choose **New > New Dictionary** to create a new dictionary. You start by specifying the type of dictionary.



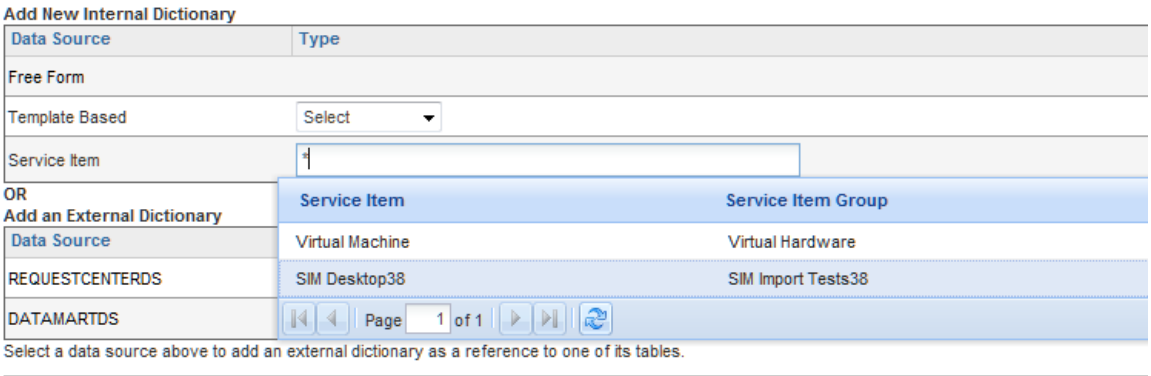
The two main categories of dictionaries, Internal and External, refer to the way data in the dictionaries is stored. Internal dictionaries represent data structures that are managed by, and within, Request Center. External dictionaries, on the other hand, use existing or new data tables outside the Request Center requisition. Internal dictionaries are generally preferable, since they minimize database administration required and include additional functionality not available in external dictionaries.

Internal dictionaries are further categorized by how they are defined.

Dictionary Type	Description/Usage
Free form	Designers may include any fields and designate their names, data types, and sequence within the dictionary.
Template Based	A design pattern predetermines fields available for inclusion in the dictionary. A person-based dictionary, based on data available in the person profile, is currently the only available template-based dictionary.
Service Item Type	A service item, defined in Service Item Manager module, provides the template for fields that can be included in the dictionary.

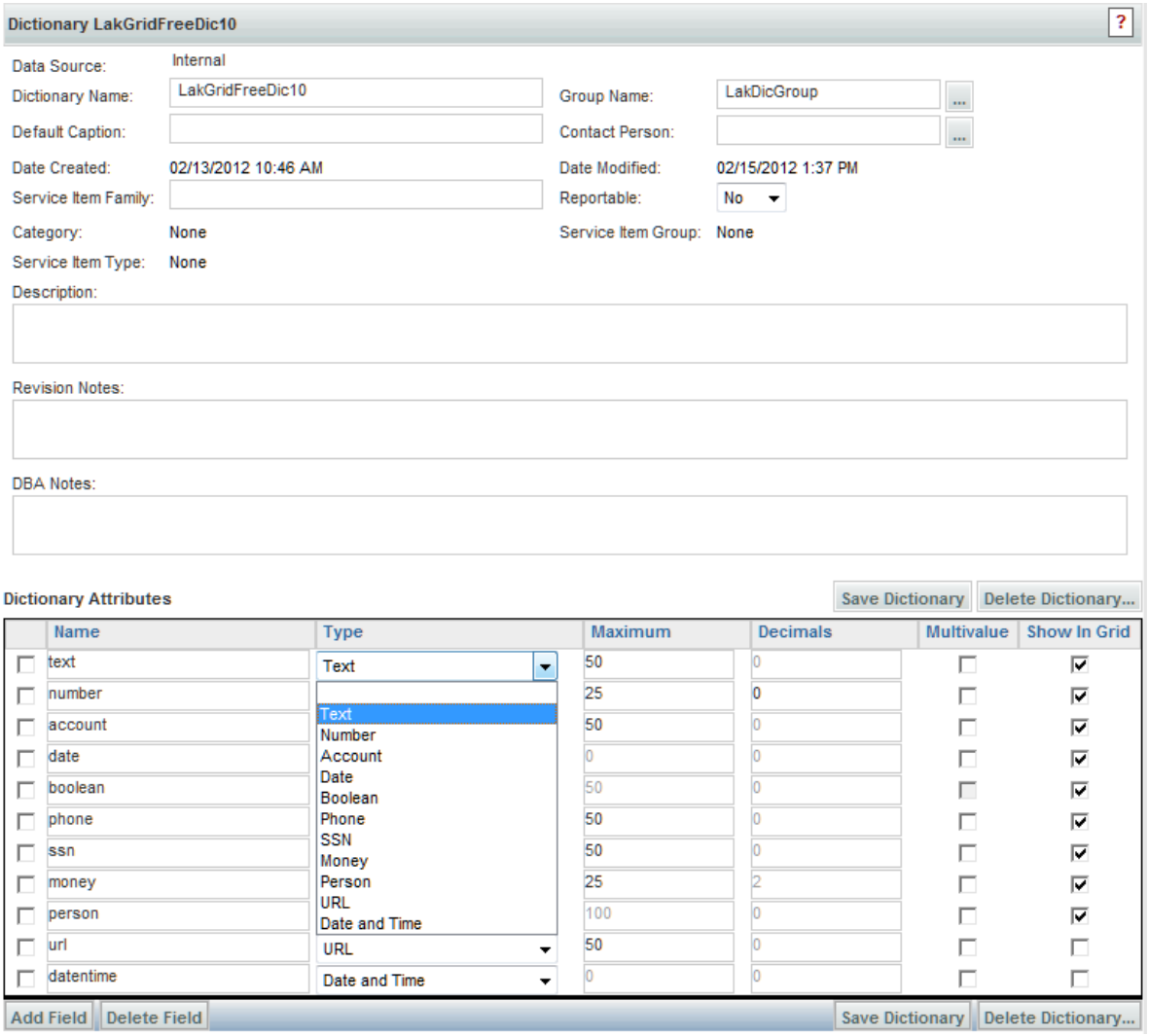
To create an internal dictionary:

- To create a free-form dictionary, click on the **Free form** link. You can then specify the dictionary name, caption, and other attributes, and start adding the desired fields. (See the "Free-Form Dictionaries" section on page 2-5.)
- To create a person-based dictionary, click the down-arrow on the Select list for the Type of Template-based dictionary and choose **Person Based**. (See the "Person-Based Dictionaries" section on page 2-7.)
- To create a Service Item-based dictionary, in the Service Item field, search for the service item by name, then choose the item from the popup, as shown below. (See the "Service Item Dictionaries" section on page 2-11.)



Free-Form Dictionaries

A free-form dictionary means just that: service designers are free to specify the fields in the dictionary, the order in which they occur, and the data types assigned to each, as shown below.



The field name (“Name” column in the illustration above) can consist of alphanumeric characters and the underscore and must start with a letter. It cannot contain spaces or other special characters. Reserved words in JavaScript (such as “this”) cannot be field names.

The data type (“Type” column in the illustration above) influences the HTML representation that can be chosen when the dictionary is included in a form. That, in turn, influences how rules and ISF functions may be applied to the field, as well as the usage of the field within the Data Mart, should the dictionary, or a service which includes that dictionary, be made reportable.

Check the check box in the “Show in Grid “column for the fields you wish to show if the dictionary is configured as a grid on the form. (See “Using Grids on Forms” section on page 2-16 for more information on grids.)

Type	Description and Active Form Implications
Text	Default data type, supports alphanumeric data; should be used for fields to be rendered as single- and multi-line text.
Number	Data type for all numbers, including integers and whole numbers; it is critical to specify the precision in the Decimals column, as the application will validate decimal precision as well as length.
Account	Documentation only; data treated as alphanumeric.
Date	Data compatible with the database's native datetime type, but display restricted to the date; calendar widget supplied for data entry.
Boolean	Data object whose possible values are “true” and “false”, presented as “Yes” and “No”.
Phone	Documentation only; data treated as alphanumeric.
SSN	Documentation only; data treated as alphanumeric.
Money	Data validated to contain only a valid number; monetary symbols or commas cannot be typed; accepts numerical characters and up to 3 decimal places.
Person	Data validated against a person ID in the personnel profiles; a Person Search dialog box is available via a “Search” button automatically rendered on the service form. The Person data type is provided primarily for backward compatibility; if this capability is required, a person-based dictionary should be created.
URL	Data stored as alphanumeric; a saved value is represented both as text and as an HTML representation of the value, providing a link to the specified URL.
Date and Time	Data stored as a date and time; calendar widget supplied for data entry contains a time-selection widget.

When a dictionary is reportable, the ability to modify the dictionary definition is temporarily disabled. The data types of fields in reportable dictionaries cannot be switched between numeric/money, date/datetime, and the character types. If you would like to make any other change to the dictionary, you can switch the Reportable setting to “No”, save the dictionary, make the changes, and then restore the setting to its previous value. Before designating a dictionary as reportable, be sure to read the guidelines on defining reportable objects in the *Cisco Service Portal Reporting Guide*.

Person-Based Dictionaries

The application maintains a repository of all people in the user organization who may need to access Service Portal applications. The personnel information in this repository is generally populated via directory integration—the application is instructed to retrieve data from an enterprise-wide LDAP directory—but may be manually maintained via Organization Designer.

Service forms typically need to refer to such personnel information. For example, a service request always has a customer—the recipient of the service; and an initiator or requestor—the person who sits at the keyboard and requests the service. In most cases, the customer and the initiator are the same person—an employee requests a service for him/herself. Alternatively, an administrator or other authorized employee may initiate a service request on behalf of another person, who becomes the customer for the service.

Service requests may also contain other information relating to corporate personnel. A frequent use case is when a requester must designate one or more approvers for a request. In this scenario, the user selects the approver by searching through a drop-down list of available personnel, and the approver's contact information is included in the service form data for easy reference.

Person-based dictionaries provide the mechanism for retrieving data on a particular person from the repository and designating which aspects of the personnel data should be displayed on the service form.

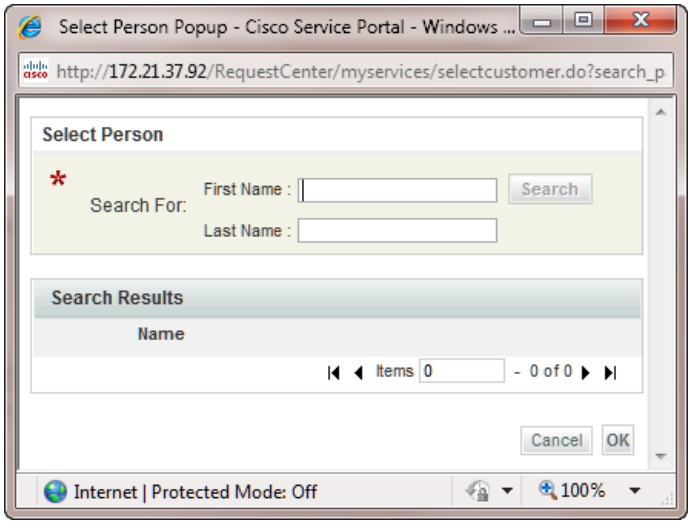
This search capability is provided by person-based dictionaries. The dictionary name and group are freely editable. Such person-based dictionaries automatically include the “Select_Person” attribute.

The “Select_Person” attribute provides the capability to search for people, either within the repository or, if directory integration has been enabled for service forms, in an external directory.

The Select_Person attribute has a data type of “Person”, not text. This data type governs the appearance of the attribute when it is used in a form. For example, the “Name” field on the service form below is defined as the “Select_Person” attribute.

In any new person-based dictionary, you will see “Show in Grid” checked by default for the Select_Person since it also checked for use (“Use”) by default. “Show in Grid” cannot be unchecked for the Select-Person, just as you cannot uncheck “Use”.

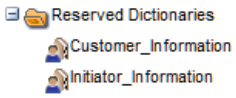
When a person-based dictionary is included on a service form, clicking the “Select” button brings up a “Person Search” dialog box which allows the user to specify search criteria and choose a single person, as shown below.



Once a person is chosen, the search dialog box is dismissed, and any fields used in the dictionary are automatically filled in with the current values of the corresponding fields in the chosen person's profile. The name appears in FirstName LastName format.

Reserved Dictionaries

A Service Portal instance automatically includes two person-based dictionaries, for the customer and initiator information. These dictionaries are in the "Reserved" service group.



The Customer_ and Initiator_Information dictionaries supplement the standard behavior of automatically recording the customer and initiator for all requests. All information on the customer and initiator is available throughout the requisition life cycle via Business Engine namespaces, and is displayed as part of the Requisition Summary page for the request in My Services and Service Manager.

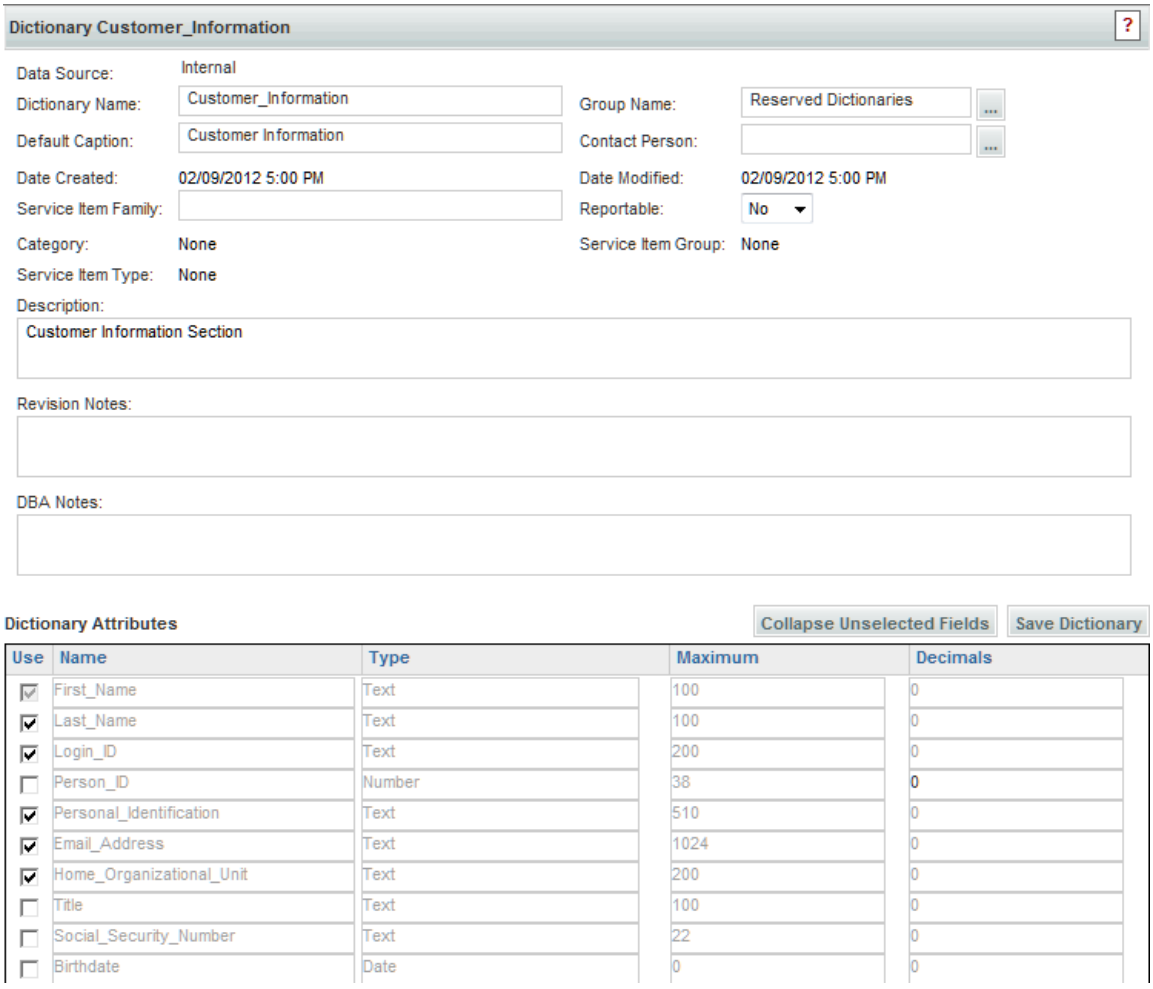
Requisition			
Requisition Number:	14	Status:	Ongoing
Customer:	admin admin	Initiator:	admin admin
Customer E-Mail:	rc@newscale.com	Created Date:	12/01/2011
Customer Work Phone:		Submit Date:	12/01/2011
Bill To:	Site Administration	Closed Date:	

However, the fields that are displayed are not configurable. Further, these fields reflect the values that were retrieved from the person's profile. The person ordering the service has no opportunity to correct information that may be out of date or to supply additional information needed to fulfill the current request. In addition, for reporting and governance reasons, it may be required to track customer and initiator information as they were when the request was submitted, not reflecting any subsequent changes.

For these reasons, service designers typically create dictionaries containing customer and initiator information and include these dictionaries in an active form component which is, in turn, included in all services. The dictionary and group names of the Initiator and Customer Information dictionaries cannot be changed. Other general properties of the dictionaries are editable. The reserved dictionaries are included on a reserved form, the Customer-Initiator form. Form contents and appearance can be defined and its behavior manipulated by any form rules or ISF available.

The reserved dictionaries list all available personnel information, and allow the designer to designate which attributes should be part of each dictionary. Designers should choose the attributes to be included in the dictionary by checking the corresponding attribute Name. For example, you should typically include Supervisor information, since some service requests may require the customer's supervisor's approval. Attribute names and data types cannot be changed. You should also be sure to include any attributes that need to be manipulated in a service, even if the field will always be hidden. Choosing the attribute ensures that it is populated with the corresponding value from the person's profile. Designers can then configure the form containing the dictionary to hide the field as appropriate.

The personnel profile includes 10 custom fields (named Custom1 through Custom10) that are not used by Request Center. Any of these fields can be included in the Customer or Initiator dictionary. Some of these fields may be mapped to person attributes imported via directory (LDAP) integration. Others may be reserved for assignment or manipulation via the Display Properties or Conditional Rules available when configuring the Customer-Initiator Active Form Component which includes this dictionary.



For a Person Based dictionary, and the Customer and Initiator dictionaries, you use the Dictionaries component of Service Designer to choose which dictionary fields (attributes) will appear on a service form. Simply navigate to the dictionary, and check or un-check the check boxes in the “Use” column to determine which fields are used, or included, on a form that uses the Person Based dictionary, or on the Customer-Initiator form—the active form component which automatically includes both Customer and Initiator dictionaries.

In addition to a customer or initiator's first and last name, and login ID, it is common to add fields for the person's email address, and home OU. The Person_ID is the unique identifier assigned to the person. Basic form processing does not require use of this field; however, it may be useful in writing data retrieval rules, for example, to dynamically retrieve additional information about the person or his/her supervisor that does not have a corresponding attribute in the dictionary. Similarly, the Supervisor_ID is also selectable for inclusion in the reserved dictionaries. This would allow the service designer to dynamically construct a “chain of command” for approval of chosen requests.

Similarly, you may want to include location information, especially in the Customer dictionary. This information may be required, for example, to determine which queue a task should be routed to, or simply to indicate the person’s address, in case contact via an actual physical visit is required.

Service Item Dictionaries

A “service item” is a type of “configuration item”—a piece of hardware, software, or equipment which can be delivered in response to a service request and whose life cycle can be managed via subsequent service requests. It may be physical, such as a physical hardware device (for example, a cell-phone or server); but it may also be virtual, such as software (for example, an application or a login ID). The Service Item Manager module in Service Portal is used to define the service item or to review the definition of preconfigured service items such as a virtual machine. These service items can in turn be used to create service item-based dictionaries (SIBDs) and used for capturing or displaying service item instance information as part of a service request. For more information on service items and Service Item Manager see the “Service Items and Service Item Manager” section on page 3-4. For detailed information on incorporating service items in service design, see Chapter 3, “Lifecycle Center”.

Service Item Dictionaries contain the fields which provide the data stored on a service item. For more information see the “Defining Service Item-Based Dictionaries” section on page 3-22.

Integration Dictionaries

The Integration dictionary group is automatically created in all Service Portal instances. Any dictionary that is created through the Integration Wizard (Service Designer’s wizard for creating web services integrations between Service Portal and external systems) is automatically placed in this group. Once created, integration dictionaries can be moved to any dictionary group. Designers cannot manually place dictionaries in this group.

See Chapter 1, “Service Designer” for more information on the Integration Wizard.

Active Form Components – Forms

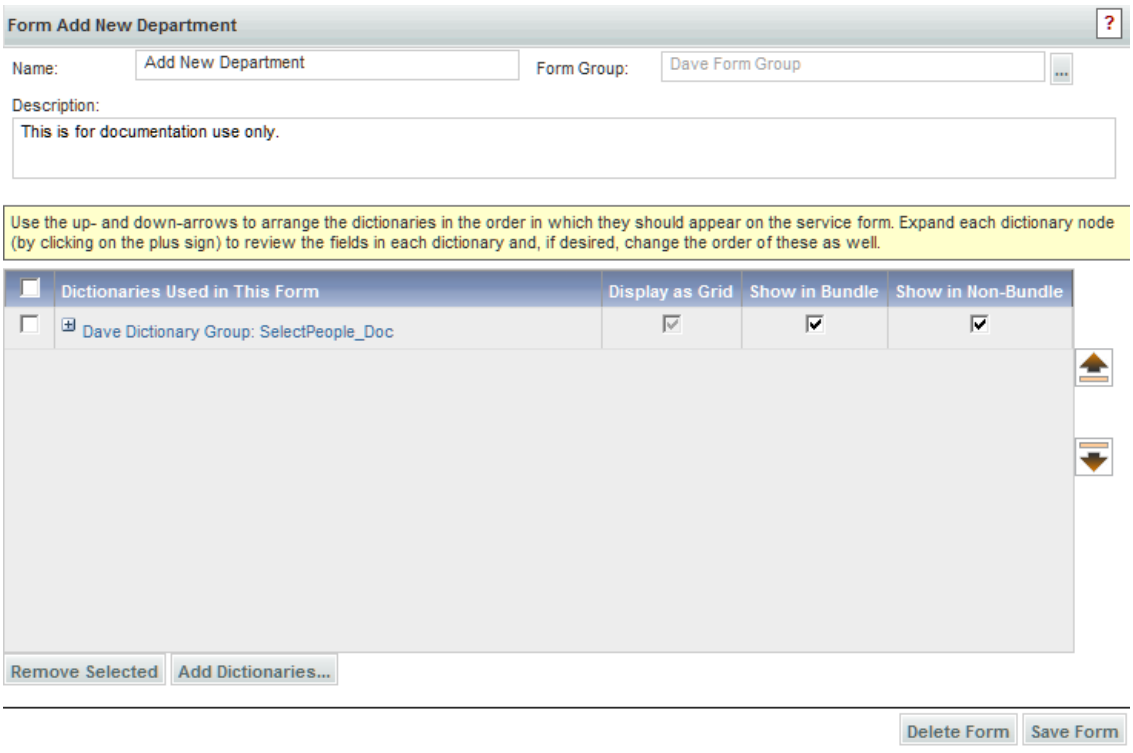
A form is a building block for implementing a service. Each orderable service consists of one or more forms. Each form specifies the appearance and behavior of the web page presented to users when they order a service from the service catalog; authorize or review requests for services; and complete the steps required for delivery of the service to its recipient. That web page is called a “service form”.

The service designer specifies the following form components:

- **Form Content**: Dictionaries included in the form and the order in which the dictionaries and fields that comprise the dictionaries are displayed
- **Display Properties**: How the individual attributes which comprise each dictionary are rendered on the web page when a user is working with a service form
- **Access Control**: Which users, or group of users, are able to view or edit specific dictionaries which comprise the service form at a each moment in the requisition life cycle
- **Active Form Rules**: Rules which can conditionally alter the appearance or behavior of the dictionaries or individual attributes displayed on the service form, or can dynamically retrieve data from relational datasources
- **Active Form Behavior**: The events which trigger execution of a conditional rule or a script written using JavaScript in conjunction with ISF

Form Content

The first step in configuring a form is typically to specify the dictionaries that are used in that form and the order and orientation in which those dictionaries, and the fields in each dictionary, appear.



The Form Content tab lists the dictionaries included in the form. Dictionary names are prefixed with the dictionary group name. Click the plus sign (+) to the left of the dictionary name to display the fields in that dictionary. To change the order in which a dictionary or field is displayed, at the right of the page choose the item to move and click the Up- and Down-Arrow keys until the item is in the desired sequence.

Note The dictionary names on the Form Content tab are a link to the Dictionaries component of Service Designer. If you need to add another field to a dictionary, you can simply **Ctrl-Click** on a dictionary name from here and go directly to the dictionary.

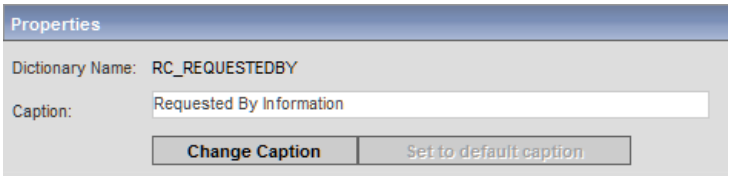
Dictionaries do not need to be included in a form in order for rules defined in that form to refer to a dictionary or field. Further, a form may contain no dictionaries at all! In this case, the form is a repository for rules. The form must be included in a service that also includes other forms which, in turn, include the dictionaries to which the rules apply.

The same dictionary can be used in multiple forms, provided that only one of those forms is included in a service. However, all properties regarding the dictionary's appearance and behavior would need to be specified in each form, so this is not an ideal architecture.

Display Properties

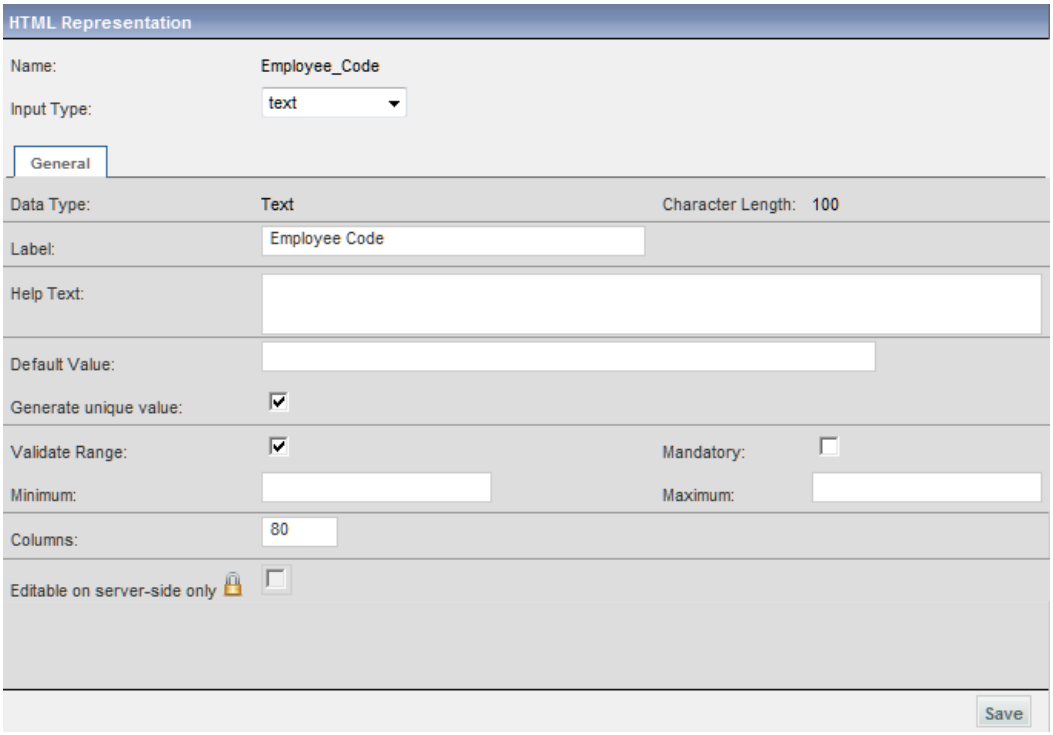
The service designer uses the Display Properties of the active form component to configure the appearance of each dictionary and of each field in the dictionary on the service form.

Each dictionary is assigned a caption, which is rendered as a tab heading for the dictionary when it is displayed on a service form. The default caption is the dictionary name; this can be freely changed or restored. If the dictionary is not assigned a caption, the tab heading will not appear on the service form, and the dictionary will appear immediately after the preceding dictionary.



When the dictionary was defined, each field was assigned a data type, defining the field's storage requirements. Within the form, each field must be assigned an “HTML representation”—how the field is to be rendered for any service that includes this form.

The HTML representation includes the “Input Type” for the field—the HTML element which will represent the field on the service form. The Input Type directly influences how form rules and ISF functions can be applied to the field and its contents. HTML representation also determines the detailed options available for configuring each field. The screen below shows the options available for a field assigned an HTML representation of “text”, which is rendered as an HTML text box.



HTML field representations are summarized in the table below.

Input Type	Description and Active Form Implications
text	Rendered as an HMTL “text box”.
textarea	Rendered as a multi-line text area.
password	Rendered as a password field, where values are displayed or echoed as asterisks.
hidden	Rendered as an HTML text box, but with a display type of “hidden”.
radio	Rendered as an HTML option list; the field has multiple options—the field value is the option currently chosen. Not available for fields set to “Show in Grid”.
select (single)	Rendered as an HTML drop-down box from which the user can choose a single value.
checkbox	Rendered as a set of HTML check boxes for all the possible options as designed in Service Designer. Not available for fields set to “Show in Grid”.
select (multiple)	Rendered as an HTML drop-down box from which the user can choose multiple values. Not available for fields set to “Show in Grid”.
SSN	Rendered as a single-line text box; “SSN” provides documentation only.
Person	The field is rendered as two objects: a drop-down list displays people and allows users to choose one; a second (hidden) object contains the unique identifier of the person chosen. This HTML representation is automatically applied to the Select_Person field within a person-based dictionary. It could also be applied to a field in a free-format dictionary with a “Person” data type; however, this latter usage is provided primarily for backward compatibility and is not recommended.

Input Type	Description and Active Form Implications
URL	Rendered as a single-line text box with a link generated for accessing the URL once the value has been saved.
read-only	Rendered as text—a user will not be able to enter data. The service designer is responsible for providing a value (typically via a form rule or default value display property) to such fields.

Generate a Unique Value for a Dictionary Field

You can generate a unique ID as the value for any dictionary field with an Input Type of "Text", "hidden", or "read-only" by checking the check box "Generate unique value" on the Display Properties tab, as shown in the Employee_Code field example above. This helps satisfy a requirement you may have for creating a unique value for a service item name or attribute, for example.

When a new service is requested, a universally unique identifier (UUID) is generated and set as the value of the field upon form load for all fields checked "Generate unique value". Conditional rules and data retrieval rules are executed after this step; so if there is a conditional rule or data retrieval rule that sets the value of this field, then the UUID value may be overwritten by the rule execution.

The UUID is generated as per the ISO standard—a 36 character string in the format: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX.

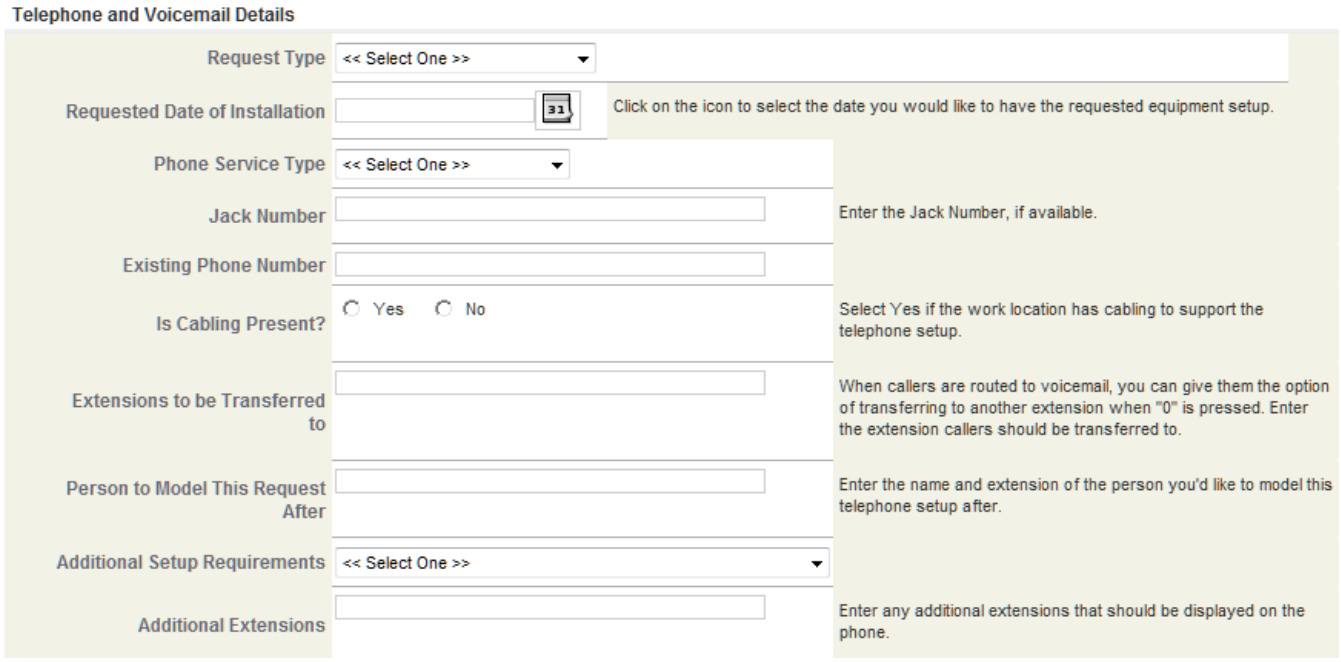
Note The UUID is not generated when a service is ordered through a Requisition API (RAPI).

Editable on Server -Side Only

Dictionary fields that contain sensitive information, or contain values set by default or with auto-retrieval mechanisms (for example, price of a virtual machine service item, or login ID of a person), should have the check box "Editable on sever-side only" checked to protect them from being altered by malicious hacking attempts. The field is shown with a lock icon to indicate a security feature. See the "Service Form Performance and Security Considerations" section on page 2-77 for more information.

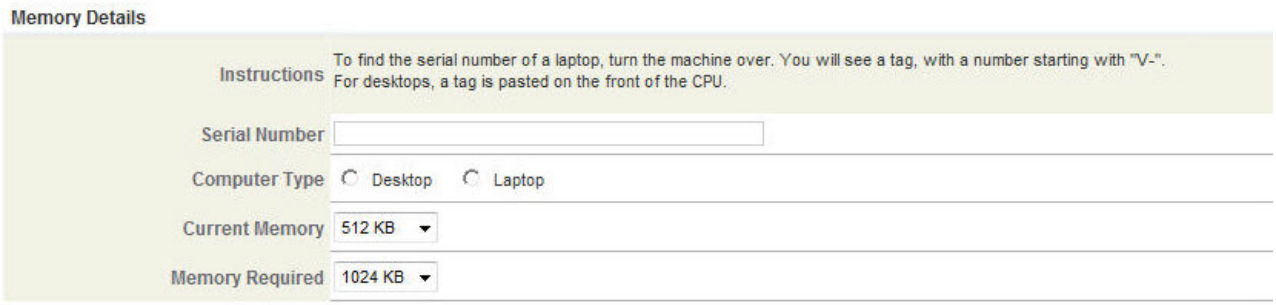
Service Form Appearance

The application dynamically determines the layout of the service form, based on the widths of the fields in all dictionaries displayed during the current moment. In nongrid dictionaries, each field is laid out horizontally, with the label followed by the field's input area and then the help text.



You can expect the form layout to change if you change the number of columns in a field or the field label.

Fields that have an HTML representation of read-only are not expected to have help (instructional) text. However, the default value of the field itself may be used for providing such help text:

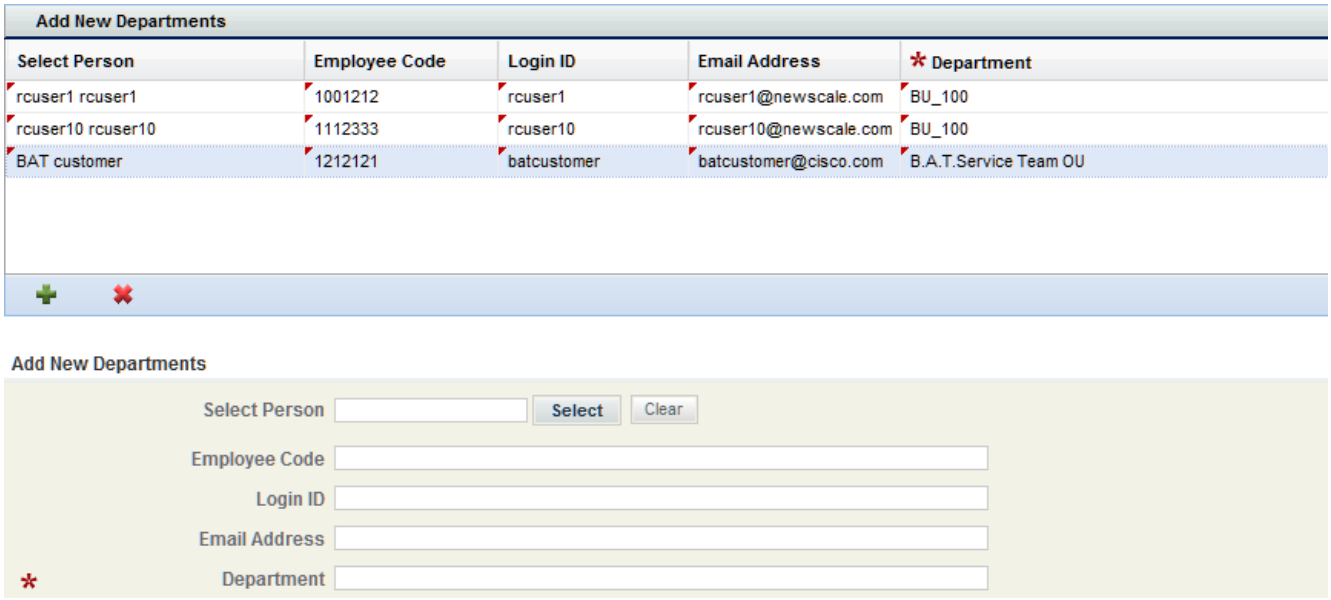


Field labels are automatically generated as bold. Additional or alternative HTML formatting can be applied to a specific field by using the **Advanced Formatting** button.

Using Grids on Forms

Service Portal supports the use of grids on a service form. You can configure any dictionary (with the exception of External) to be displayed as a grid on a form. A grid is created using all of the dictionary fields that have been checked as “Show in Grid” (in the dictionary definition) as columns on the grid. A grid enables the user to enter multiple data instances of fields (rows) on one form. To add a new row, the user clicks the Add button ( ), as shown below. When the Add button is clicked, an empty blank row is inserted at the end of the last row and highlighted.

The example service form below shows the difference between a grid and nongrid dictionary layout. Two dictionaries with the same fields have been used, with the first one set to display as a grid.



When to Use a Grid

A grid is useful when you are designing forms that require multiple data instances of the same fields to be entered on one form. Rather than creating multiple sections of the same fields, you can create one grid with the fields. A grid enables you to create and configure only one dictionary rather than multiple to hold the multiple sets of the same fields.

For the most part, configuring a grid is like configuring a nongrid dictionary, except that the dictionary is effectively turned ninety degrees so that field labels appear atop the fields. The primary difference is that grid cells are not truly individual fields when rendered in the browser, although they are stored as individual fields in the database. This difference results in some restrictions you should be aware of. Those restrictions, and other differences between grid and nongrid dictionaries, are as follows:

- The Input Types of Radio, Checkbox, and Select (Multiple) cannot be used.
- Field labels appear as column headers. Advanced formatting cannot be configured for the field labels in a grid.
- Buttons cannot be used.
- Some ISF Dictionary- and Field-Level Functions cannot be used (see "Conditional Rules and ISF for Manipulating Grids" section on page 2-24 for details).
- Grid fields cannot be chosen as the "Triggering Field" when creating a data retrieval rule.
- Grid dictionaries and their fields cannot be chosen as the "Triggering Condition" for conditional rules—they can only be "Action" targets.
- A subset of conditional rule Actions are supported; and these apply to the column as a whole, not to individual cells.
- Requisition API (RAPI) cannot be used to submit services containing grid dictionaries.
- The use of grid dictionary fields for Business Engine namespace, Service item tasks, and Service Link agent parameters is not supported.
- Server-side conditional rules are not supported on a grid.

Designing a Grid

To design a grid for use on a service form, follow the steps below:

Step 1 Choose the fields you wish to have appear in the grid, through the "Show in Grid" check box in the dictionary's definition. See the "Setting Dictionary Fields to Display in a Grid" section on page 2-18.

Step 2 Choose "Display as Grid" when adding the dictionary to the active form component. See the "Setting a Dictionary to be Displayed as a Grid" section on page 2-19.

Step 3 Set grid options and display properties. See the "Grid Options" section on page 2-21, the "HTML Representation of Grid Fields" section on page 2-21, and the "Grid Properties" section on page 2-22.

For examples of using grids see the "Example: Laptop Selection" section on page 2-43 and the "Example: View All Laptops" section on page 2-58.

Setting Dictionary Fields to Display in a Grid

In the Dictionary Attributes section of a dictionary, check the check box in the "Show in Grid "column next to the fields you wish to show in a grid on a form. Fields set to "Show in Grid" and "Use" in a dictionary will display as columns in the grid on the service form (see Grid Properties, page 2-22) when the dictionary is set to "Display as Grid" (see Setting a Dictionary to be Displayed as a Grid, page 2-19).

In any new person-based dictionary, you will see "Show in Grid" checked by default for the Select_Person, Login_ID, Personal_Identification, Email_Address, and Home_Organizational_Unit fields, since these are also checked for use ("Use") by default. The same is true for the Name field a new service-item-based dictionary. "Show in Grid" cannot be unchecked for the Select-Person and Name fields, just as you cannot uncheck "Use" for those fields.

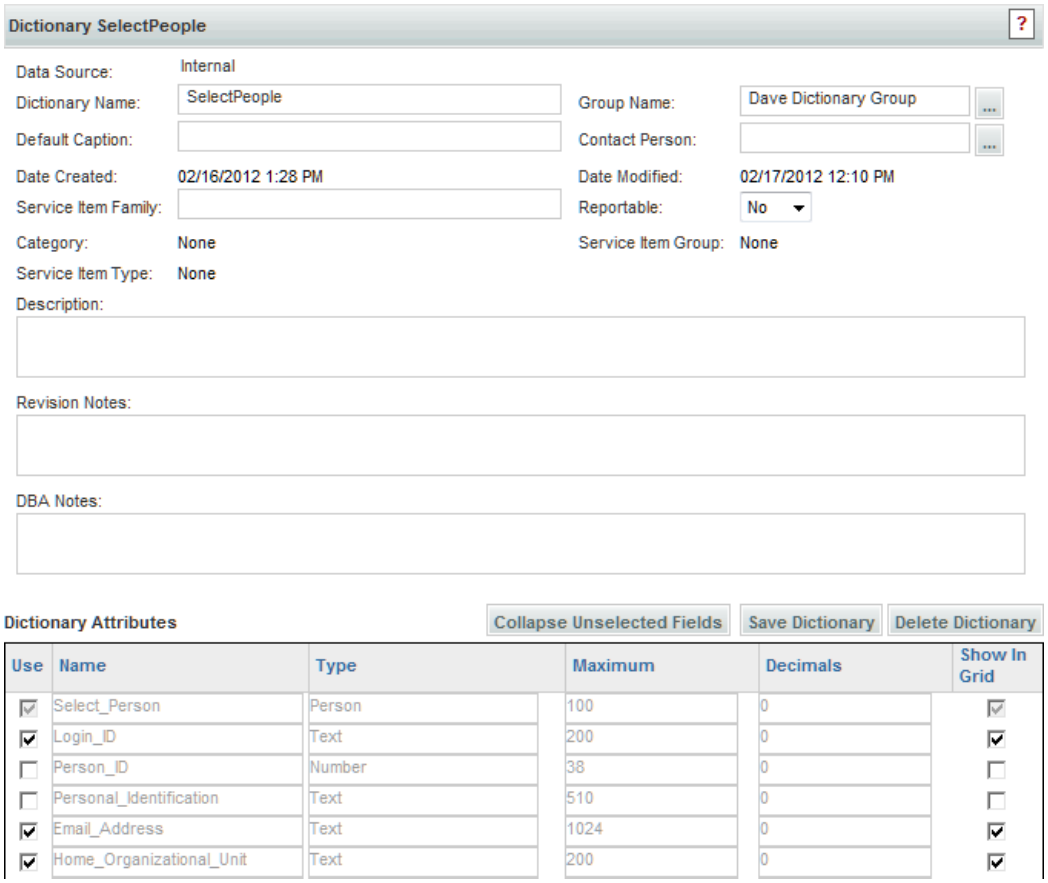
The total number of fields checked to "Show in Grid" in a dictionary cannot exceed the number set in the "dictionary.attributes.maximum.showingrid.count" property in the newscale.properties file (the default value is 20). In other words, grids cannot have more than 20 columns by default.

"Show in Grid" and Multivalue are mutually exclusive, as a grid does not have the capability for a multi-value cell.

Fields in External dictionaries cannot be used in grids and hence do not have the "Show in Grid" column.

Fields in Reserved dictionaries (Customer_Information and Initiator_Information) cannot be used in grids because those dictionaries inherently represent only one set of data.

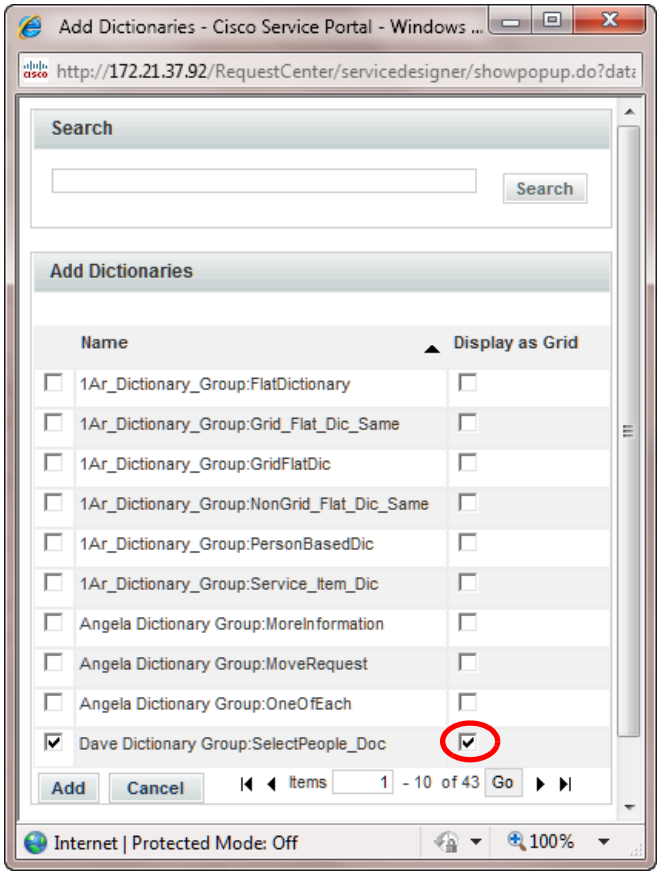
An example dictionary with fields set to "Show in Grid" is shown below.

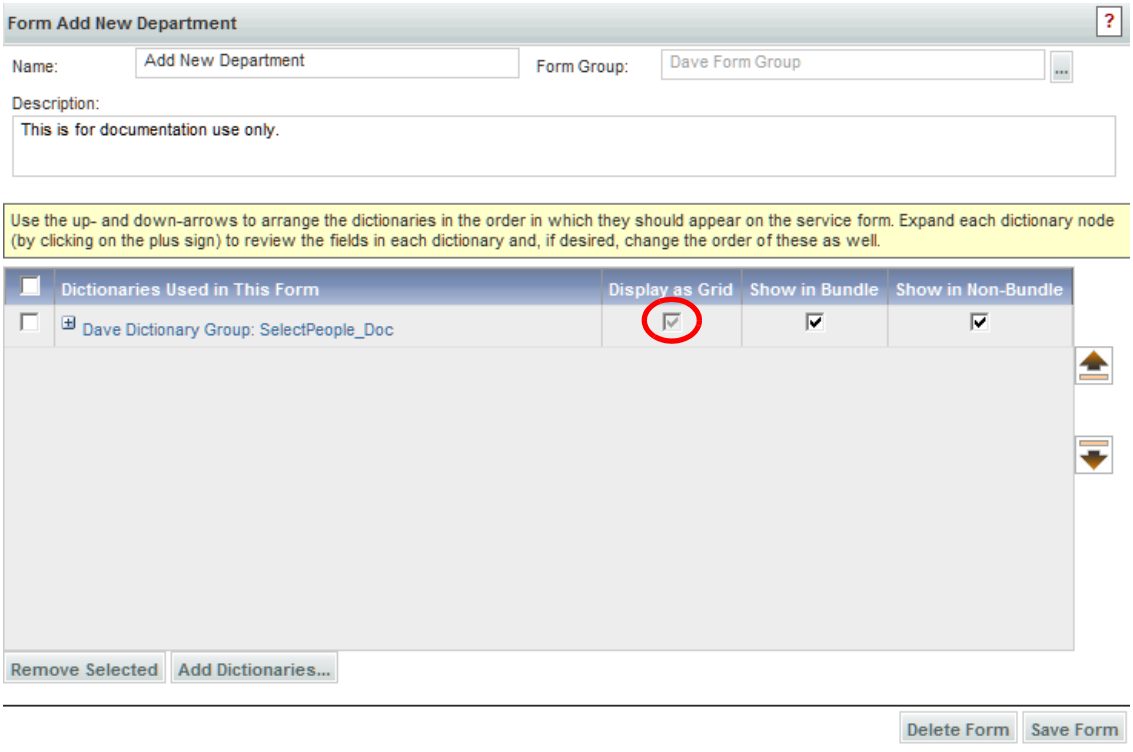


Setting a Dictionary to be Displayed as a Grid

Any internal dictionary (except Reserved dictionaries) can be set to appear as a grid by checking the check box “Display as Grid” on the Dictionary popup window that appears when adding dictionaries to include on a form from the Content tab, as shown below.

Only fields set to “Show in Grid” and “Use” will appear as columns when a dictionary is added with “Display as Grid” checked. Fields set to “Use”, but not to “Show in Grid” will not appear.

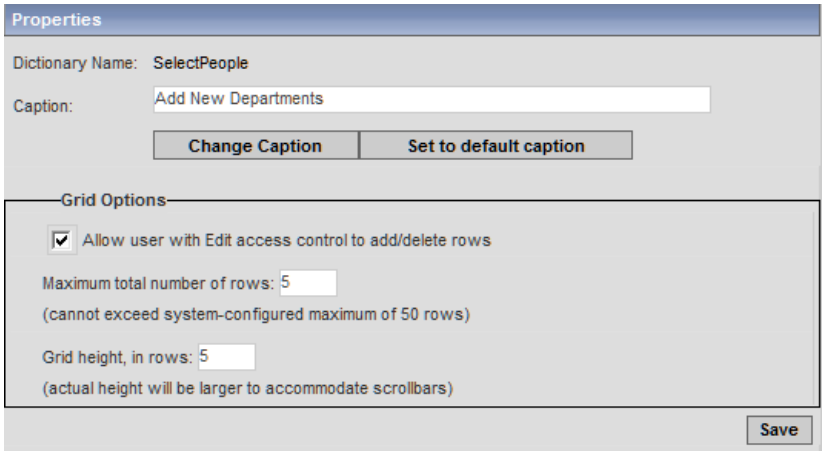

Add Dictionaries - Cisco Service Portal - Windows ...
http://172.21.37.92/RequestCenter/servicedesigner/showpopup.do?data
Search
Search
Add Dictionaries
Name
Display as Grid
1Ar_Dictionary_Group:FlatDictionary
1Ar_Dictionary_Group:Grid_Flat_Dic_Same
1Ar_Dictionary_Group:GridFlatDic
1Ar_Dictionary_Group:NonGrid_Flat_Dic_Same
1Ar_Dictionary_Group:PersonBasedDic
1Ar_Dictionary_Group:Service_Item_Dic
Angela Dictionary Group:MoreInformation
Angela Dictionary Group:MoveRequest
Angela Dictionary Group:OneOfEach
Dave Dictionary Group:SelectPeople_Doc
Add
Cancel
Items 1 - 10 of 43 Go
Internet | Protected Mode: Off
100%


Form Add New Department
Name:
Add New Department
Form Group:
Dave Form Group
Description:
This is for documentation use only.
Use the up- and down-arrows to arrange the dictionaries in the order in which they should appear on the service form. Expand each dictionary node (by clicking on the plus sign) to review the fields in each dictionary and, if desired, change the order of these as well.
Dictionaries Used in This Form
Display as Grid
Show in Bundle
Show in Non-Bundle
Dave Dictionary Group: SelectPeople_Doc
Remove Selected
Add Dictionaries...
Delete Form
Save Form

The “Display as Grid” check box on the Content tab appears checked for any dictionary you've added as a grid. The check box is always dimmed and cannot be changed on the Content tab; its purpose is to show which dictionaries on the form are in grid format. If you inadvertently add a dictionary as a grid, or forget to check “Display as Grid” when adding a grid dictionary to the form, you can simply remove the dictionary and add it again, with or without “Display as Grid” checked in the Dictionary popup window.

Grid Options

The Display Properties tab of a grid dictionary has an additional Grid Options section as shown below:



If “Allow user with Edit access control to add/delete rows” is checked, the grid will appear on the form with two buttons to add ( ) and delete ( ) rows to any user with Edit permission on the dictionary. If unchecked, these buttons do not appear, and users with Edit permissions on the dictionary cannot add/delete rows to the grid—only modify existing values.

The “Maximum total number of rows” controls the maximum number of rows that can be added. The default number is 5.

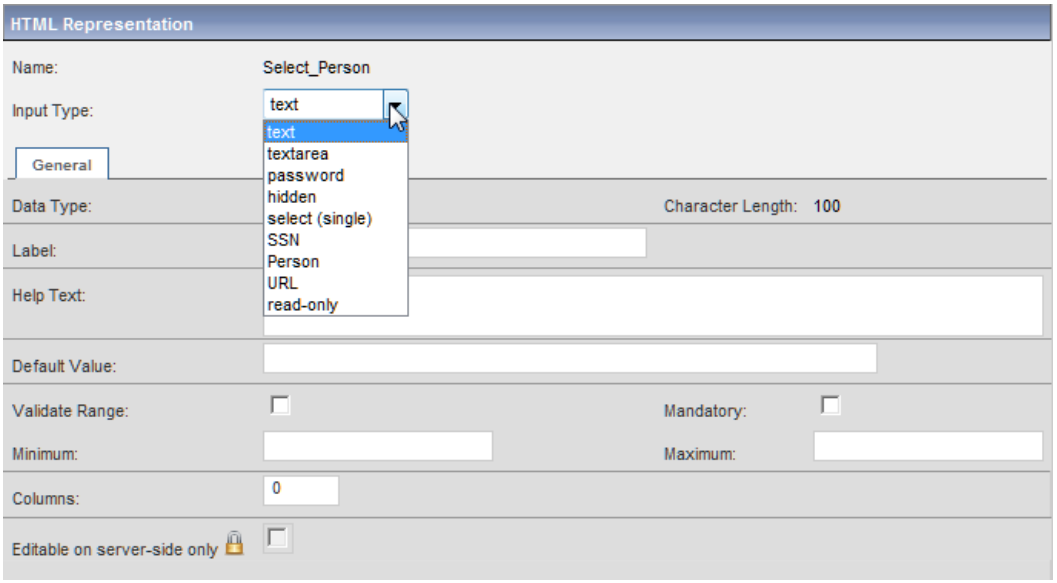
Note The maximum value cannot exceed the “dictionary.grid.maximum.rows.count” property set in the newscale.properties file. The default value is 50.

The “Grid height, in rows” setting controls the visible height of the grid. A vertical scroll bar appears when the number of rows entered exceeds this value. The default value is 5. The actual height will be larger to accommodate the scroll bars.

HTML Representation of Grid Fields

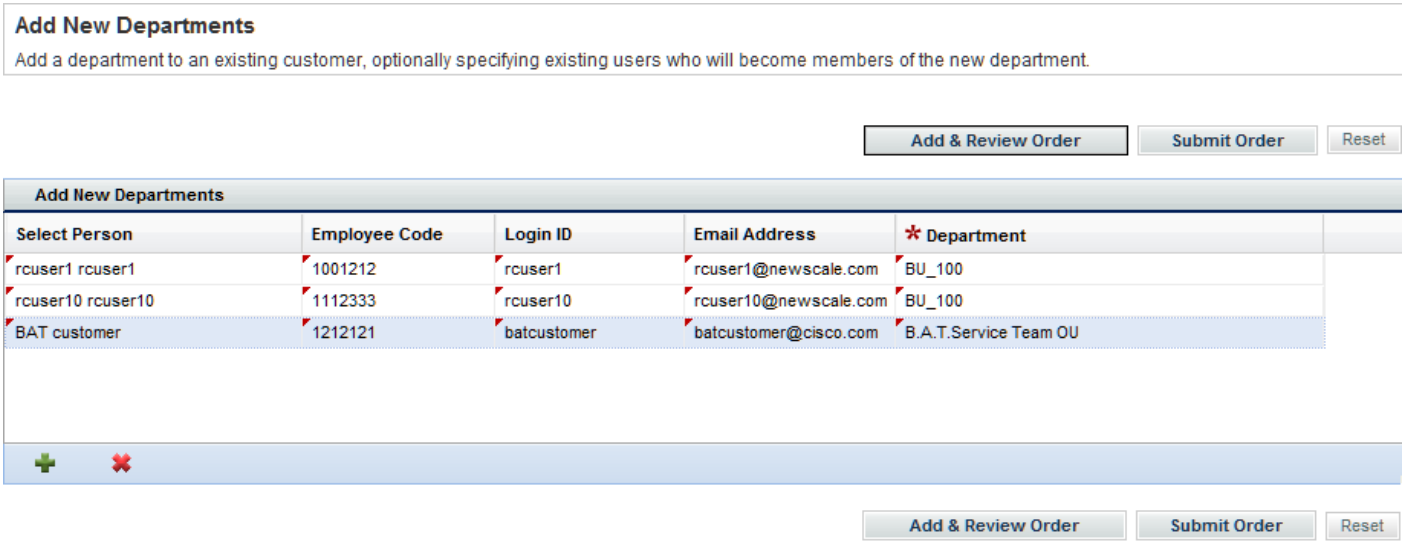
In the Display Properties tab of a grid dictionary, the HTML representations “select (multiple)”, “radio”, and “checkbox” are not available in the drop-down list of Input Types, as these controls cannot be rendered in a grid. Likewise, because you cannot configure Advanced Formatting or Buttons in a grid, these two controls do not appear.

An example HTML representation of a dictionary field set to “Show in Grid” is shown below:



Using a Grid on a Form

If a service has been designed with one or more dictionaries in the service set to "Display as Grid" (grid dictionary), then at time of ordering, the Service Request Form will display the dictionary as a grid, with those fields of the dictionary which were marked "Show in Grid" appearing as columns. An example form grid is shown below.



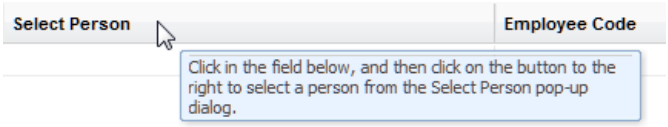
Grid Properties

The dictionary Caption in the Display Properties tab displays as the header of the grid. If the dictionary caption is blank, then the Properties tab appears as the header of the grid. The Label property in the Display Properties tab of a grid dictionary field displays as a column header.

Setting a grid dictionary field in the Display Properties tab to Mandatory causes the column header to be shown with a red asterisk (*), as shown above for the Department field.

A small red triangle appears in the upper left-hand corner of a text field to indicate newly entered text that has not been saved or submitted, as shown in the example above.

The Help Text in the Display Properties tab for a grid dictionary field appears as a tool tip for a column header, as shown below.



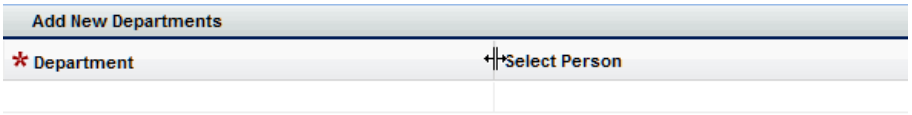
When you place the mouse over a cell in the grid, the content of the cell appears in a tool tip. This is useful when the cell content is longer than what is viewable in the cell.

You can use the Tab key to move to the next field of a grid, and to the Add and Delete buttons. When the Add button ( ) is clicked, an empty blank row is inserted at the end of the last row and highlighted.

Automatic retrieval of service item data after entering a service item name works for each row of the grid (see the "Example: Laptop Selection" section on page 2-43). Also, automatic population of service item details works when ordering the service from the Service Item Related Services tab. When the service form has a service item dictionary as a grid control, the service item details are copied to the end of grid. This way if other conditional or data retrieval rules have populated the grid, then the service item details are added at the end of the grid.

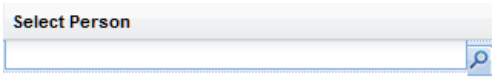
The grid's height is controlled by the "Grid Height, in rows" value (see Grid Options, page 2-21). If the grid has more rows than the specified limit, a vertical scroll bar appears.

The Columns value in the Display Properties tab for a grid dictionary field determines the width of each grid column. If the total width of all columns exceeds the width of the grid, an horizontal scroll bar appears. The grid column width can also be temporarily adjusted on the Service Form by positioning the cursor on the line between a column and the next. The cursor changes to a double line with two arrows pointing in opposite directions, as shown below.



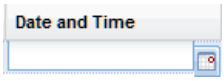
Once the new cursor appears, you can drag holding down the left mouse button to adjust the column width.

If the Data Type of a column is defined as Person, then a "Person Search" button is visible in the cell when the cell is clicked, as shown below.



Clicking the button will display the standard "Person Search" dialog box (see the "Person-Based Dictionaries" section on page 2-7). Person-based dictionaries can be configured as grids. Once the form user makes a selection through the Person Search, the other fields in that row that are used in the dictionary are automatically filled in with the current values of the corresponding fields in the selected person's profile.

Date/time fields can be used in grids, too. When a user clicks in such a field, a calendar icon appears:



When the calendar icon is clicked, a date or date/time dialog box appears for columns that have their dictionary Data Types defined as Date, or Date and Time, respectively,

If the Data Type is set to URL, the cell contents are rendered as a clickable link.

Clicking on the URL link opens your default browser to the entered URL.

Any error messages related to the grid appear on top of the grid. Each message appears as a link that can be clicked by the form user to go directly to the cell causing the error.

Conditional Rules and ISF for Manipulating Grids

Grid dictionaries are different from nongrid dictionaries in that they are rendered one way in the browser but stored another way in the WDDX generated in the database for the associated requisition entry. Because of this difference, not all conditional rule actions are supported; and those that ***are*** supported apply to an entire column of data. You will need a mixture of conditional rules and ISF functions to manipulate individual cells. Here is a quick summary of the conditional rule actions and ISF functions you can use on grids. (These details are also presented later in the sections on conditional rules and ISF, but are summarized here if you are familiar with both those features and just want to understand how they apply to grids.)

Conditional Rules as Applied to Grids

Grid dictionaries and their fields cannot be chosen as triggering conditions for rules; they can only be chosen as the targets of conditional rule actions. The following conditional rule actions, which apply to individual fields in nongrid dictionaries, apply to an entire column in a grid:

- Show Fields
- Hide Fields
- Make Read-Only
- Make Writeable
- Enable
- Disable

All other conditional rule actions are not supported on grid columns or cells.

Note that in case of a grid, there is no observable difference between the Make Read-Only and Disable actions (whereas there is a slightly different user interface effect seen in nongrid fields). The same is true for the Make Writable and Enable actions.

ISF in Grids

JavaScript functions cannot be assigned to any field-level event within a grid dictionary.

The existing ISF framework has been extended to support grid dictionaries and fields, as described in the tables below.

Table 2-1 Dictionary-Level Functions:

Function	Usage in Grid
serviceForm.***dictionaryName***.setVisible (Boolean)	Hides or makes visible the Grid. Has no effect on a dictionary which is hidden via Service Designer.
serviceForm.***dictionaryName***.isVisible()	Returns true if the Grid is visible and false otherwise.
serviceForm.***dictionaryName***.getCaption(Boolean stripTags)	Gets the title text for the grid, optionally stripping any HTML.
serviceForm.***dictionaryName***.setCaption(String newCaption)	Sets the title text for the grid.
serviceForm.***dictionaryName***.setReadOnly (Boolean)	Sets all the columns in the dictionary to be read-only or read-write; has no effect if in Service Designer the columns were already set to read-only.
serviceForm.***dictionaryName***.isReadOnly()	Returns true if the dictionary is read-only.
serviceForm.***dictionaryName***.getGridSize()	Returns the number of rows in the grid.

Table 2-2 Field-Level (Grid Column-Level) Functions

Function	Usage in Grid
serviceForm.***dictionaryName***.fieldName.setReadOnly(Boolean)	Makes the column read-only or read-write.
serviceForm.***dictionaryName***.fieldName.isReadOnly()	Returns true if the column is read-only and false otherwise.
serviceForm.***dictionaryName***.fieldName.setVisible(Boolean)	Makes the column hidden or visible (displayed). If the column is hidden via Service Designer settings, it cannot be made visible.
serviceForm.***dictionaryName***.fieldName.isVisible()	Returns true if the column is visible and false otherwise.
serviceForm.***dictionaryName***.fieldName.getInstructionalText(stripTags)	Returns the instructional text for a column, optionally stripping any HTML from the text based on the Boolean stripTags argument.
serviceForm.***dictionaryName***.fieldName.setInstructionalText(text)	Sets the instructional text for a column.
serviceForm.***dictionaryName***.fieldName.getPrompt(stripTags)	Returns the column header—optionally stripping any HTML from the prompt based on Boolean stripTags argument.
serviceForm.***dictionaryName***.fieldName.setPrompt(prompt)	Sets the column header.

Table 2-3 Grid Cell-Level Functions

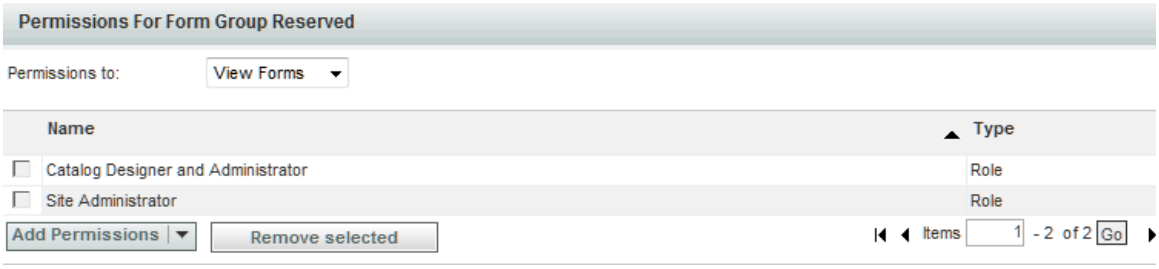
Function	Usage in Grid
serviceForm.*dictionaryName* .*fieldName*.getCellValue (RowIndex)	Returns the cell value of the column at specified RowIndex.
serviceForm.*dictionaryName* .*fieldName*.setCellValue (RowIndex, Value)	Sets the cell value of the column at the specified RowIndex.

Reserved Customer-Initiator Form

The Customer-Initiator form is provided in all application instances, in the "Reserved" form group. This form can be configured using the same capabilities as are available for user-defined forms, with the exception of Access Control settings. If the customer and initiator are the same (that is, a person is ordering for him/herself rather than on behalf on someone else), the Initiator_Information dictionary is hidden. The Customer Information dictionary is displayed with the participant permissions specified via Access Control. Default values for fields in both dictionaries are supplied and available for subsequent use.

The Customer-Initiator form should typically be included in every service, so that client-specific details about the Customer and Initiator that are not included in standard My Services and Service Manager displays, or in the corresponding query subjects in the Data Marts, can be recorded and accessible. You will, therefore, need to standardize the attributes included in both the dictionaries and the form, so that all attributes required by diverse services are available. This may entail including attributes that are then hidden by form rules in services for which they are not relevant.

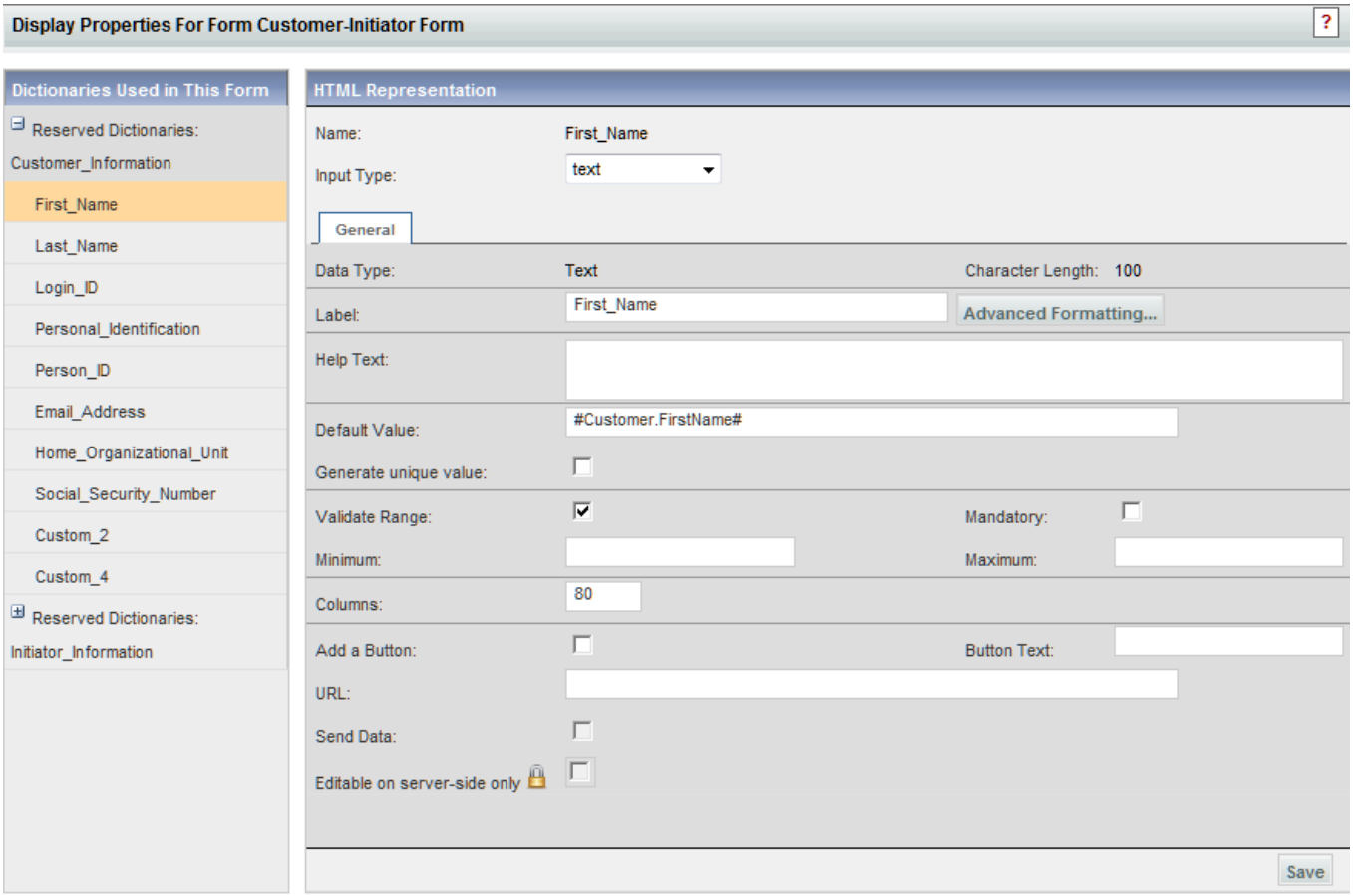
Access to the reserved form group should be restricted (via object permissions), to prevent unauthorized changes to this form. By default, members of the "Site Administrator" and "Catalog Designer and Administrator" roles are allowed to "Design Forms" in this group. Membership in these roles should be tightly controlled.



Lightweight Namespaces

Any forms that include person-based dictionaries use lightweight namespaces to provide the values to the form fields, based on corresponding values in fields stored in the profile for the selected person. This includes both the Customer-Initiator form and any user-defined forms. Most lightweight namespaces have the format #Customer.**FieldName**#, or #Initiator.**FieldName**#. However, some, like the components of the location (address) may be more complex; for example, #Customer.DetailedLocation.Office#. See the "Lightweight Namespaces" section on page 6-33 for a complete list.

Note The use of grid dictionary fields for lightweight namespaces is not currently supported.

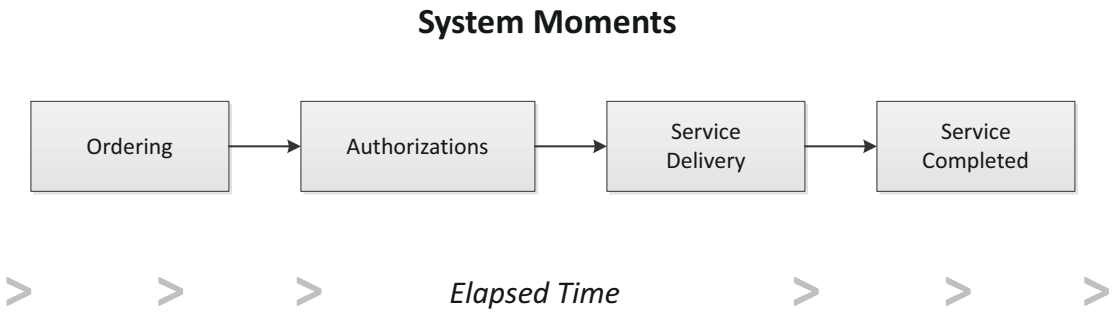


The namespace is automatically supplied as the default value for the field. If desired, the initial assignments can be replaced. However, there is no automatic way to restore the original default value; you must type the appropriate namespace. Lightweight namespaces are listed and described in Chapter 6, "Namespace".

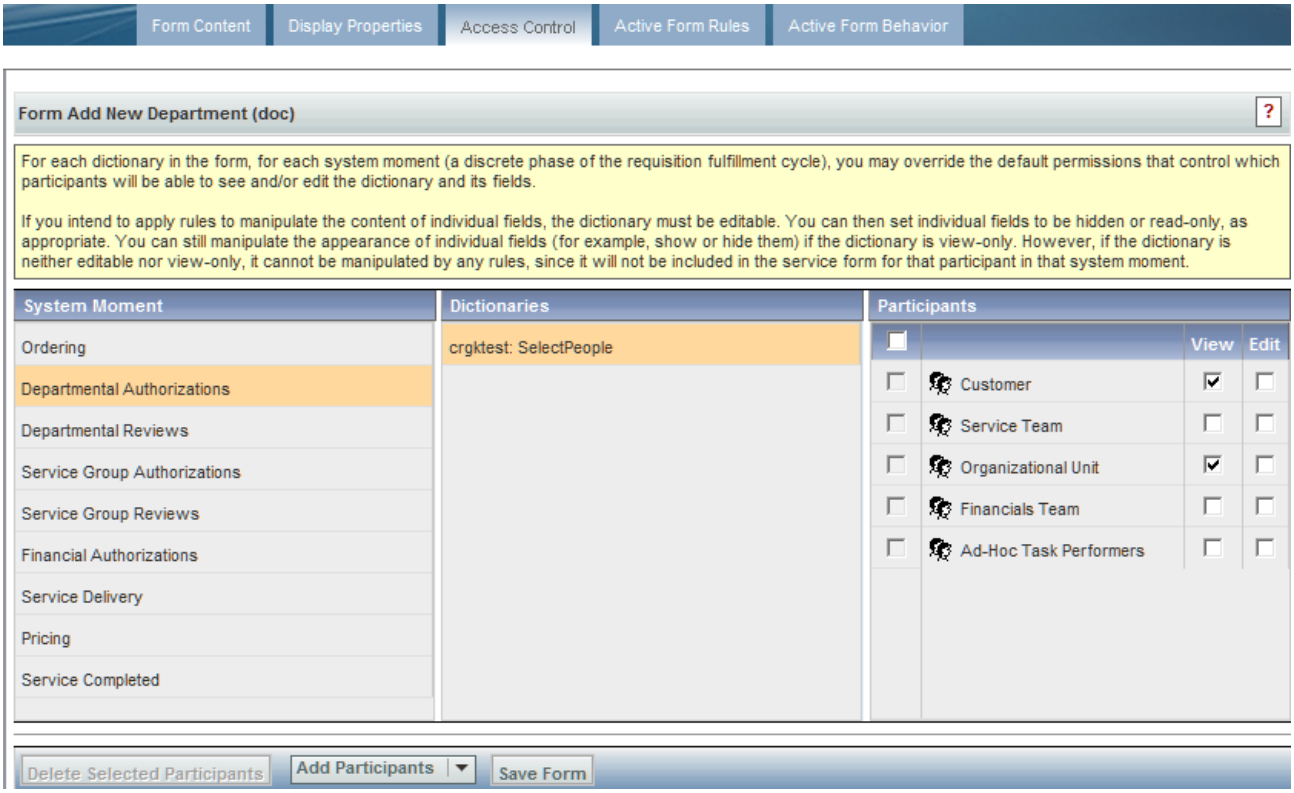
Lightweight namespaces referring to customer or initiator data can also be used as default values for fields in dictionaries other than the reserved dictionaries. In this case the service designer will, of course, be responsible for mapping from the dictionary field to the appropriate person attribute. This capability allows you to define dictionaries that contain both person-based and other data.

Access Control

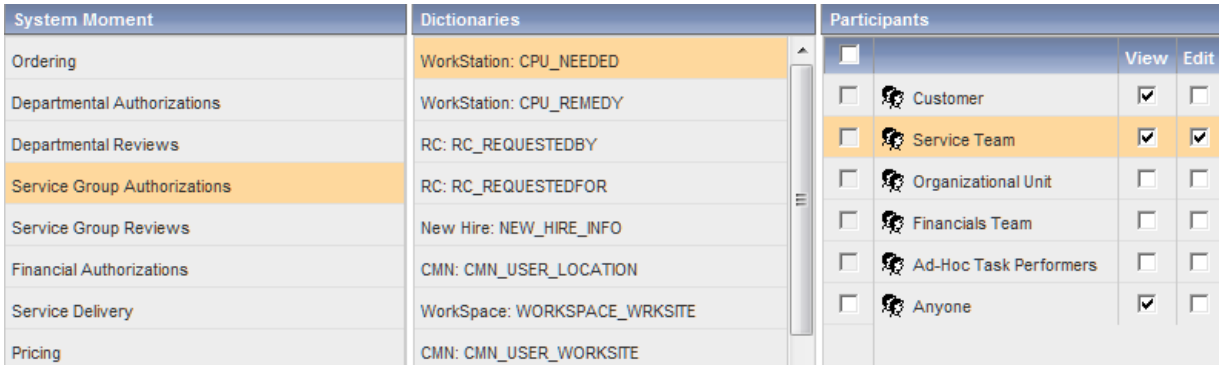
Access Control capabilities determine which users may view or edit dictionaries in the form during all moments that comprise the requisition life cycle. The "System Moment" tracks what point of the requisition life cycle the requisition is in:



All requests start in the Ordering moment. The request remains in the Ordering moment until it is submitted. The customer/initiator is the only participant during the Ordering moment. Only those dictionaries in which the initiator must provide the details of the request are typically editable. Any dictionaries used solely by approvers, reviewers or task performers have no access assigned.



A request may have zero or more Authorization moments, depending on which authorizations and reviews have been configured for that service. The participants, and the access control granted to each, will vary based on the type of authorization and the nature of the action required. Typically, one or more dictionaries are editable in such moments, to allow authorizers/reviewers to adjust data previously entered or enter new information such as a reason why the request was approved or rejected.



The typical participants in the delivery plan are listed and can be assigned appropriate access controls by checking (or unchecking) the desired permissions:

Participant Type	Description
Customer	The user who completes and submits the service order form; typically has View access to dictionaries in the Service Delivery moment, except those that contain confidential information
Service Team	Service performers and managers who are members of the service team OU which "owns" the service group where this service resides
Organizational Unit	Members of the customer's Home OU, such as those who review or authorize service requests
Financials Team	Refers to the OU, if any, which is configured in Administration as the Financials Team for the purpose of site-wide Financial Authorizations
Ad-Hoc Task Performers	Any performer, or queue members, who receive an Ad-Hoc task

The "Service Team" participant refers to the service team that owns the service group in which the service resides. If you include this form in multiple services, the service team participant for each service may vary, depending on what group the service is in. Therefore, it may be preferable to "Add Participants" to the form, explicitly indicating the service teams that are needed to access dictionary data at this time. For services with complicated delivery plans, each task in the plan may need to be performed by a different service team. In this case too, additional participants should be added.

The Customer-Initiator form presents additional concerns, since it will likely be used in virtually every service. Therefore, especially if the Customer and Initiator dictionaries should have View only permissions, it might be most efficient to "Add access for Anyone" via the Additional Participants button.

Specifying that a dictionary can be viewed but not edited during a particular moment has the effect of rendering all fields in the dictionary as HTML labels (read-only text).

- The text for fields that can have multiple selections (check boxes, multi-select drop-down lists) will show any options previously chosen for those fields as a comma-separated list of values.
- The label for a Person field shows the name of the person previously chosen. The accompanying Search button is disabled. A second (hidden) object contains the unique identifier of the person chosen.
- URLs are rendered as an HTML label that is hyper-linked.
- Help text for the fields in the dictionary is not displayed.
- Conditional rules that attempt to set a value in the field are ignored.

Since HTML DOM objects (that is, <input> tags) do not exist for fields in noneditable dictionaries a limited subset of rules or ISF functions can be applied. These limitations are documented in the detailed discussion of conditional rules and ISF functions which follows.

The access levels are applied at the dictionary-level. In a particular moment, the entire dictionary, comprising all its fields, is either hidden; displayed, but with all its displayable fields rendered as boilerplate text; or displayed with its fields rendered as a set of HTML input objects which can be edited by users or manipulated by rules.

All access control assignments pertain to all tasks performed as part of the delivery plan. If you need to assign different permissions to dictionaries for different users in different tasks, you will need to do this via a conditional rule. The Access Control setting should always grant all permissions that are required on a particular dictionary for a particular user/group. Conditional rules can remove access privileges, but cannot grant them if they are not originally specified via the Access Control tab.

Assigning permissions via the Access Control tab has important ramifications for designers specifying active form rules and for ISF programmers:

- For hidden dictionaries: Hiding a dictionary via Access Control means that no objects defined in that dictionary are present in the service form for the moments and participants for which the dictionary is hidden. Therefore, it is not possible to use rules or ISF to manipulate dictionary or field contents or visibility.
- For view-only dictionaries: Fields in such dictionaries cannot be made editable. However, rules or ISF can be used to get the current value of a field in a dictionary set to view-only by Access Control; to temporarily hide the field from view; or to apply other functionality provided by the framework.
- For editable dictionaries: Full capabilities are available to manipulate the appearance and behavior of the dictionary or fields within the dictionary.

If you need to manipulate the values of fields in a dictionary that are hidden from the user (for example, to populate a set of fields with default values, without the user being aware of those values), you can configure the form rules that act upon these dictionaries to be executed on the server side (see the "Server-Side Events" section on page 2-32).

Setting a dictionary's Access Control to "None" (that is, with neither View nor Edit checked) prevents the dictionary from being sent to the browser. However, you can still configure form rules to store data in such a dictionary, provided those rules execute on the server (see the "Server-Side Events" section on page 2-32). This is a powerful technique for keeping the service form as rendered in the browser as small as possible, while still collecting and storing the results of the browser session in the database.

Administrative Override of Access Control Settings

Access controls assigned via Service Designer are ignored when the service form is run by a user who has been granted the "Manage Service Dictionaries" capability. (This capability is included in the "Site Administrator" and "Distributed Service Designer" roles.) For such users, all dictionaries are editable in all moments, regardless of the access controls specified. This capability is provided for ease of developing and testing the form appearance and should be assigned sparingly.

Service-Specific Additions to Access Control Settings

Access control that must be granted to a form component used in multiple services may need to vary on a service-by-service basis, depending on the service teams involved in the fulfillment of each request. Additional participants for viewing/editing dictionaries can be added to the service definition. On the Form tab of the service, choose the dictionary, then add the appropriate Participants. No other changes can be made to Access Control settings at the service level.

Active Form Rules

The application supports two types of Active Form Rules:

- Data Retrieval rules retrieve data stored in a relational database and either display the values returned into fields on the current form, or validate data on the form against the retrieved results.
- Conditional rules specify a set of conditions that must be met in order for a set of actions to be carried out. (Calling these rules "conditional" is a bit of a misnomer—the set of conditions can be empty, so that the rule is applied whenever its triggering event occurs.)

Most of the rule definition is done "declaratively"—that is, rule wizards help you define the rules, by walking you through a series of steps. In most cases, you don't have to write any code at all, just answer a series of questions. When a user orders a service containing the form, the rules are retrieved and code that operates in the context of a web page is automatically generated.

Form and Field Trigger Events

A critical part of defining all rules is specifying when the rule will run. Each rule may have one or more "triggering events", events which occur while a user is working with a service form and which cause the rule to be executed. For example, loading a service form into the browser (when a customer orders a service or a service performer opens the form in Service Manager) is a "form load" event—"When the form is loaded". During this event, typical conditional rules might hide dictionaries or fields not relevant to the current context, while data retrieval rules might prefill default values on the form. When a user changes the value of the field is a "field" event—a "When an item is changed". Typical rules attached to such an event might supply values to a drill-down. For example, only once a user has chosen a region does a drop-down list with facilities within that region populate.

Form- and field-level events supported by the service form framework are summarized below:

Form Event	Description
When the form is submitted (browser-side)	The user has pressed the button to indicate he/she is done with the form. The button may be labeled "Submit Order", "Update", "Done", depending on the context, but pressing any such buttons triggers the form-submitted event.
When the form is loaded (browser-side)	The service form is displayed in the user's browser. The user has chosen the service to work with (when ordering a new service); a service performer asks to display the service form data, to complete any data entry required for the current task and to review data previously entered.
When the form is unloaded (browser-side)	Processing of the form is complete.
Before the form is loaded (server-side)	The service form is being prepared on the server prior to being sent to the browser. (See the "Server-Side Events" section on page 2-32.)
After the form is submitted (server-side)	The service form has been submitted on the browser and sent to the server for processing. (See the "Server-Side Events" section on page 2-32.)

Field Event	Description
When the item is changed	The user types something into the specified form field.
When the item loses focus	The user exits from the specified form field, either by tabbing to another field or clicking the mouse in another field.
When the item is clicked	The user clicks on a check box, radio button, or item in a drop-down list.
When the item is focused on	The user clicks in the specified field.
When the mouse is moved off the item	The user moves the mouse off the specified field.
When the mouse is on the item	The user moves the mouse onto the specified field.

There are currently no field events triggered within a grid.

In the table above, the last two form events are considered “server-side” events and are shown with a lock icon to indicate a more secure event. They provide flexibility for service designers to apply form rules before/after the service form is sent to/received from the browser. Data that is not meant for user consumption (for example, predefined attributes for workflow or external system integration) can be excluded from the service form shown to the user and manipulated after form submission with form rules. Using server-side events can be useful not only for keeping the amount of data sent to the browser to a minimum, but also for ensuring the highest possible level of security.

All other form- and field-level triggers are considered “client-side” or “browser-side” events. They are used primarily for actions that are display-oriented, or actions which require interaction with the user. Making a field read-only, marking it mandatory, or setting focus to it are all browser-specific actions. For this reason, ISF and JavaScript functions take effect only when they are associated with client-side events.

More details about the differences between server-side and client-side events are described below.

Server-Side Events

The two form-level events that execute server-side—pre-Load and post-Submit—provide powerful opportunities for executing rules on the server, as opposed to within the browser session. Such rules are commonly referred to as “server-side rules”. Server-side rules have the following properties:

- As opposed to client-side rules that act upon the service form loaded on the browser, server-side rules act upon any dictionaries in the service form. This includes dictionaries that are never sent to the browser (that is, to which users have neither View nor Edit permissions).
- The powerful behavior described above means that if you attach one or more data retrieval rules to the pre-Load event, those rules take effect on all dictionaries, for all system moments. Therefore if you create a form rule intended to prefill data at the Ordering moment but be ignored in subsequent system moments when the dictionaries become read-only, you should not associate it with server-side events.
- The form is rendered in the browser only after the pre-Load data retrieval completes. This often provides a better user experience as the time between initial rendering of the service form on the browser and the completion of client-side events can be reduced. However, depending on the size and complexity of the form, the end-user's *perception* of the form load may be better if you minimize the pre-Load data retrieval.
- Unlike client-side rules triggered at form load time, form rules that are triggered at preload time are not re-executed when the form is refreshed after hitting validation errors.

Data Retrieval Rules

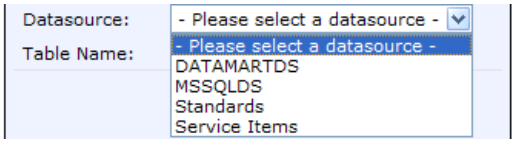
Data retrieval rules retrieve data stored in a relational database and either display the values returned onto fields on the current form or validate data on the form against those values. They perform this retrieval by executing a SQL query against the source database, returning the values of the columns you have specified to the service form.

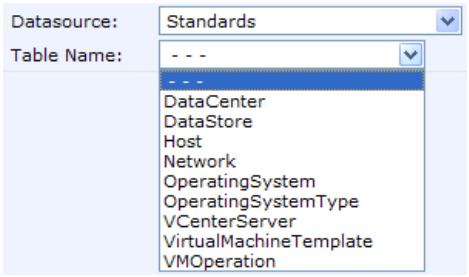
Prerequisites to Creating Data Retrieval Rules

Datasource. In order for Service Portal to access a database, a corresponding **"**datasource**"** must be defined. A datasource identifies the database and supplies information for connecting to it, including a valid user name and password. The application comes with one datasource preconfigured, named REQUESTCENTERDS, which provides access to Service Portal data. A database administrator, in conjunction with a system administrator, will need to define any additional datasources, containing company-specific data, and publish the definitions to the application servers on which Service Portal is installed. Detailed instructions for configuring and installing datasources are given in the *Cisco Service Portal Installation Guide*.

Standards and Service Items can be used as a source of data in data retrieval rules. However, their usage is restricted to a rule with the Query Type of "Database Table Lookup"; they are not available for use in rules that use hand-written SQL.

When you define a database table lookup, the list of datasources includes Standards (and the Service Items), as well as the transactional and Data Mart databases. If you choose "Standards", the available Standards tables appear, and one can be chosen as the Table Name for the query. All other aspects of composing the data retrieval rule are identical to those explained previously in this section.

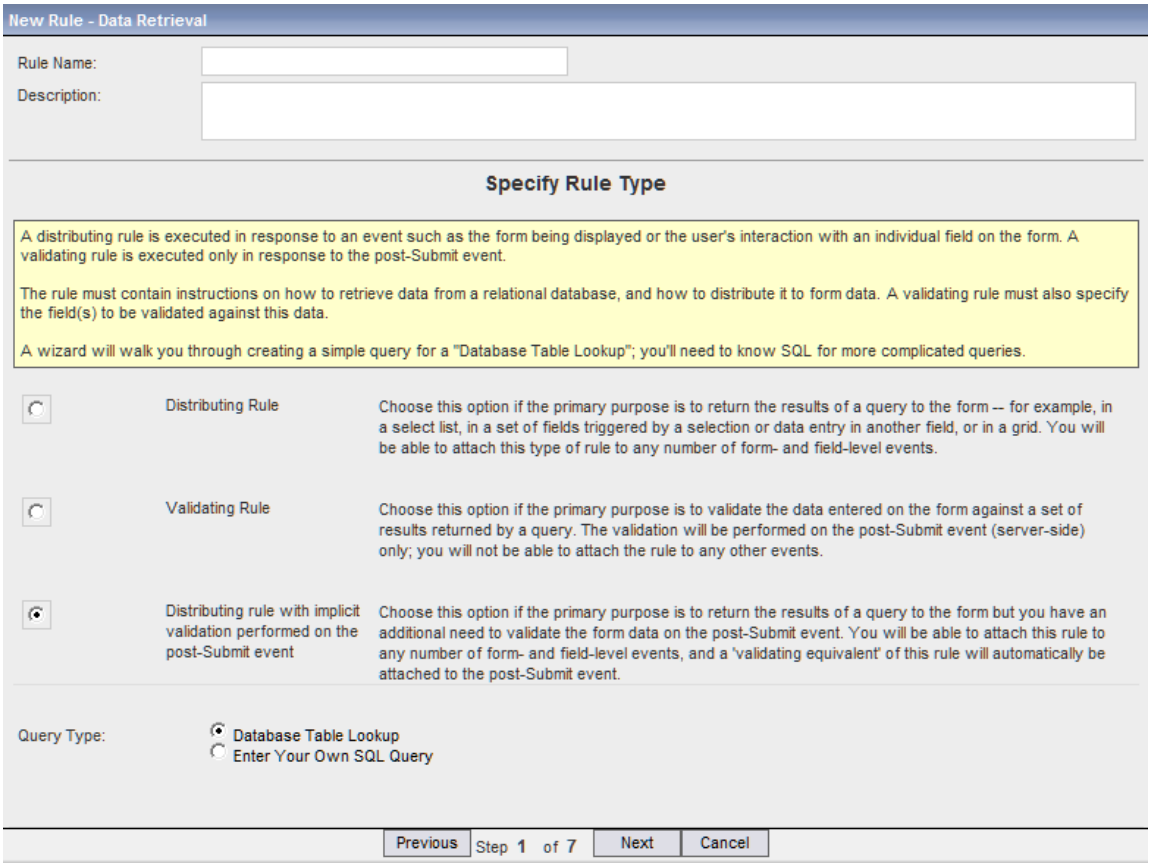




Source Data. You will also need to know the structure of the data you need to retrieve, and the table or tables in which it is stored. An IT specialist knowledgeable in the source system can typically supply this information. If all of the data you need to retrieve resides in a single table, you can simply specify the name of the table and columns to retrieve; Request Center will automatically build a SQL query which retrieves all columns in the table. If, however, you need to retrieve data from multiple tables or to manipulate the values (for example, concatenate values together or perform calculations) for use on the service form, you will need to write and test a SQL query yourself. A database or IT specialist with knowledge of the source system and access to tools for developing SQL are indispensable for this task. Once the query is tested, you can cut and paste it, with minor modifications outlined here, into the Data Retrieval Rule Wizard using a Query Type of "Enter Your Own SQL Query".

Creating a Data Retrieval Rule

You create a data retrieval rule by choosing the **Active Form Rules** tab in the Active Form Components option, clicking **New Rule**, and asking to create a **New Data Retrieval Rule**. The first page of the Data Retrieval Rule wizard appears, as shown below.



Fill in each page of the wizard, then click **Next** to proceed to the following page. The last page of the wizard will contain a summary of the rule just defined. Click **Save** to save the rule, or **Previous** to return to a previous page and change the definition.

The first page of the Data Retrieval Rule wizard contains the following fields:

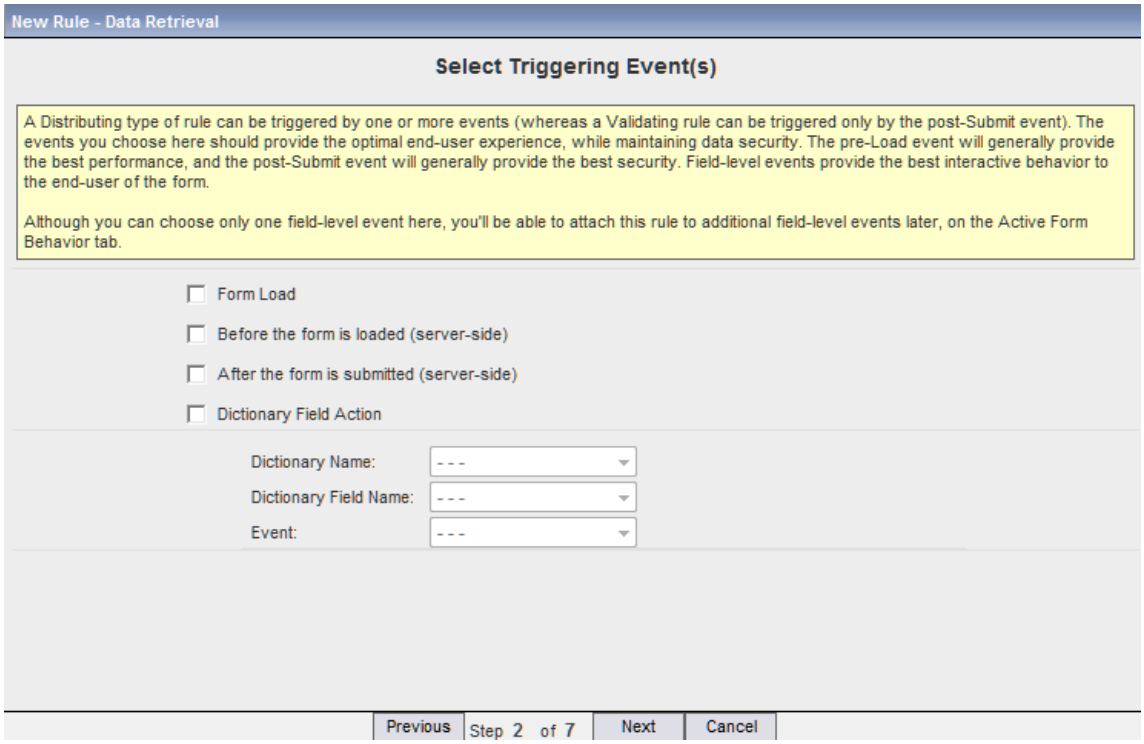
- **Rule Name** and **Description**: Enter the name of the rule and a description. This can be any word or phrase. Rule names should be mnemonic, clearly indicating the dictionary and field (if applicable) and triggering event. All rules defined within a form are listed on the left of the Active Form Rules tab. Reviewing and maintaining rules are much easier if you can clearly tell—from the rule name—which rule you want to view or edit. Additional information can be added in the Description field.
- Specify Rule Type:
  - **Distributing Rule**: Choose this option if the primary purpose is to return the results of a query to the form—for example, in a select list; in a set of fields triggered by a selection or data entry in another field; or in a grid. You can attach this type of rule to any number of form- and field-level events.
  - **Validating Rule**: Choose this option if the primary purpose is to validate the data entered on the form against a set of results returned by a query. The validation is performed on the post-Submit event (server-side) only; you cannot attach the rule to any other events.

  - **Distributing Rule with implicit validation performed on the post-Submit event**: Choose this option if the primary purpose is to return the results of a query to the form but you have an additional need to validate the form data on the post-Submit event. You can attach this rule to any number of form- and field-level events, and a "validating equivalent" of this rule will automatically be attached to the post-Submit event.

- Query Type: **Database Table Lookup** or **Enter Your Own SQL Entry**. Database Table Lookup is the quick-and-dirty option. If all the data you want is available within one table (or within a database view that a Database Administrator has created), use this type. The wizard will walk you through a set of screens to define your query, all by filling in dialog boxes. If, on the other hand, you need a more complex query, you must write it in Structured Query Language (SQL) with a Query Type of "Enter Your Own SQL Query".

Choose Triggering Events (Distributing Rules)

If you specified the Rule Type as a **Distributing Rule** or **Distributing Rule with implicit validation performed on the post-Submit event**, you need to choose one or more triggering events, as shown below.



- Choose Triggering Events: The rule can be executed (triggered) by checking the check boxes for the following events:
  - **Form Load:** When the service form is initially displayed in My Services or Service Manager.
  - **Before the form is loaded (server-side)**: This server-side rule is executed on the server prior to sending it to the browser. (See the "Server-Side Events" section on page 2-32.)
  - **After the form is submitted (server-side)**: Available only for a "Distributing Rule", this server-side rule is executed on the server after the form is submitted on the browser and sent to the server for processing. (See the "Server-Side Events" section on page 2-32.)

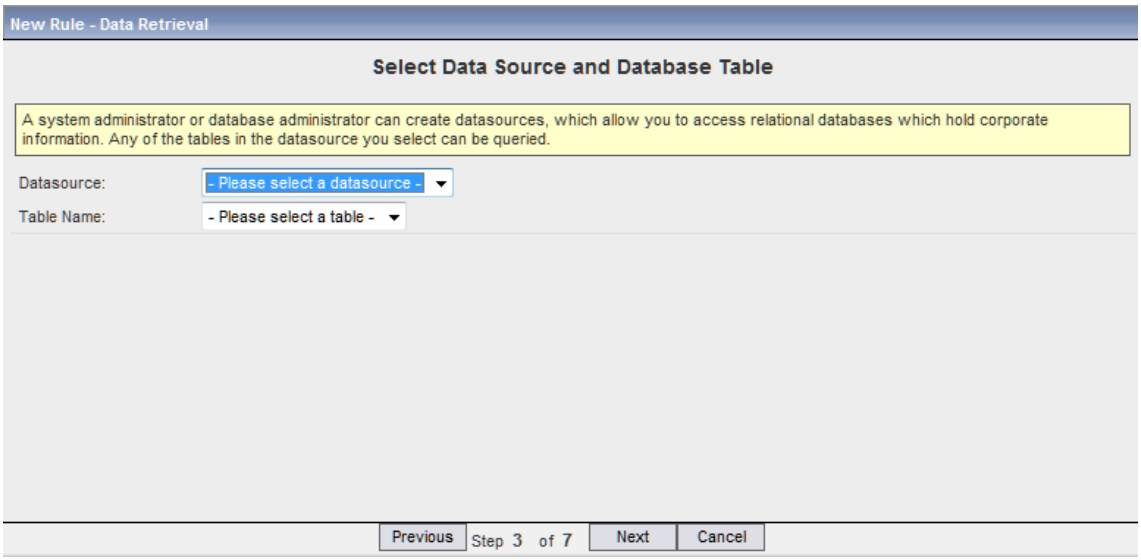
  - **Dictionary Field Action**: In response to an event that occurs as a user is working with the form, filling out data and moving from field to field.

- **Dictionary Name**, **Dictionary Field Name**, and **Event**: If you specify that the Event is a "Dictionary Field Action", you must define that action by choosing the Dictionary Name, Dictionary Field Name, and Event from the drop-down menus. The "triggering" events are similar to, but not identical to, events that web page designers may be familiar with. The list of available events may vary, depending on the input type assigned to the field. For example, a radio button has an event "When the item is clicked", which is not applicable to a text field.

Note Grid fields cannot be chosen as the "Triggering Field" when creating a data retrieval rule.

Database Table Lookup – Data Source and Data Table

If you designated a "Database Table Lookup" as the Query Type, you must specify the datasource and table in which the data reside. Once you specify the datasource, the drop-down list for the Table Name is populated with all tables accessible in that datasource.



The system offers the following data sources:

Data Source	Description
REQUESTCENTERDS	Contains the tables in the Service Portal database.
DATAMARTDS	Contains the tables in the Data Mart database.
Standards	Contains the tables you created using Service Item Manager's **Design Standards** tab or imported using Service Item Manager's **Import Data** tab. See the "Using Standards in Data Retrieval Rules" section on page 3-29 for more information.
Service Items	Contains the Virtual Machine service item; the ServiceItemHistory and ServiceItemSubscription tables; and any service items defined in Service Item Manager's **Design Service Items** tab. See the "Using Service Items in Data Retrieval Rules" section on page 3-28 for more information.

Database Table Lookup – Lookup Conditions

Database Table Lookup rules typically need to have an associated "Where" clause—criteria that specify the row or rows which are retrieved from the specified table. These criteria consist of one or more conditions that must be met for the rows to be retrieved. The conditions may compare the value of a column in the table to either a literal or the current value of a field on the form.

For example, a condition

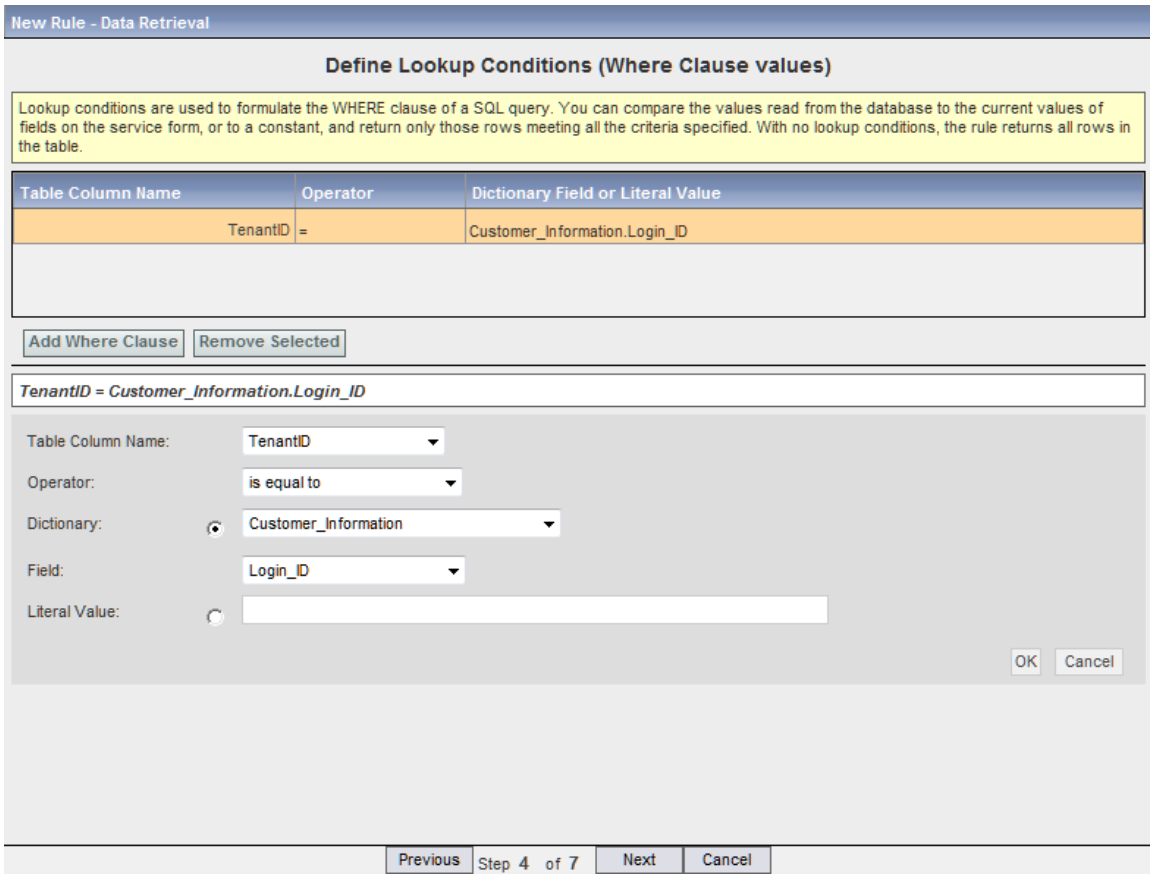
```
AssetId = Memory.AssetId
```

would retrieve the rows in the Memory table where the column AssetId was equal to the current value of the AssetId on the service form. Such a rule would typically be applied as a Change event for the AssetId field.

Any number of conditions may be entered. As you enter each one, click **OK**. The condition just specified will appear at the top of the page. If you use multiple conditions, all must be true (that is, the conditions are AND'ed together) for a row to be returned.

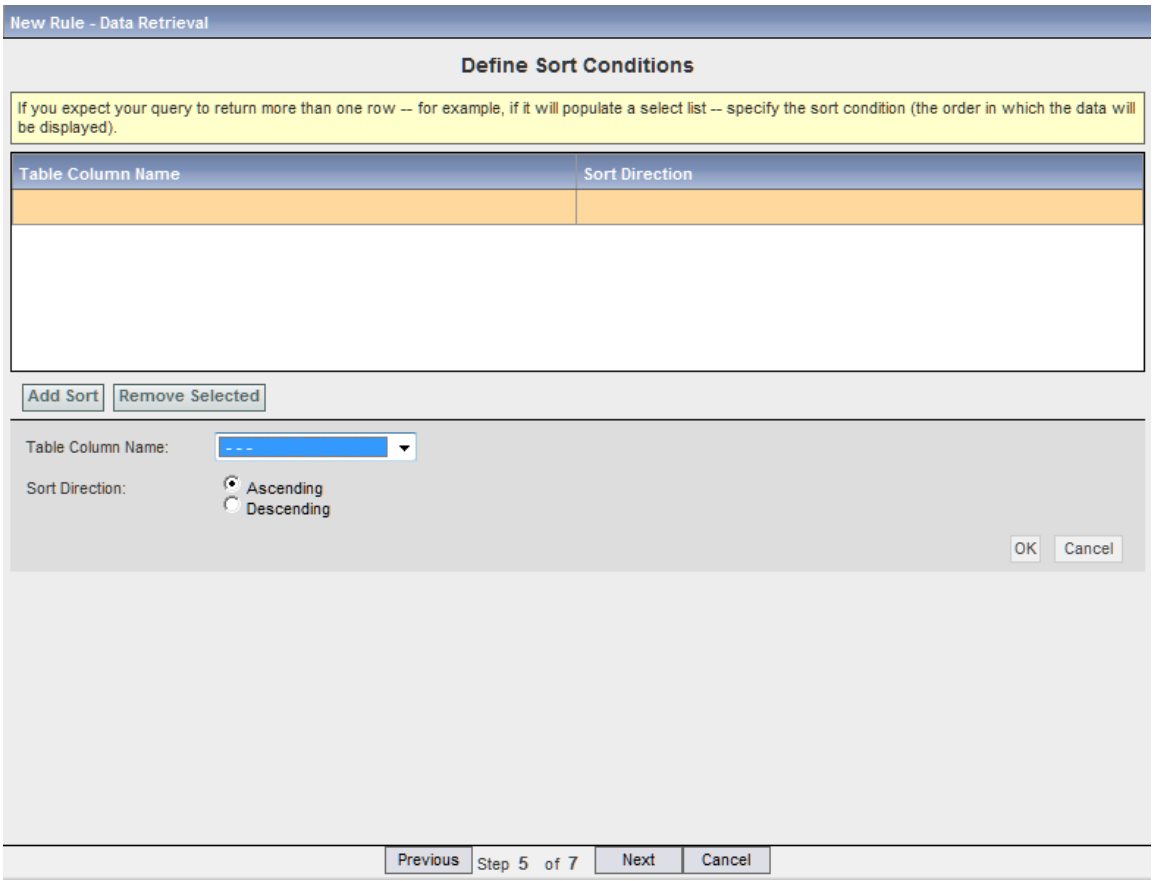
If no Lookup Condition is specified, all rows in the specified table are returned. This typically occurs when you are populating a drop-down list (single-select or multi-select). Rather than attaching such a rule to a Form Load event, it might easier and more efficient to simply define a table-based option list as part of the field's Display Properties.

Note Grid dictionaries cannot be used to create a Lookup Condition.



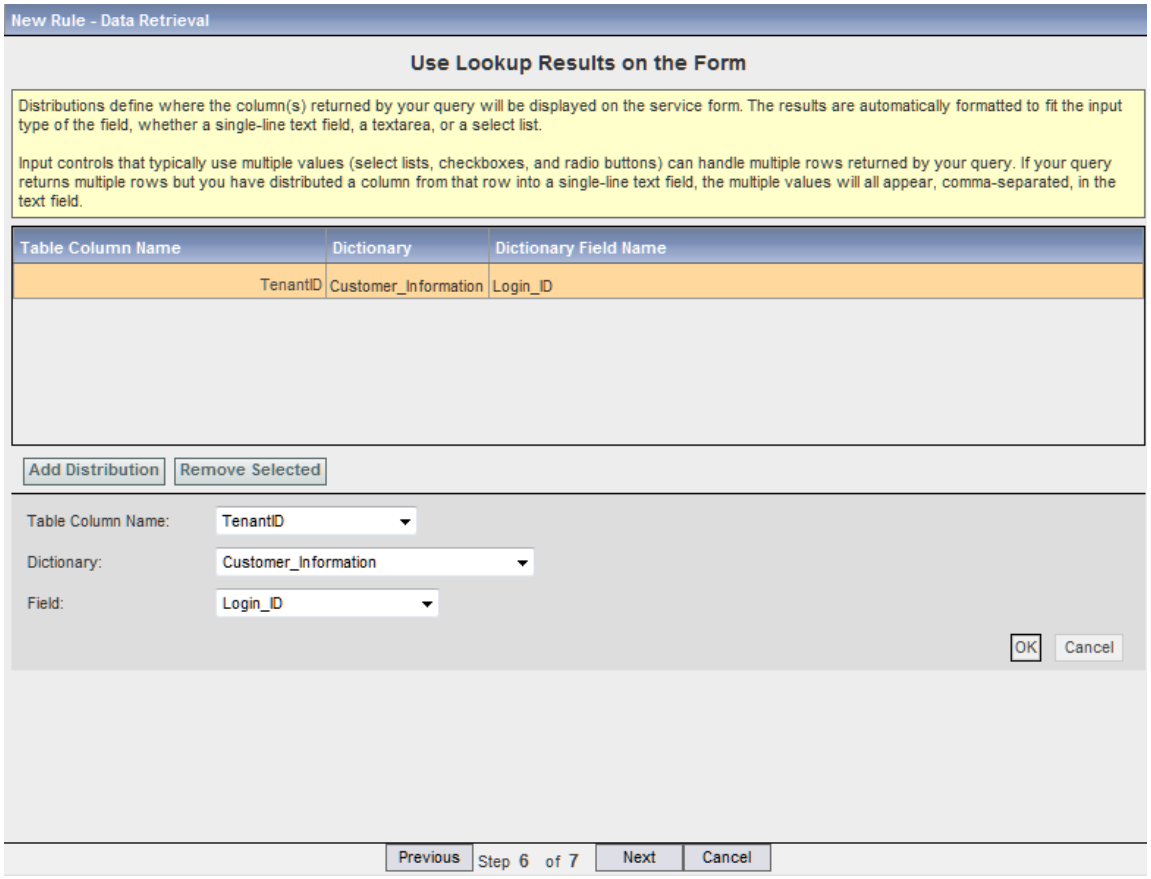
Database Table Lookup – Sort Conditions (Distributing Rules)

If you expect the rule to return more than one row, you may want to sort the data returned so that the results are in an appropriate order. Any number of sort fields can be specified, and a sort direction (ascending or descending) given for each. As you enter each sort condition, click **OK** to add the new field to the Sort Conditions displayed at the top of the page.



Using Lookup Results on the Form (Distributing Rules)

For a Distributing Rule or "Distributing Rule with implicit validation performed on the post-Submit event", at least one distribution target must be defined. A "distribution" defines how the values returned from both table-based lookups and SQL queries are used on the form.



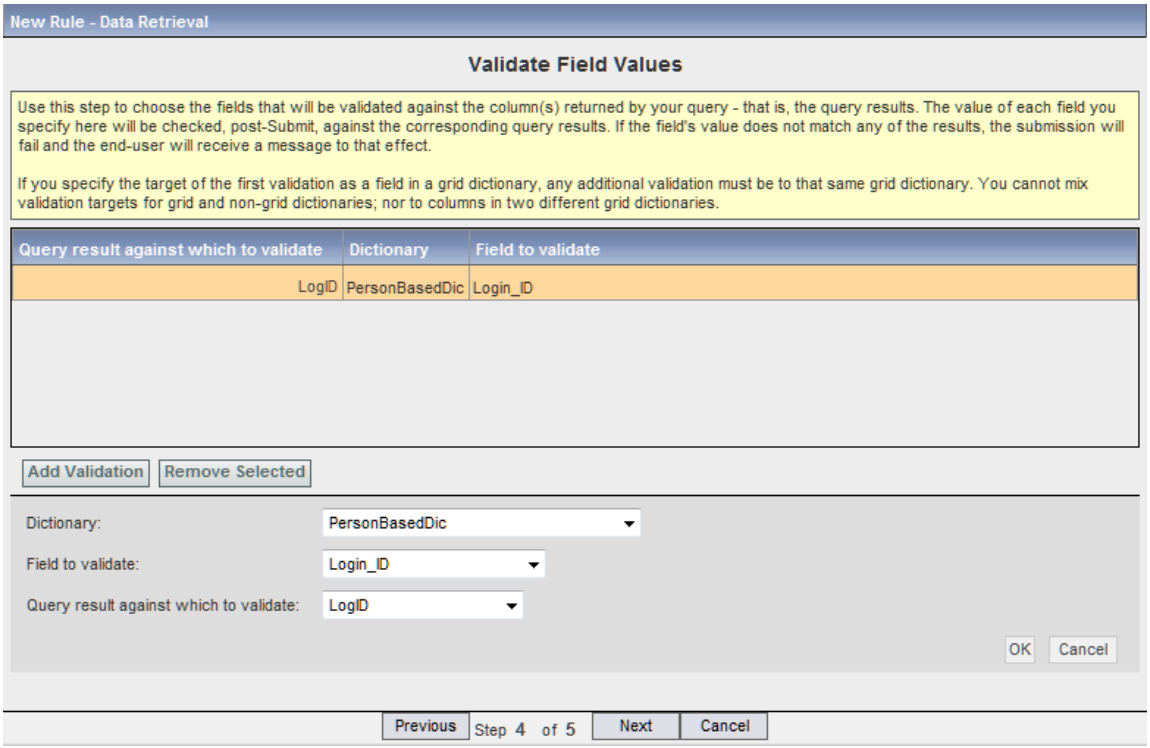
Distributions map column values returned in the query to fields used in the service. Each rule may include one or more distributions. For example, a rule used to populate a drop-down list may have just one distribution (mapping the column to a dictionary field that has the HTML representation of a single- or multi-select element). Alternatively, a rule may have multiple distributions, each mapping from one column to one dictionary field. The target dictionary fields need not be writeable on the form—they can be read-only or hidden.

A rule cannot distribute results to a combination of grid and nongrid dictionaries, nor to two different grid dictionaries. Once you have chosen one grid dictionary field as a distribution target, any additional targets are limited to fields in that same dictionary.

The target of the distribution may be a field on a dictionary that is not included in the current form component. It is the responsibility of the service designer to ensure that any dictionary referenced is included in another form component which, in turn, is included in a service with the current form component.

Validate Field Values (Validating Rule)

For a Validating Rule, at least one validation field must be defined, as shown below.



Use this step to choose the fields that are validated against the column or columns returned by your query—that is, the query results. The value of each field you specify here are checked after the form is submitted, against the corresponding query results. If the field's value does not match any of the results, the submission will fail and the end-user will receive a message to that effect.

You cannot choose fields in a dictionary configured as a grid here, as fields in a grid are rendered as multiple cells in a column.

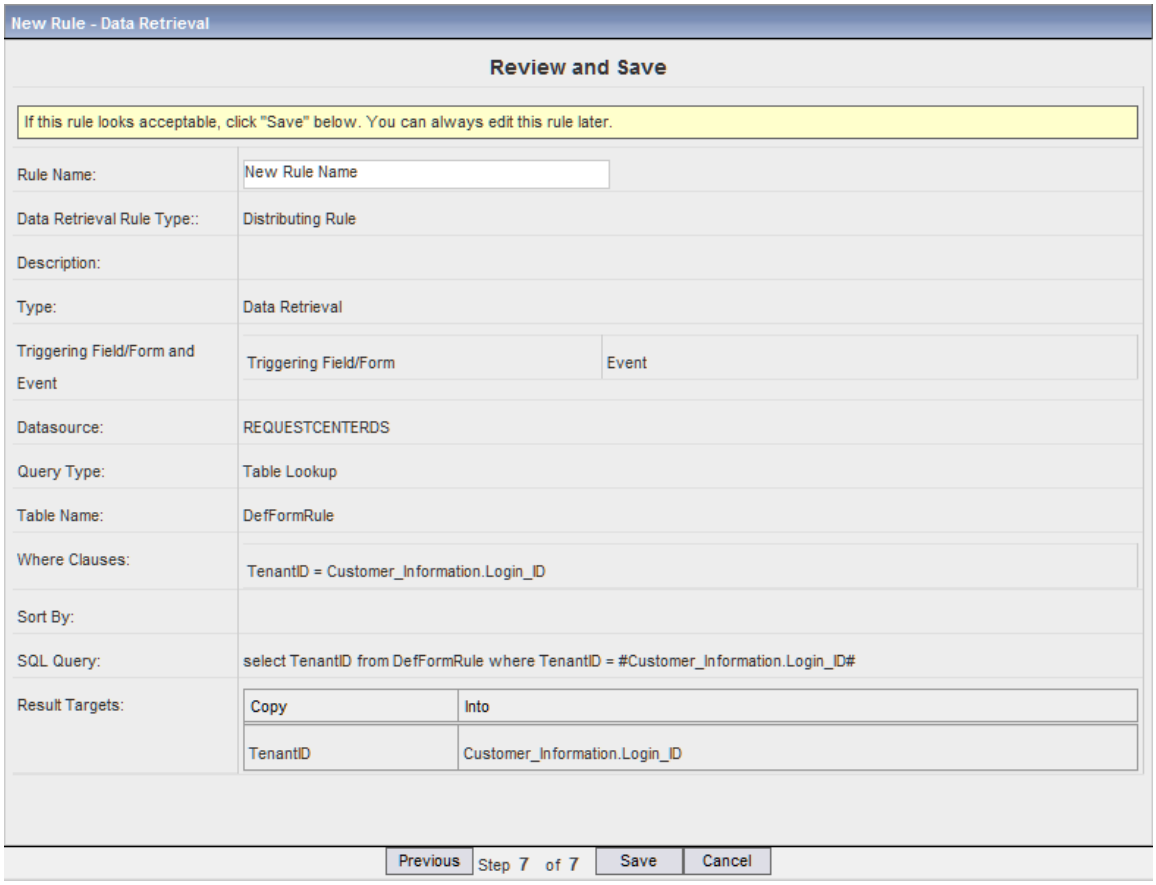
A rule cannot validate to a combination of grid and nongrid dictionaries, nor to two different grid dictionaries. Once you have chosen one grid dictionary field as a validation, any additional validations are limited to fields in that same dictionary.

The validation field may be a field on a dictionary that is not included in the current form component. It is the responsibility of the service designer to ensure that any dictionary referenced is included in another form component which, in turn, is included in a service with the current form component.

Review and Save

The rule definition is displayed on the last page of the wizard. Click **Save** to save the rule, **Cancel** to discard the rule (or changes made in this session of the wizard), or **Previous** to return to a previous page of the wizard. The rule definition may also be displayed by choosing the rule on the Active Form Rules page.

One limitation to note is that radio button and checkbox fields with options populated by the retrieval rules have their field values evaluated and stored only at the time the form data is sent to the server. Hence any form rules that validate or make use of the values in these fields will not work properly.



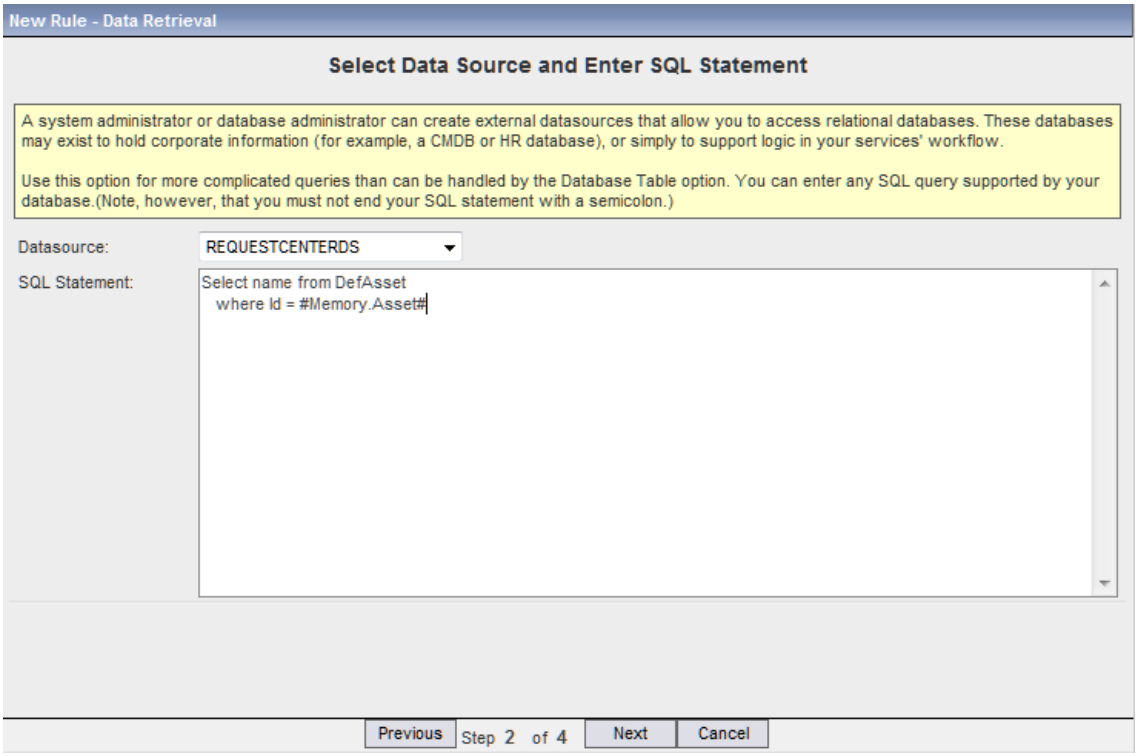
Geek Alert: Data retrieval rules are run on the server, and use bind variables. This is both secure and efficient. You can use a namespace anywhere SQL expects a value, not the name of something.

Enter Your Own SQL Query – Data Source and SQL Statement

For a Query Type of “Enter Your Own SQL Query”, you must specify the data source from which the information is to be retrieved, and write the complete SQL SELECT statement to be executed. To reference the value of a field on the form, the query must include the lightweight namespace that refers to that field.

Note The parser will automatically insert a single quote around the value returned for the namespace at runtime. This is true for all data types except 'datetime'. The value for a datetime data type must match exactly the datetime format expected by the RDBMS. Each RDBMS may have a different configuration for the datetime format.

Note If you choose the Query Type of “Database Table Lookup”, the generated SQL will appear at the end of the wizard in the SQL Query field (the field name changes to Generated SQL Query when saved). This gives you the starting-point from which you can then construct your own SQL, via the second type (that is, you can copy the generated SQL, then go back to Step 1 of the wizard, choose “Enter Your Own SQL Query” and paste the generated SQL into the SQL Statement field.



In the simplest case, a database table retrieval can replicate the results of a direct SQL query which retrieves data from a single table by a set of query criteria (where clauses) which are AND'ed together.

Database Table Retrieval Type	Direct SQL Query Retrieval Type
Table Name: DefAsset **Where Clause**: **Dict.Field**:Id = Memory.AssetId **Literal**:activeFlag = 1 **Sort Field(s)**:(None)	Select name from DefAsset where Id = #Memory.Asset# and activeFlag = 1
Distribution: Name: Memory.AssetDescription	**Distribution:** Name: Memory.AssetDescription

However, more complex SQL statements can be written using the SQL Query option. Any SQL statement supported by the driver used for the specified datasource may be specified. For example, the SELECT statement could include:

- Data from multiple tables, joined together.
- Expressions concatenating multiple text fields or performing arithmetic operations. An expression must be aliased in order to be included in a distribution.
- Database functions, both those provided by the database vendor and user-defined. (Procedures are not supported.)
- Complex relationships or operators in the WHERE clause.

Limitations:

- SQL statements are validated by the SQL processor. Symbols such as @ and # have special meaning in form data processing and are not supported in certain contexts.

- CASE-WHEN syntax is not supported for Microsoft SQL Server at this time.

A possible reason to write your own SQL query is to perform a "LIKE" comparison – for example, retrieve all servers whose name includes a search string entered by the user into the dictionary field referenced by the lightweight namespace #Hardware.ServerName#. The LIKE operator typically uses the wildcard (%) as a prefix or suffix to the search string. The SQL statement must be entered with the wildcard as a separately quoted item:

```
select server_name from CompanyServers
  where server_name like '%' || #Hardware.ServerName# || '%'
```

Unless your DBA or IT Analyst has written and debugged SQL queries for you, you will need a SQL development environment to do this. Although the Data Retrieval Wizard validates SQL, it doesn't provide any helpful hints in case the SQL is invalid—that is what you need the other environment for. See the "Prerequisites to Creating Data Retrieval Rules" section on page 2-33 for more details.

Example: Laptop Selection

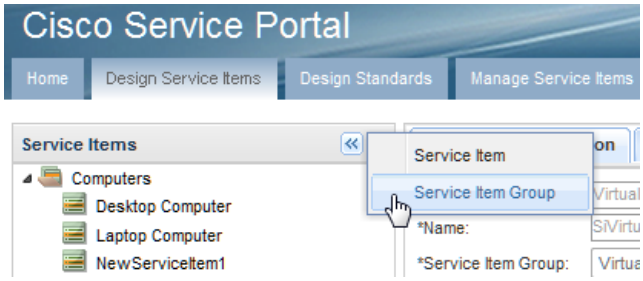
The Laptop Selection example uses a grid service item dictionary and a data retrieval rule to demonstrate automatic retrieval of service item data when a field is chosen from a drop-down menu in a grid. The data retrieval rule also includes a Lookup Condition where a data instance is excluded from selection.

This example walks you through the steps to do the following:

1. Create New Service Item Group and Service Item
2. Create New Dictionary Group and Grid Dictionary
3. Create New Form Group and Form
4. Add Grid Dictionary to Form and Edit Display Properties
5. Create New Data Retrieval Rule
6. Create New Service Group and New Service
7. Add Form to Service and View Service in My Services

Step 1 Choose **Service Item Manager > Design Service Items**. The Design Service Items page appears.

Step 2 Click the plus sign (+), and then choose **Service Item Group**, as shown below.

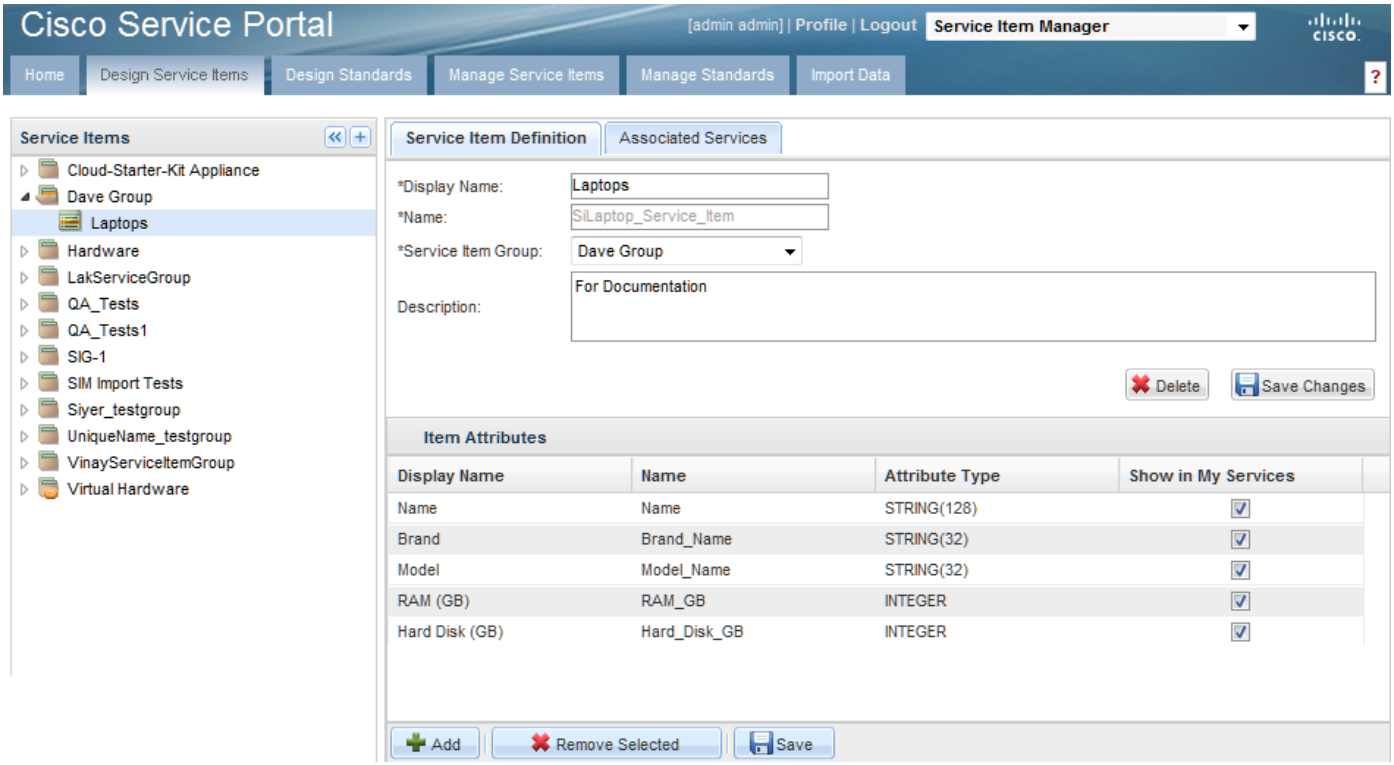


Step 3 Enter a group name (in this example, **Dave Group**) and, optionally, a description in the corresponding fields of the resultant Add New Service Item Group popup window, and then click **Add**. The new service item group appears in the Service Items panel on the left.

Step 4 Click the plus sign (+) again, and then choose **Service Item**.

Step 5 In the Add New Service Item popup window, enter **Laptops** in the Display Name field and **Laptop_Service_Item** in the Name field. Choose the service item group you created, **Dave Group**, in the Group drop-down menu, and then click **Add**. The new service item appears in the Service Items panel on the left under Dave Group.

Step 6 On the bottom of the page, click **Add** five times to add the five attributes to the Laptops Service Item, as shown below. Each attribute, or row, is a service item instance. The Display Name for each instance will become a column header for the grid.



Step 7 On the bottom of the page, click **Save** after you have added all the attributes.

Step 8 Click the **Dictionaries** component of the Service Designer module.

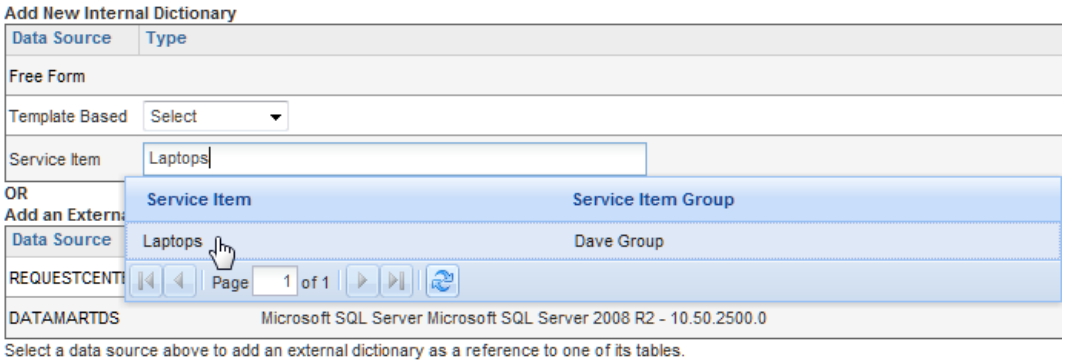
Step 9 Choose **New> New Dictionary Group**, as shown below.

Step 10 In the Title field of the resultant New Dictionary Group window, enter **Dave Dictionary Group** for this example. Enter a description for the group, if desired.

Step 11 Click **Add This Dictionary Group**. The new group appears in the Dictionaries tree to the left.

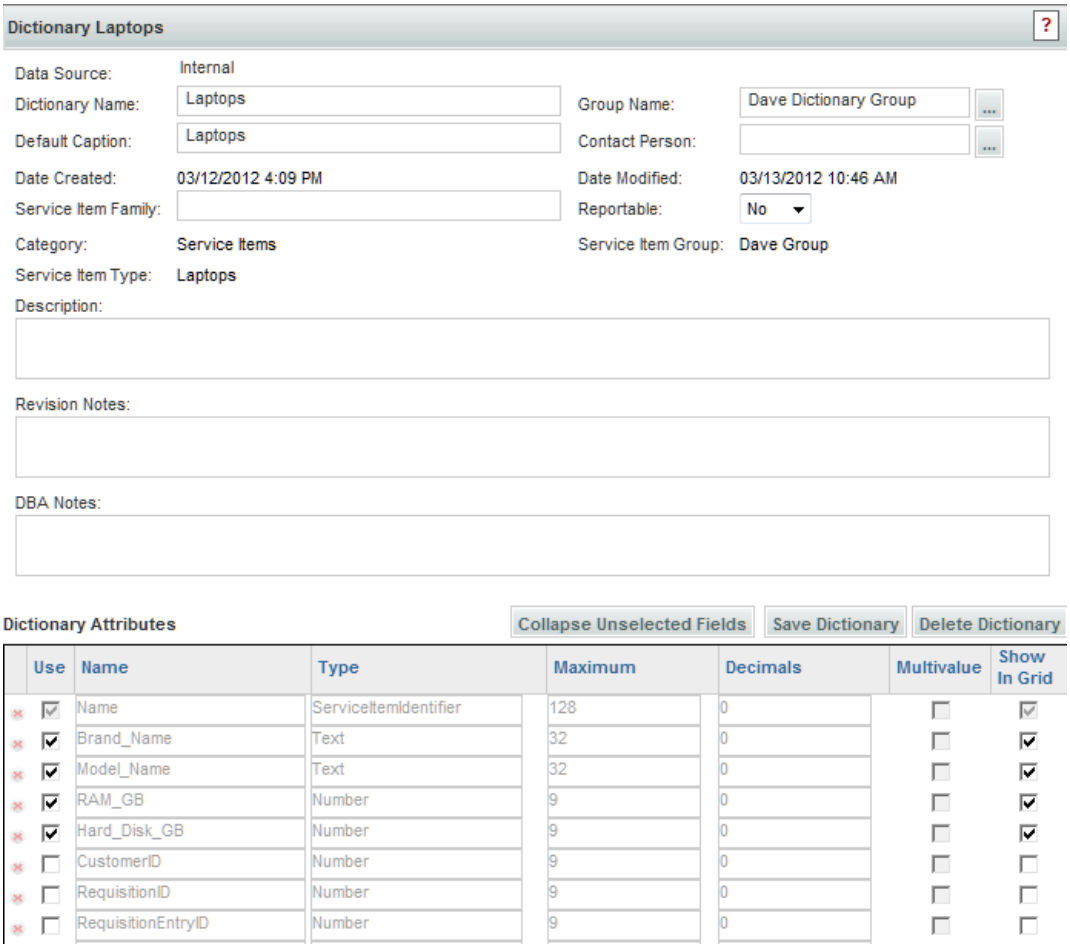
Step 12 Choose **New > New Dictionary**. The New Dictionary window appears.

Step 13 In the Service Item field of the Add New Internal Dictionary section, enter **Laptops** to search for the Laptops service item you created, then click **Laptops** from the popup window that appears, as shown below.



Step 14 In the resultant Dictionary window, configure the dictionary fields as shown in the example below.

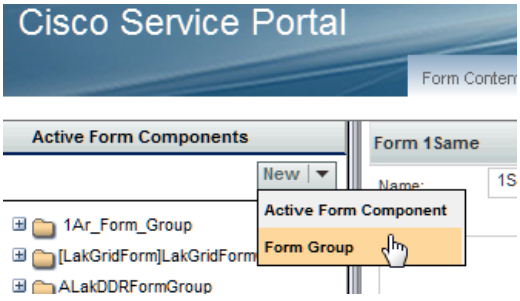
Enter **Laptops** for the Dictionary Name and Default Caption fields. The Default Caption will display as the title for the grid. Choose the dictionary group you just created, **Dave Dictionary Group**, from the Group Name drop-down menu. In the Use and "Show in Grid" columns, check only the five instances of the Laptops service item you created. By checking Use and "Show in Grid", the instances will appear as columns in the grid. The Name field is already checked by default and cannot be changed.



Step 15 Click **Save Dictionary**.

Step 16 Click the **Active Forms Components** component of the Service Designer module.

Step 17 Choose **New> Form Group**, as shown below.



Step 18 In the Name field of the resultant Form Group window, enter **Dave Form Group** for this example. Enter a description for the group, if desired.

Step 19 Click **Save**. The new form group appears in the Active Form Components tree to the left.

Step 20 Choose **New > Active Form Component**.

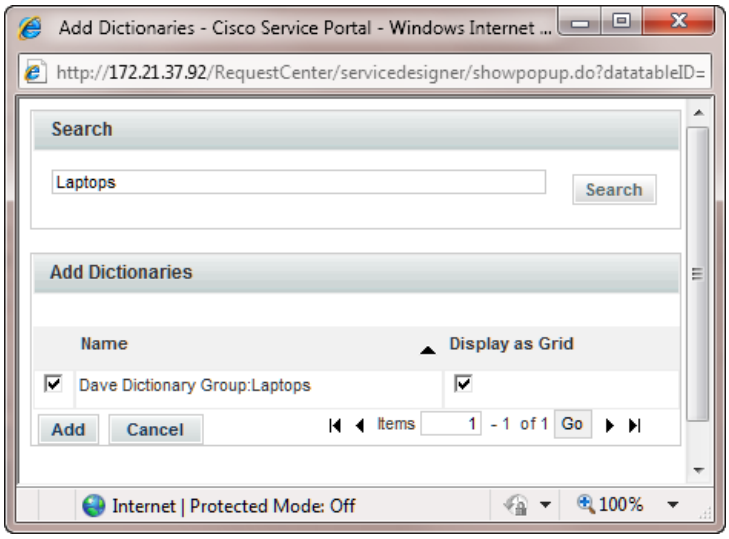
Step 21 In the Name field of the resultant New Active Form Component window, enter **Laptop Selection** for this example. Enter a description for the form, if desired. Click the browse button (...) to the right of the Form Group field. In the resultant Search popup window, enter **Dave Form Group** to search for the Form Group you created, and then click **Search**. With its radio button selected, click **Add** to enter the Dave Form Group into the From Group field.

Step 22 Click **Save Form**.

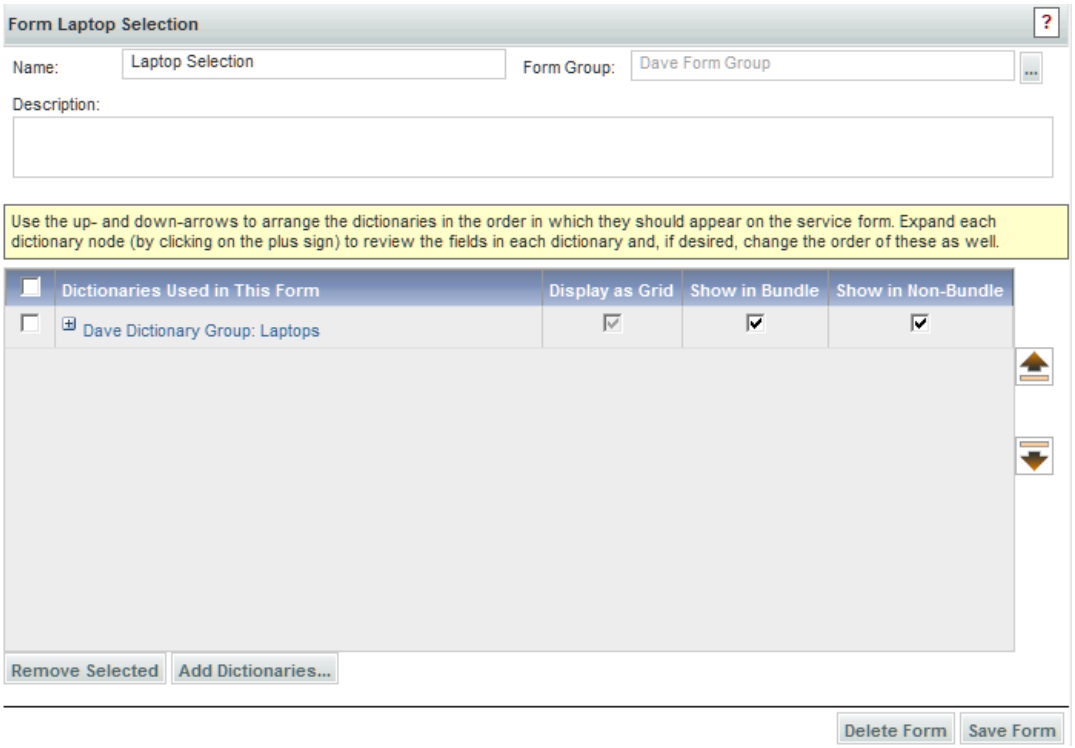
Step 23 Click **Add Dictionaries**. The Add Dictionaries popup window appears.

Step 24 In the Search field, enter **Laptops** to search for the Laptops dictionary you created.

Step 25 For the resultant Laptop dictionary, check the Name and "Display as Grid" columns, as shown below. Checking "Display as Grid" makes the fields of the Laptops dictionary that were checked to "Show in Grid" display as columns in the grid.



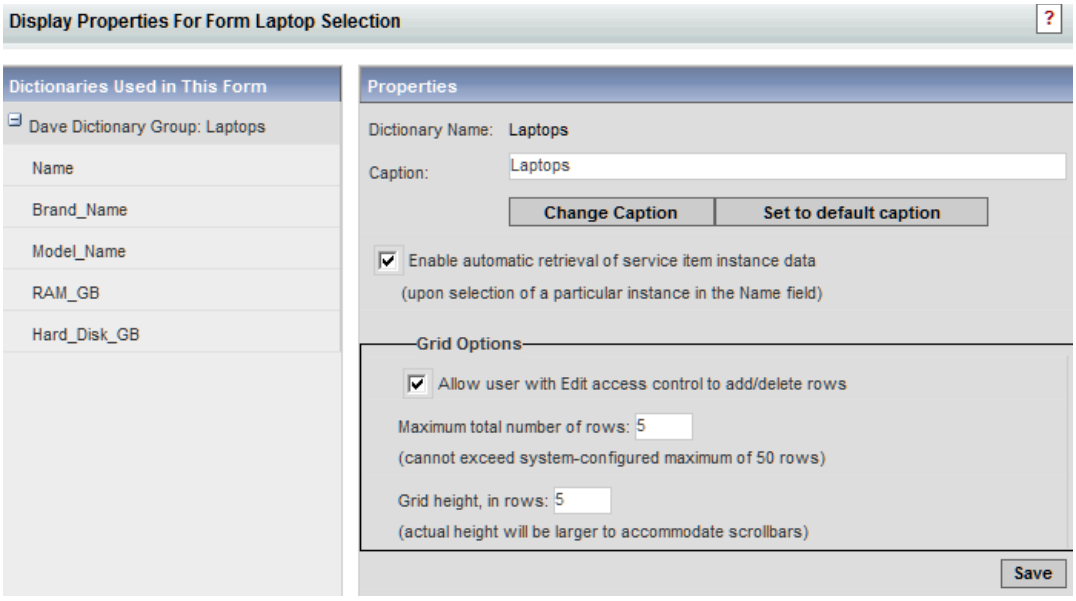
Step 26 Click **Add** to add the Laptops dictionary to the form. The Form Content tab of the Laptop Selection form appears with the Laptops dictionary checked for "Display as Grid". Note that it is dimmed here and cannot be changed. The only way to change it would be to remove it, and then add it again without checking "Display as Grid".



Step 27 Click **Save Form** to save the form.

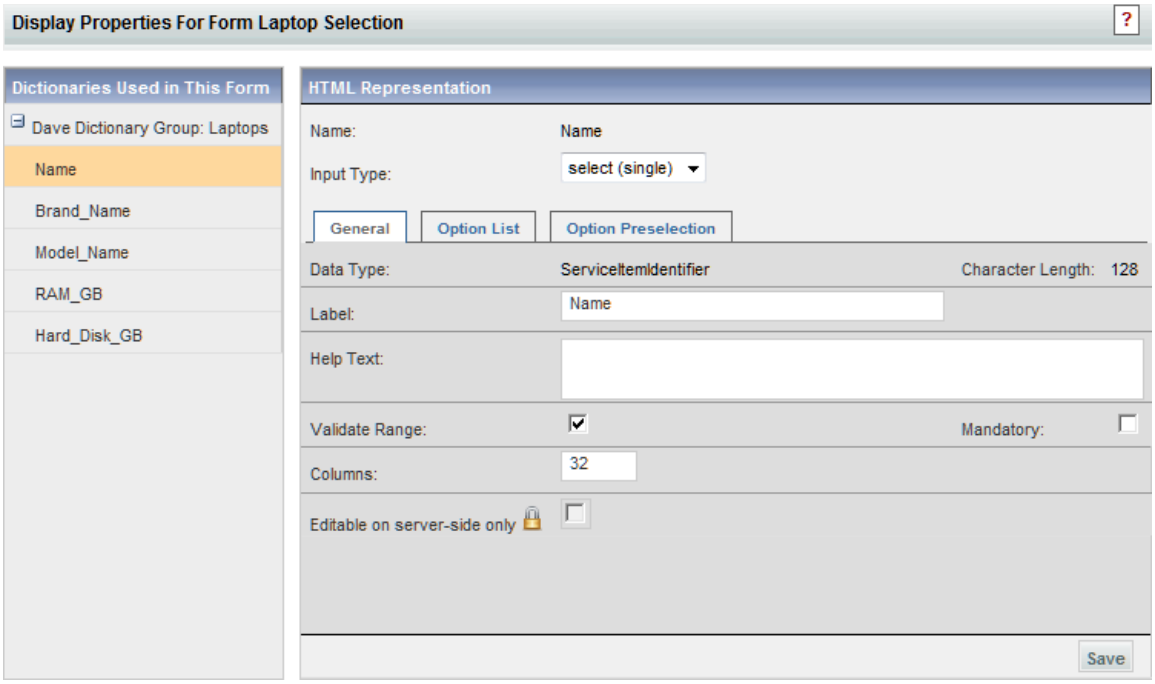
Step 28 Click the **Display Properties** tab for the Laptop Selection form.

Step 29 Check **Enable automatic retrieval of service item instance data**, as shown below. This option is disabled by default. This option enables the ability to automatically retrieve and prefill data about an existing service item instance. For this example, leave the Grid Options to their default selections and values. In particular, you want the user to have Edit access control to be able to add grid rows.



Step 30 In the lower right, click **Save**.

Step 31 Below the Laptops dictionary in the left pane, click the **Name** field to see its HTML Representation window. In the Input Type field, choose "**select (single)**" from the drop-down menu. An Input Type of "Select Single" renders the Name field as a drop-down menu from which the user can choose only a single value. Also, change the Columns value to **32** to make the Name column width smaller.



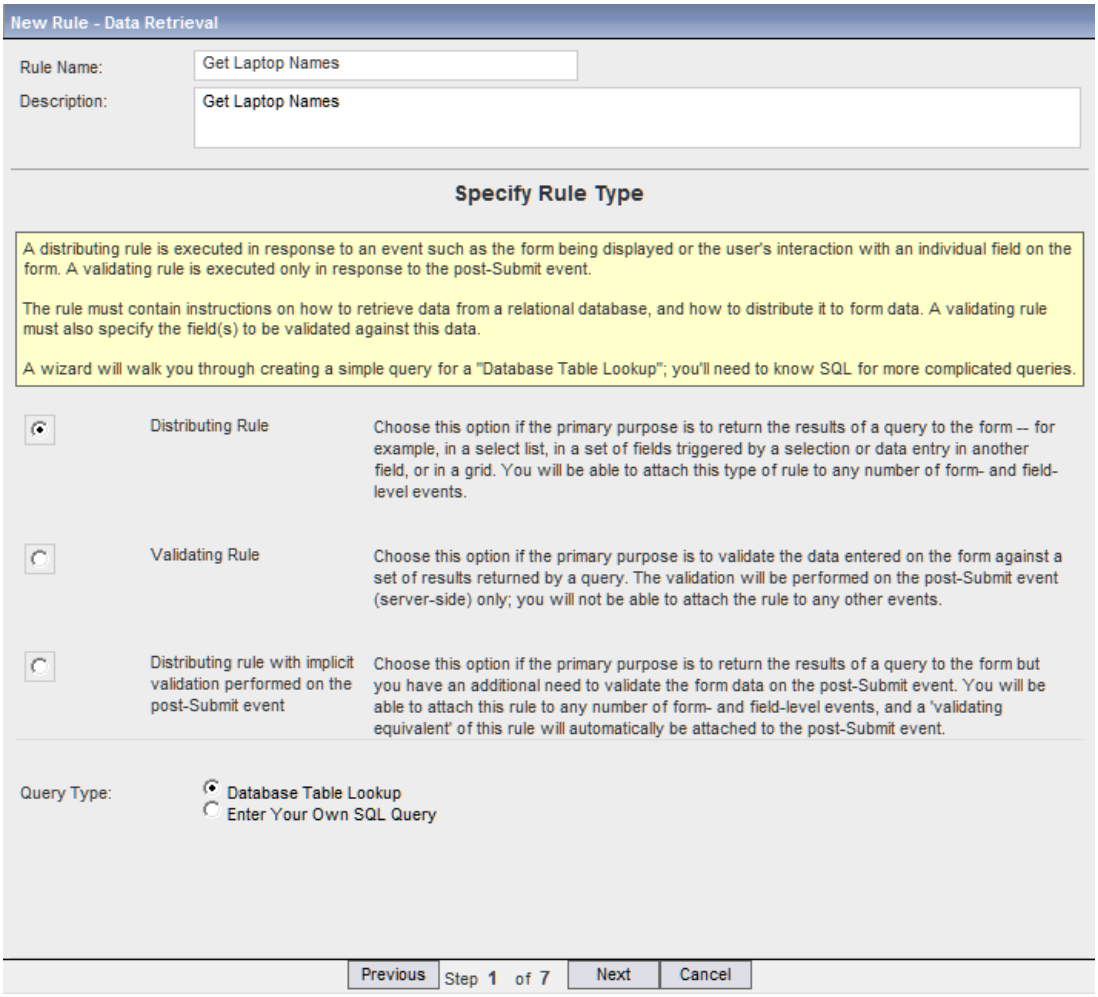
Step 32 In the lower right, click **Save**.

Step 33 Click each of the remaining fields of the Laptop dictionary and change the Label field to remove the underscore character and add parentheses around "GB". Also, change the Columns value for the RAM and Hard Disk fields to **32** to make all the columns the same width. In the lower right, click **Save** after each field change.

Step 34 Click the **Active Form Rules** tab for the Laptop Selection form.

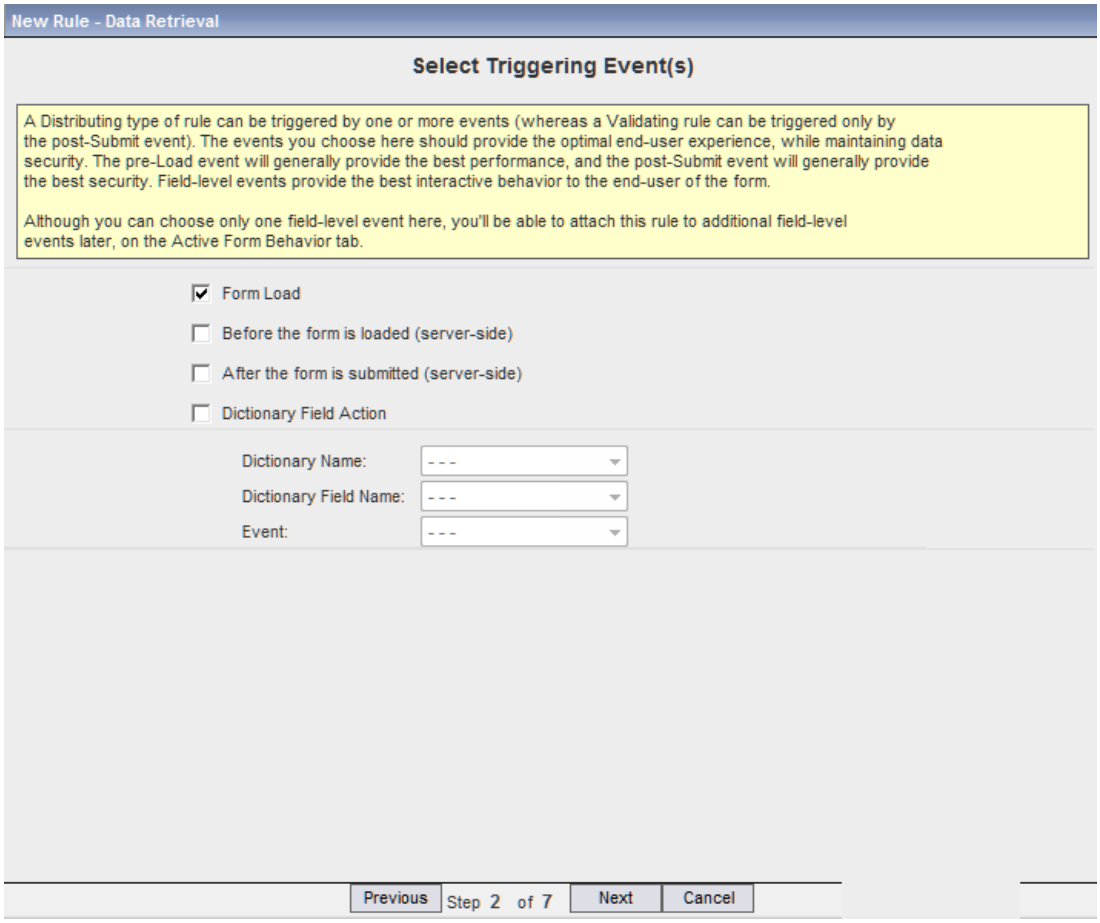
Step 35 Choose **New Rule > New Data Retrieval Rule** on the bottom left of the window, as shown below.

Step 36 Step 1 of the Data Retrieval Rule Wizard appears. Enter **Get Laptop Names** in the Rule Name and Description fields. Set the Rule Type set to **Distributing Rule** and leave the Query Type set to **Database Table Lookup**.



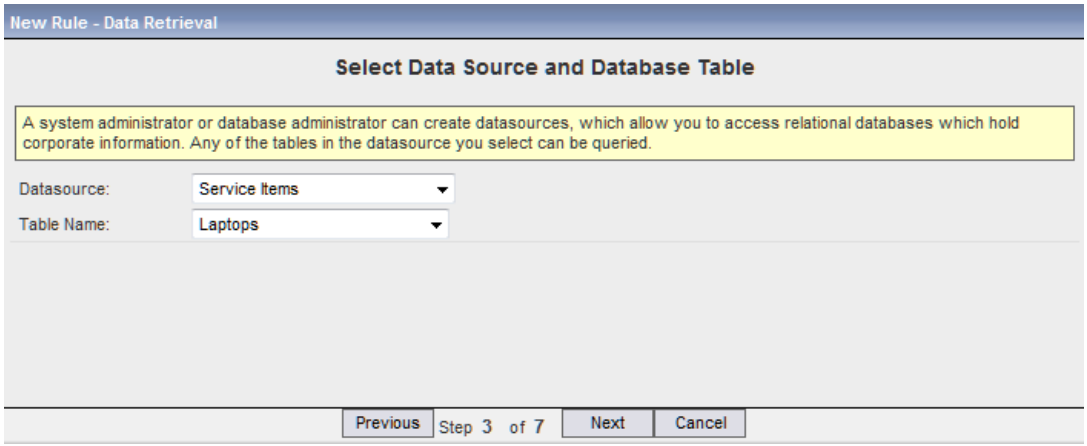
Step 37 Click **Next** to continue.

Step 38 Step 2 of the Data Retrieval Rule Wizard appears. Check **Form Load** as the triggering event, as shown below.



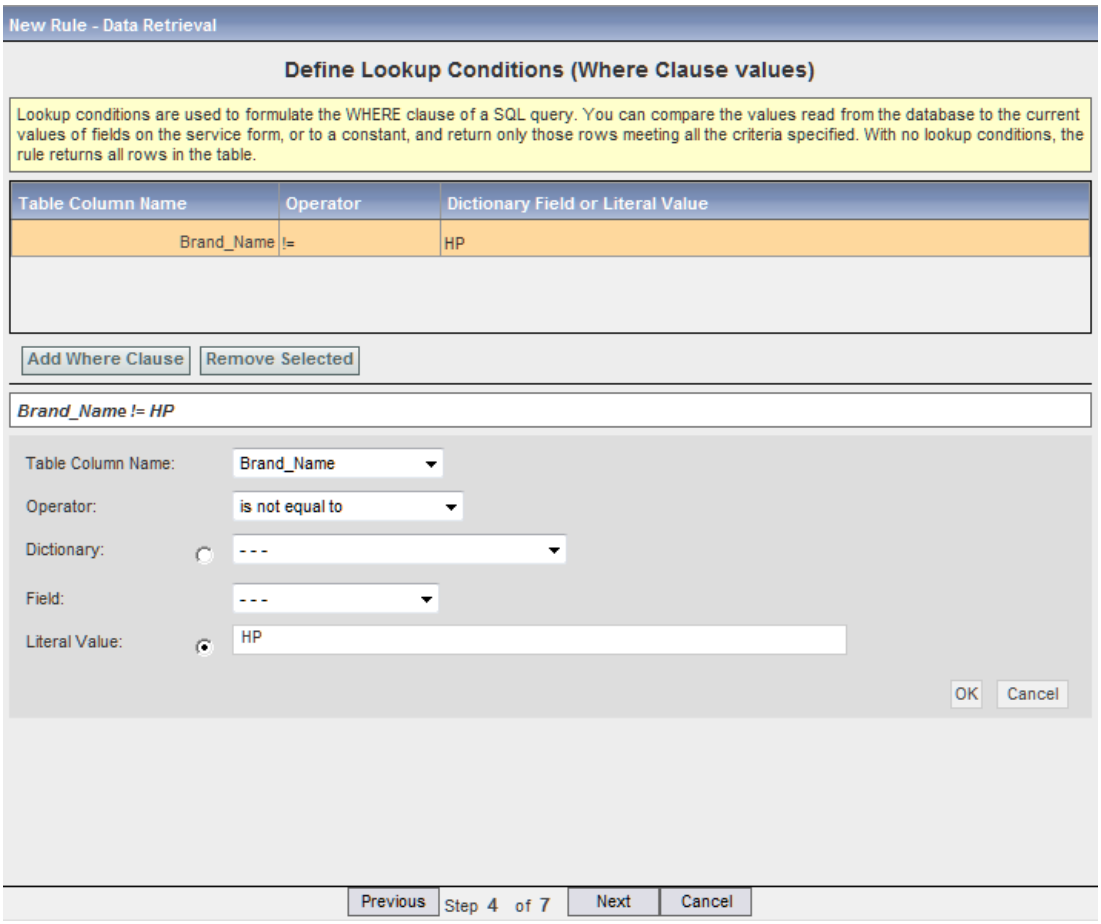
Step 39 Click **Next** to continue.

Step 40 Step 3 of the Data Retrieval Rule Wizard appears. Choose **Service Items** from the Datasource drop-down menu, and **Laptops** from the Table Name drop-down menu, as shown below. Laptops is the service item you previously created.



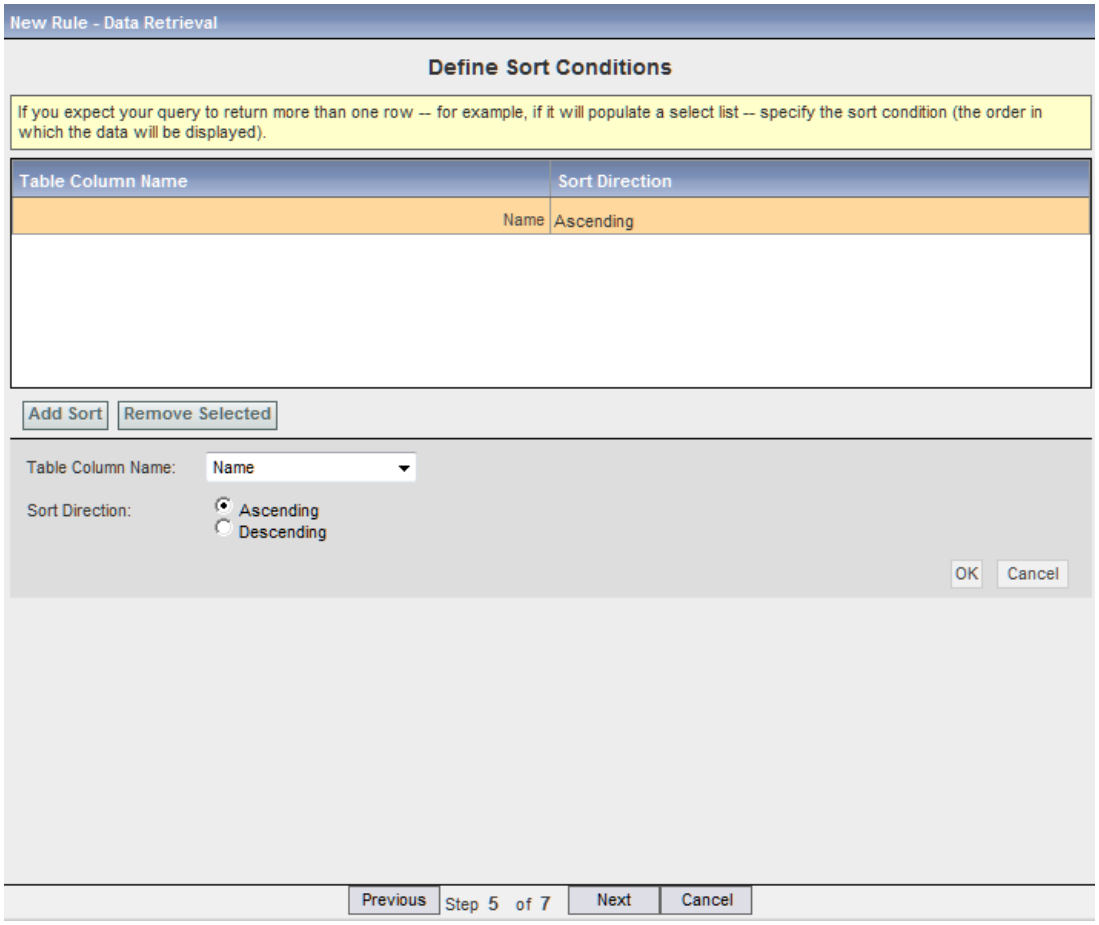
Step 41 Click **Next** to continue.

Step 42 Step 4 of the Data Retrieval Rule Wizard appears. To create a Lookup Condition where data instances of "HP" in the Brand_Name field are excluded from selection, choose **Brand_Name** from the Table Column Name drop-down menu and "**is not equal to**" in the Operator drop-down menu, and enter **HP** in the Literal Value field (enabling its radio button), as shown below. Click **OK**.



Step 43 Click **Next** to continue.

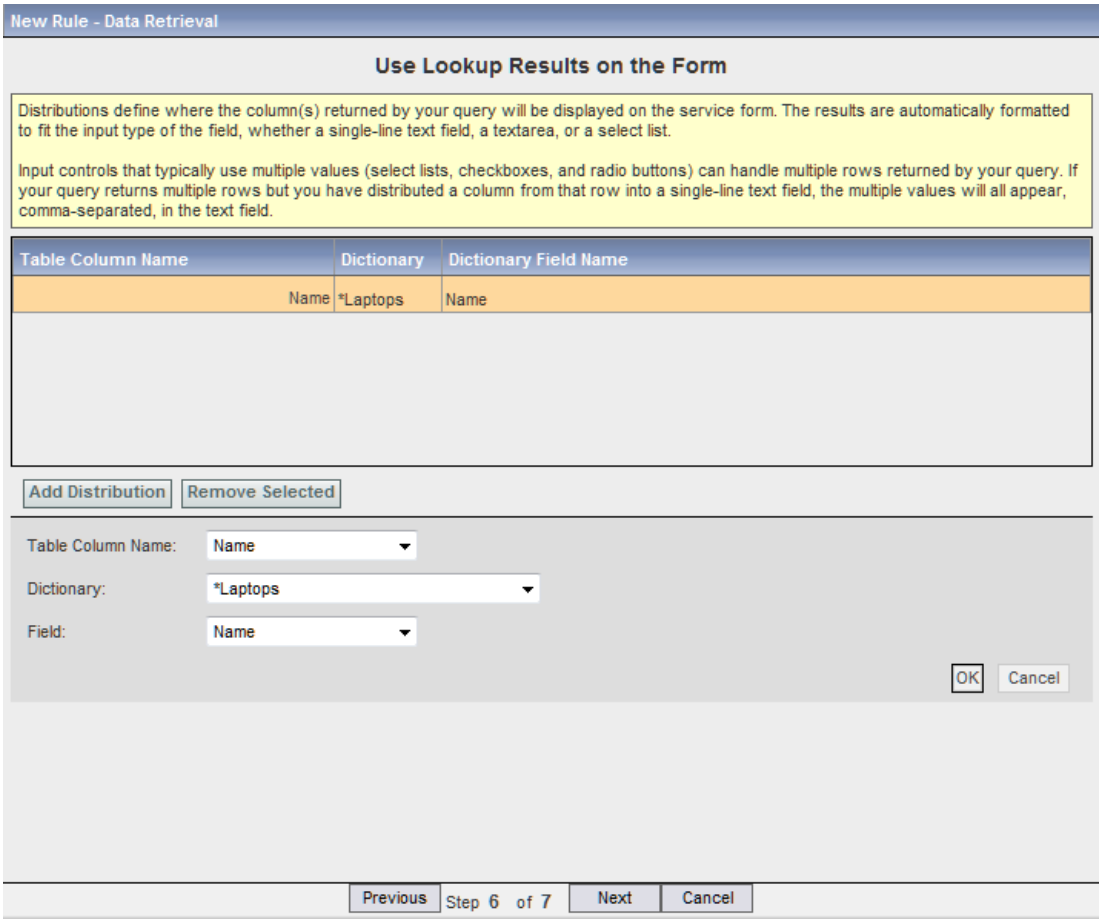
Step 44 Step 5 of the Data Retrieval Rule Wizard appears. In the Table Column Name field choose **Name** from the drop-down menu, and click the **Ascending** radio button for the Sort Direction. Click **OK** to define the sort condition.



Step 45 Click **Next** to continue.

Step 46 Step 6 of the Data Retrieval Rule Wizard appears. To map the Name column of the Laptops service item to the Name field of the Laptops dictionary, choose them from the three drop-down menus as shown below and then click **OK**.

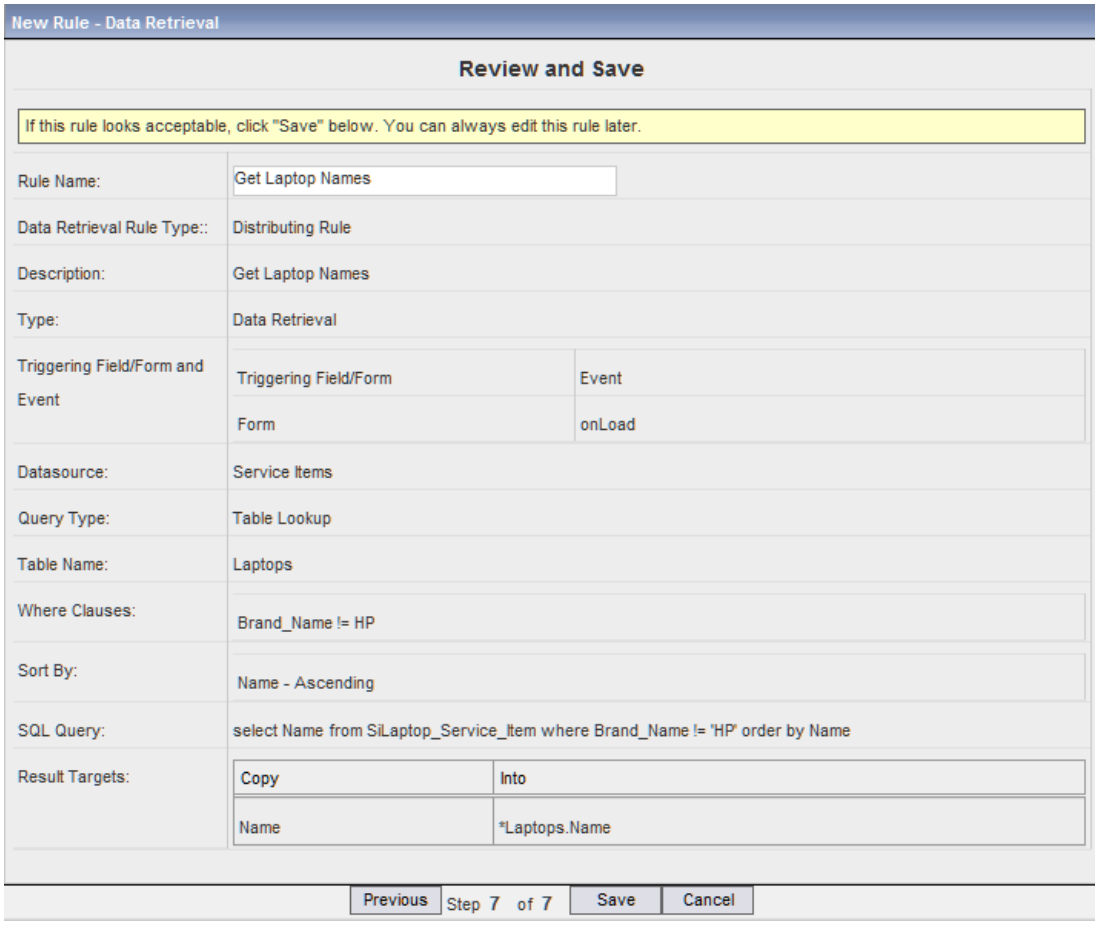
Note The "*" next to Laptops indicates that Laptops is a grid dictionary.



Step 47 Click **Next** to continue.

Step 48 Step 7 of the Data Retrieval Rule Wizard appears, as shown below.

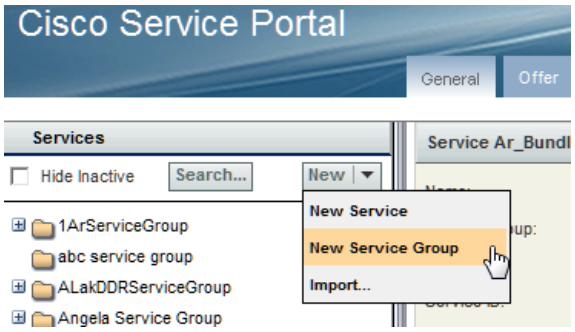
This final page of the wizard allows you to review and then save your rule. Check to make sure you have entered everything correctly. Click **Previous** if you need to go back to a Step and make changes. You can always go back and edit the rule after you have saved it.



Step 49 Click **Save** to complete the Data Retrieval Rule Wizard.

Step 50 Click the **Services** component of the Service Designer module.

Step 51 Choose **New> New Service Group**, as shown below.



Step 52 In the Name field of the resultant New Service Group window, enter **Dave Service Group** for this example. Choose **Site Administration** from the Service Team drop-down menu. Enter a description for the group, if desired.

Step 53 Click **Add This Service Group**. The new group appears in the Services tree to the left.

Step 54 Choose **New > New Service**.

Step 55 In the Name field of the resultant New Service window, enter **Laptop Upgrade** for this example. Enter "**Upgrade one or more laptops.**" for the description. Choose the service group you just created, **Dave Service Group**, from the Service Group drop-down menu.

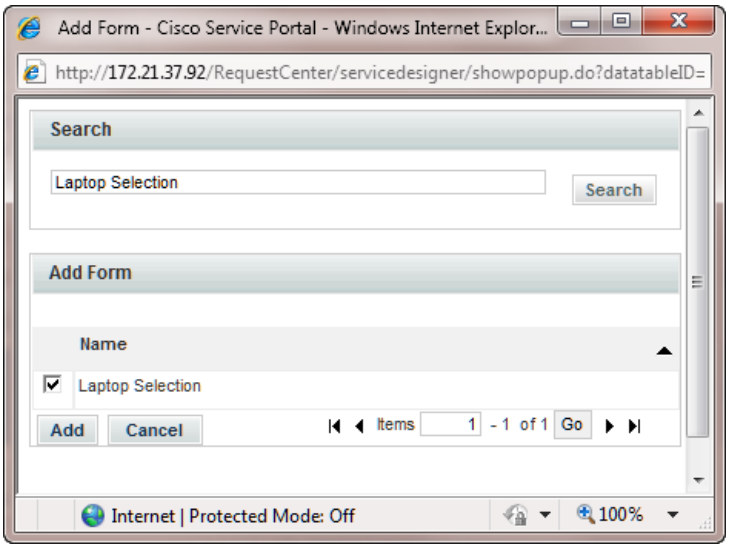
Step 56 Click **Add This Service**. The General Tab of the Laptop Upgrade service appears.

Step 57 Click the **Form** tab of the Laptop Upgrade service.

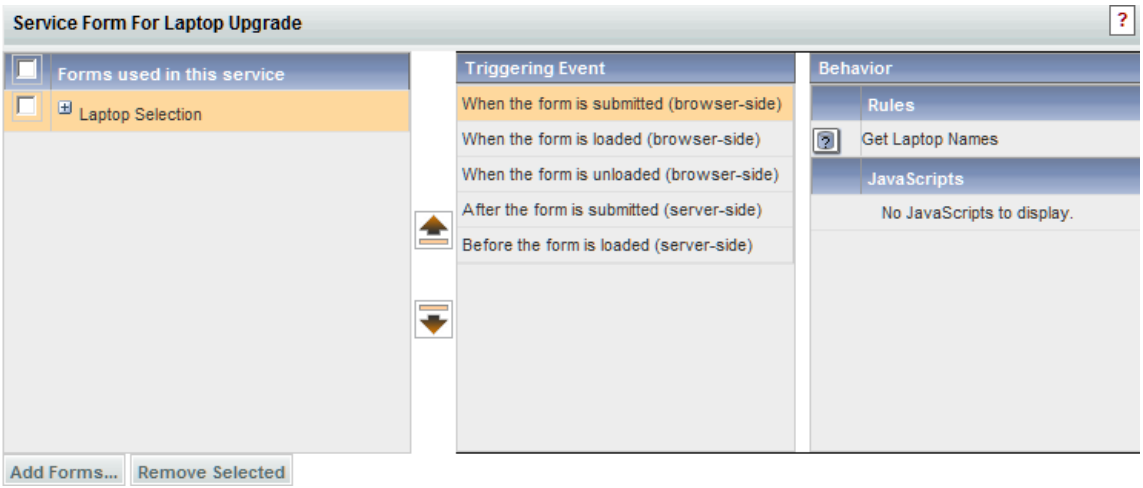
Step 58 Click **Add Forms** in the bottom left of the window. The Add Form popup window appears.

Step 59 In the Search field, enter **Laptop Selection** to search for the Laptop Selection form you previously created.

Step 60 Check the Laptop Selection form and then click **Add**, as shown below.

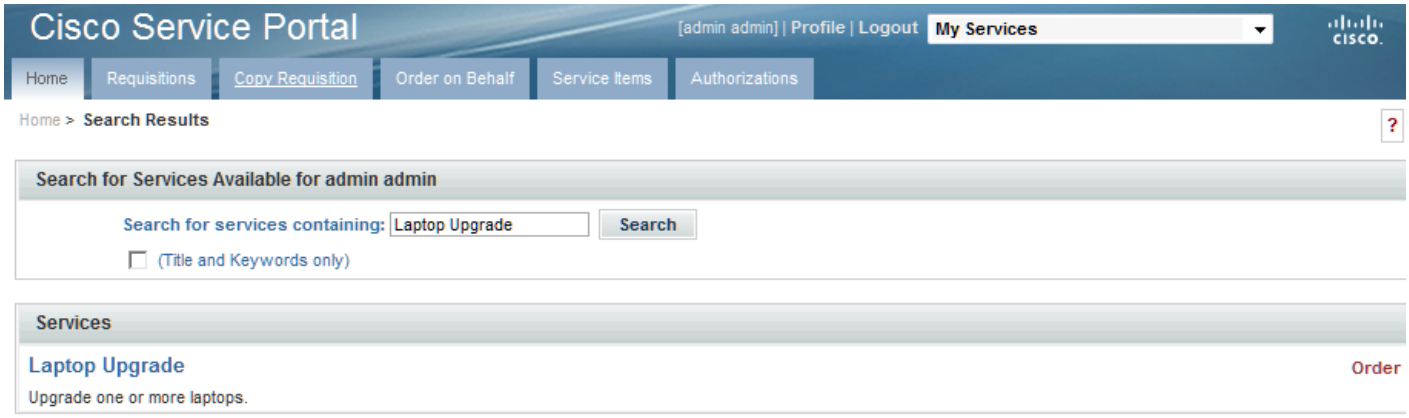


Step 61 The Laptop Selection form is added to the Laptop Upgrade service, as shown below.



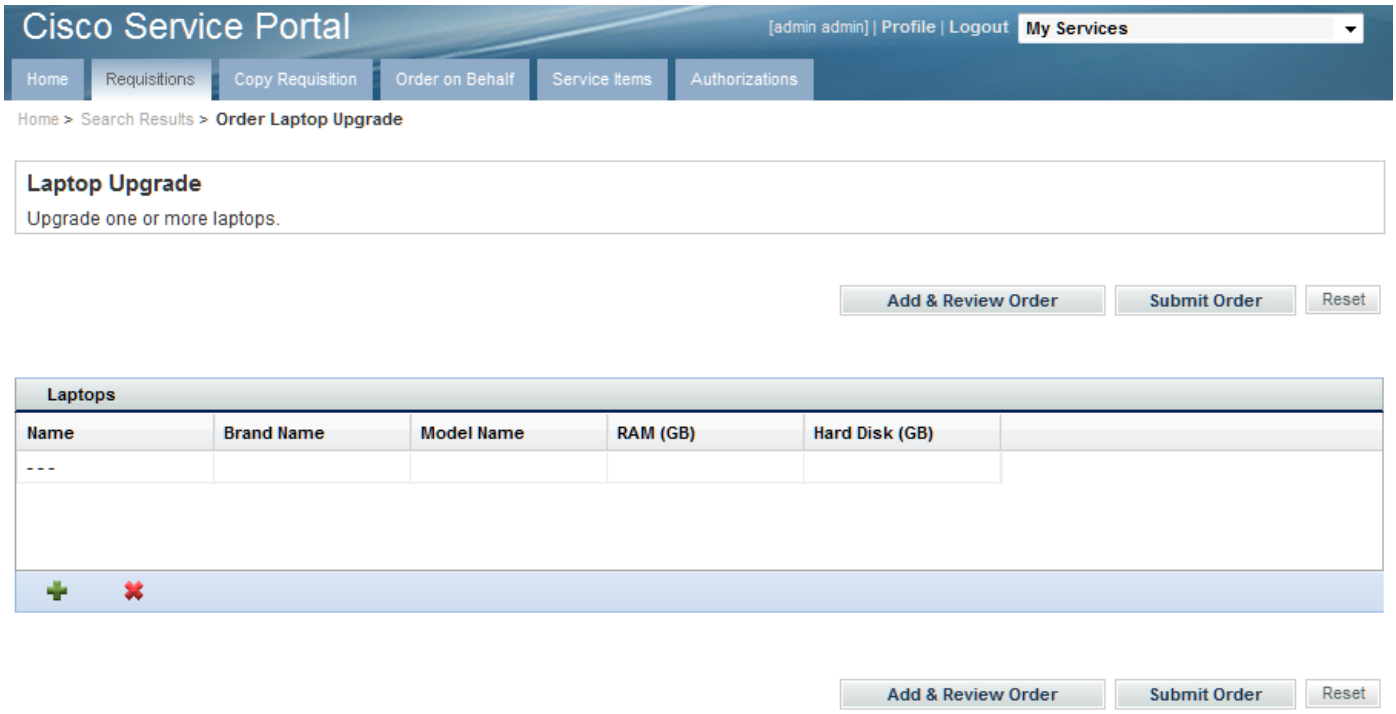
Step 62 Now, let's see what the form looks like to the user. Choose the **My Services** module of Service Portal.

Step 63 Enter **Laptop Upgrade** in the "Search for Services containing:" field and then click **Search**. The Laptop Upgrade service appears, as shown below.



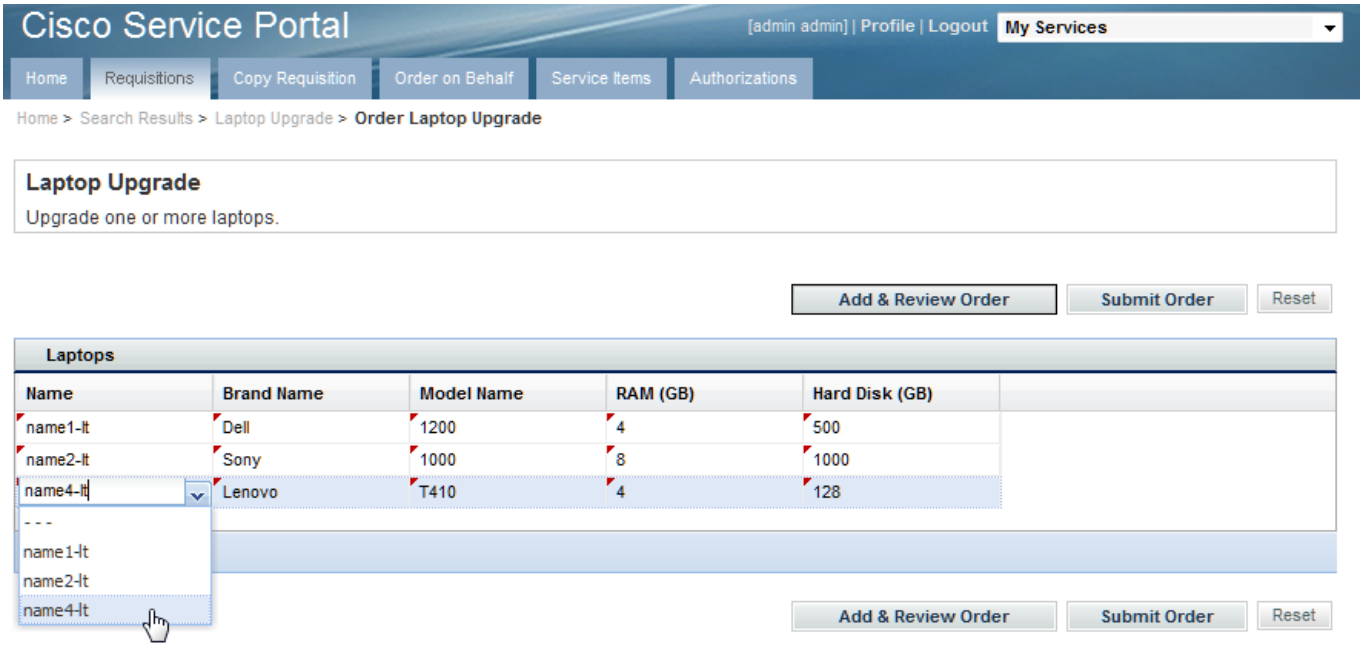
Step 64 Click **Order** to see the grid you created, as shown below.

The grid appears with the header of "Laptops" taken from the Default Name of the dictionary. The column header names are taken from the HTML Representation Label field of the dictionary fields.



Step 65 The Name field appears as a drop-down menu from which you can only choose a single value because you chose "select (single)" as the Input Type for the Name field. Click in the first row of the Name column and choose a name from the drop-down menu—the rest of the fields are automatically filled in. This happens because you checked "Enable automatic retrieval of service item instance data" for the form dictionary and mapped the Name column of the Laptops service item to the Name field of the Laptops dictionary.

Step 66 Click the **Add** row icon ( ) and enter the other two rows of service item data you created. Note that "name3.lt" does not appear in the Name drop-down menu because, in Step 3 of the Data Retrieval Rule, a Lookup Condition was created where data instances of "HP" in the Brand_Name field are excluded from selection.



Example: View All Laptops

The View All Laptops example demonstrates the use of a data retrieval rule to populate all service item instances on a grid, and a conditional rule to make the dictionary read-only.

Note You must work through the previous Laptop Selection example before the View All Laptops example because the View All Laptops example uses the same grid service item dictionary used in the Laptop Selection example (see the "Example: Laptop Selection" section on page 2-43).

This example walks you through the steps to do the following:

1. Create New Form
2. Add Grid Dictionary to Form and Edit Display Properties
3. Create New Data Retrieval Rule
4. Create New Conditional Rule
5. Create New Service
6. Add Form to Service and View Service in My Services

Step 1 Choose **New > Active Form Component**.

Step 2 In the Name field of the resultant New Active Form Component window, enter **View All Laptops** for this example. Enter a description for the form, if desired. Click the browse button (...) to the right of the Form Group field. In the resultant Search popup window, enter **Dave Form Group** and then click **Search**. With its radio button selected, click **Add** to enter the Dave Form Group into the From Group field.

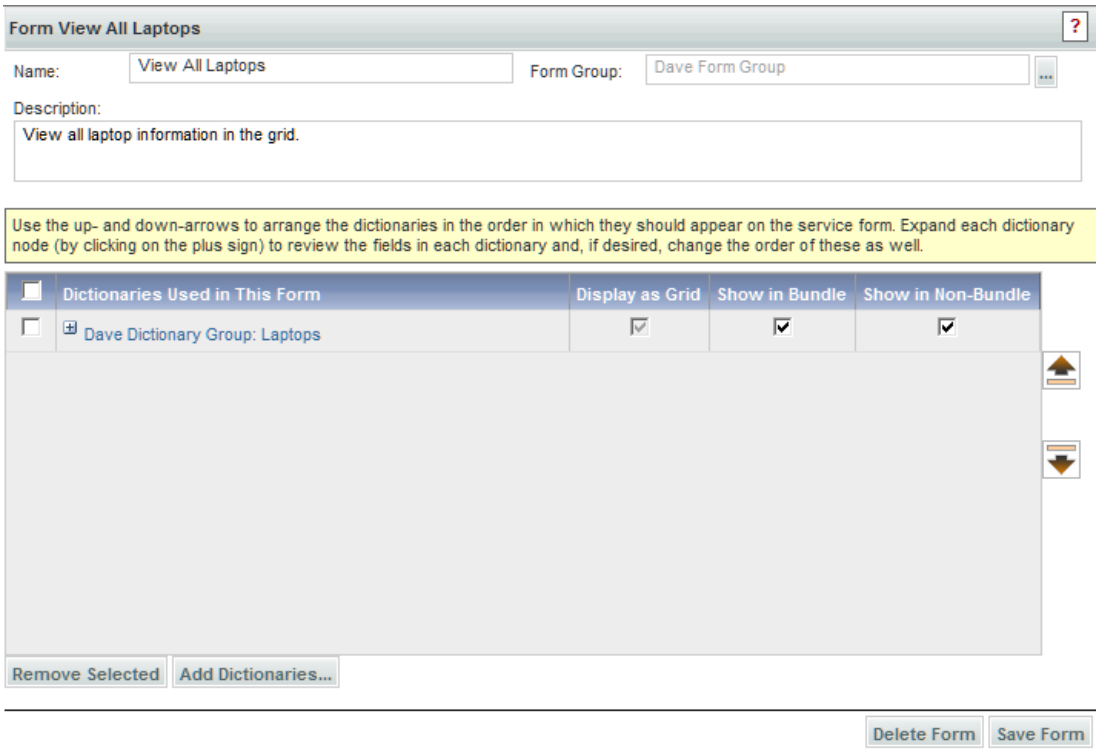
Step 3 Click **Save Form**.

Step 4 Click **Add Dictionaries**. The Add Dictionaries popup window appears.

Step 5 In the Search field, enter **Laptops** to search for the Laptops dictionary.

Step 6 For the resultant Laptop dictionary, check the Name and "Display as Grid" columns. Checking "Display as Grid" makes the fields of the Laptops dictionary that were checked to "Show in Grid" display as columns in the grid.

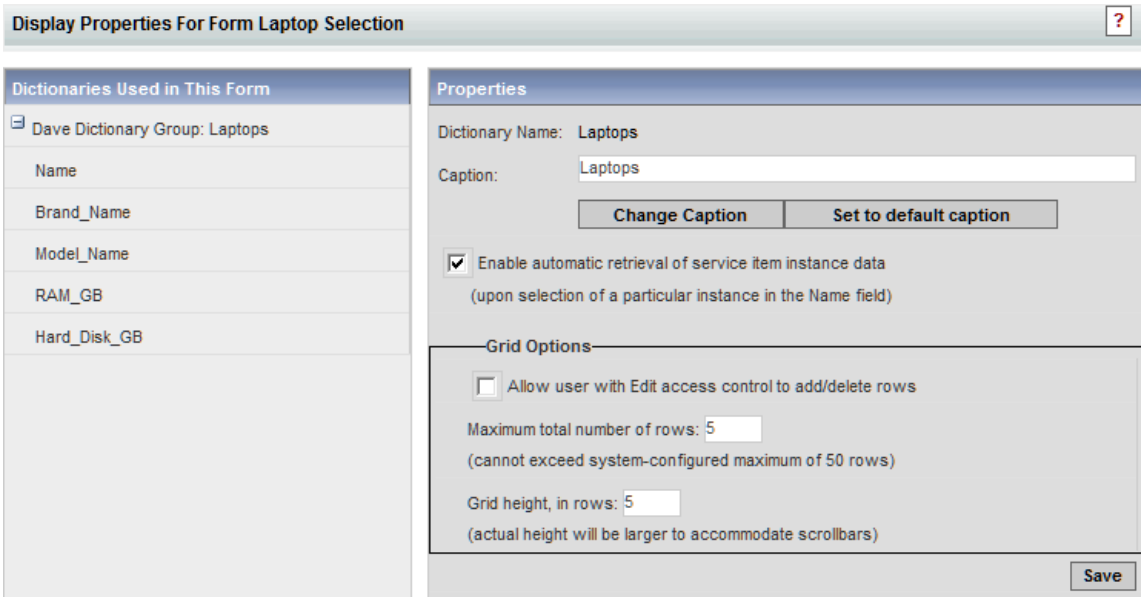
Step 7 Click **Add** to add the Laptops dictionary to the form. The Form Content tab of the View All Laptops form appears with the Laptops dictionary checked for "Display as Grid". Note that it is dimmed here and cannot be changed. The only way to change it would be to remove it, and then add it again without checking "Display as Grid".



Step 8 Click **Save Form** to save the form.

Step 9 Click the **Display Properties** tab for the View All Laptops form.

Step 10 In the Grid Options section of the Laptops dictionary properties, uncheck "**Allow user with Edit access control to add/delete rows**", as shown below.



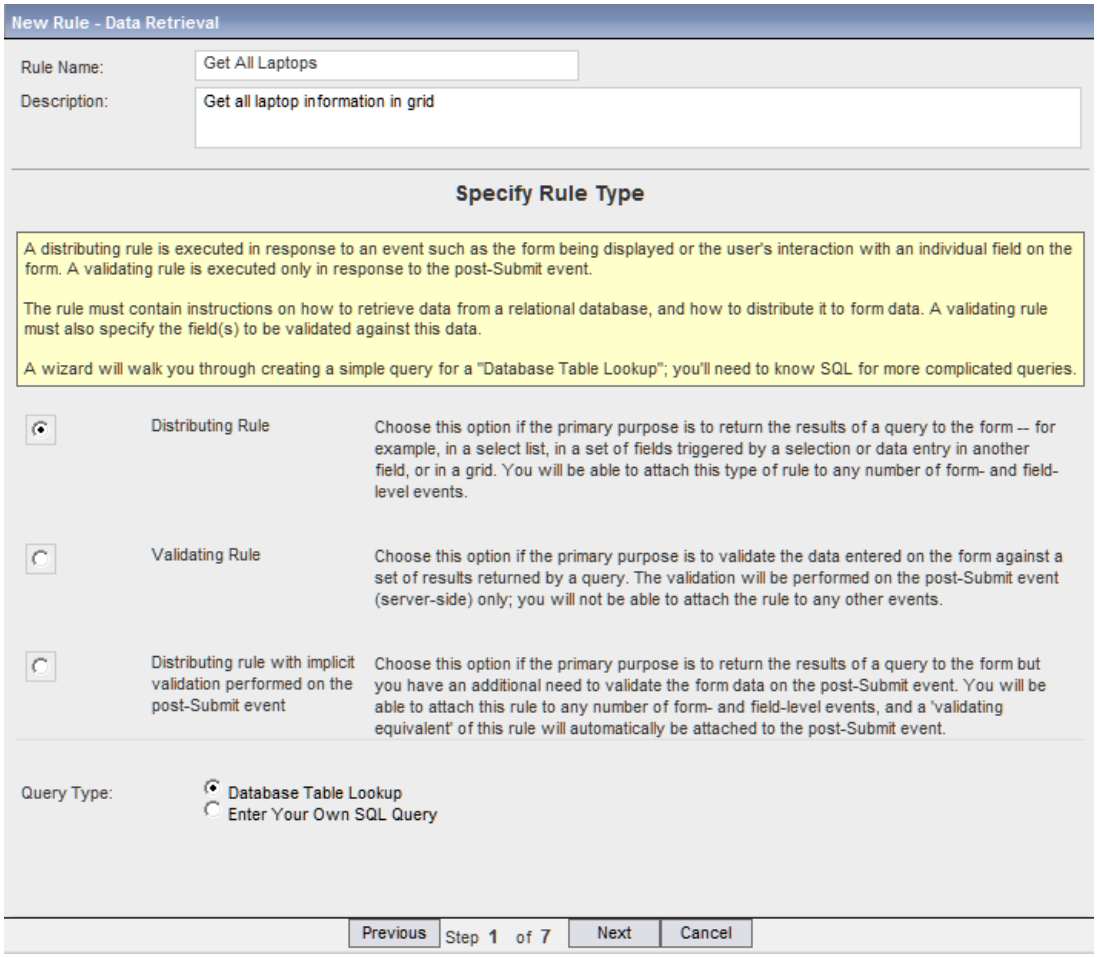
Step 11 In the lower right, click **Save**.

Step 12 Click the Name field below the Laptops dictionary on the left to see its HTML Representation window. Change the Columns value to **32** to make the Name column width smaller. Click each of the remaining fields of the Laptop dictionary and change the Label field to remove the underscore character and add parentheses around “GB”. Also, change the Columns value for the RAM and Hard Disk fields to **32** to make all the columns the same width. In the lower right, click **Save** after each field change.

Step 13 Click the **Active Form Rules** tab for the View All Laptops form.

Step 14 In the bottom left, choose **New Rule > New Data Retrieval Rule**.

Step 15 Step 1 of the Data Retrieval Rule Wizard appears. Enter **Get All Laptops** in the Rule Name field, and **Get all laptop information in grid** in the Description field. Set the Rule Type set to **Distributing Rule** and leave the Query Type set to **Database Table Lookup**.



Step 16 Click **Next** to continue.

Step 17 Step 2 of the Data Retrieval Rule Wizard appears. Choose **Form Load** as the triggering event.

Step 18 Step 3 of the Data Retrieval Rule Wizard appears. Choose **Service Items** from the Datasource drop-down menu, and **Laptops** from the Table Name drop-down menu.

Step 19 Click **Next** to continue.

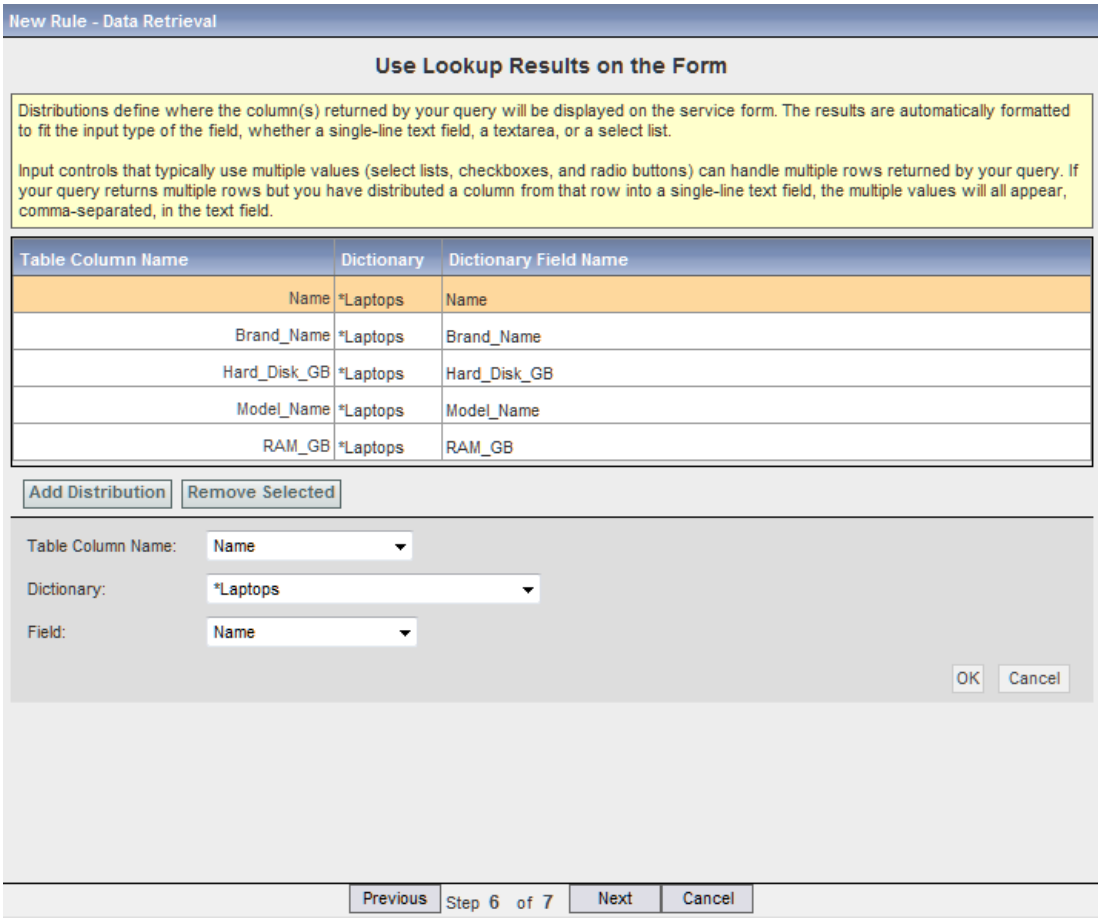
Step 20 Step 4 of the Data Retrieval Rule Wizard appears. We won't create any Lookup Conditions, so just click **Next** to continue.

Step 21 Step 5 of the Data Retrieval Rule Wizard appears. In the Table Column Name field choose **Name** from the drop-down menu, and click **Ascending** for the Sort Direction. Click **OK** to define the sort condition.

Step 22 Click **Next** to continue.

Step 23 Step 6 of the Data Retrieval Rule Wizard appears. To map each column of the Laptops service item to the fields of the Laptops dictionary, choose them from the drop-down menus as shown below and then click **OK** after each one.

Note The "*" next to Laptops indicates that Laptops is a grid dictionary.



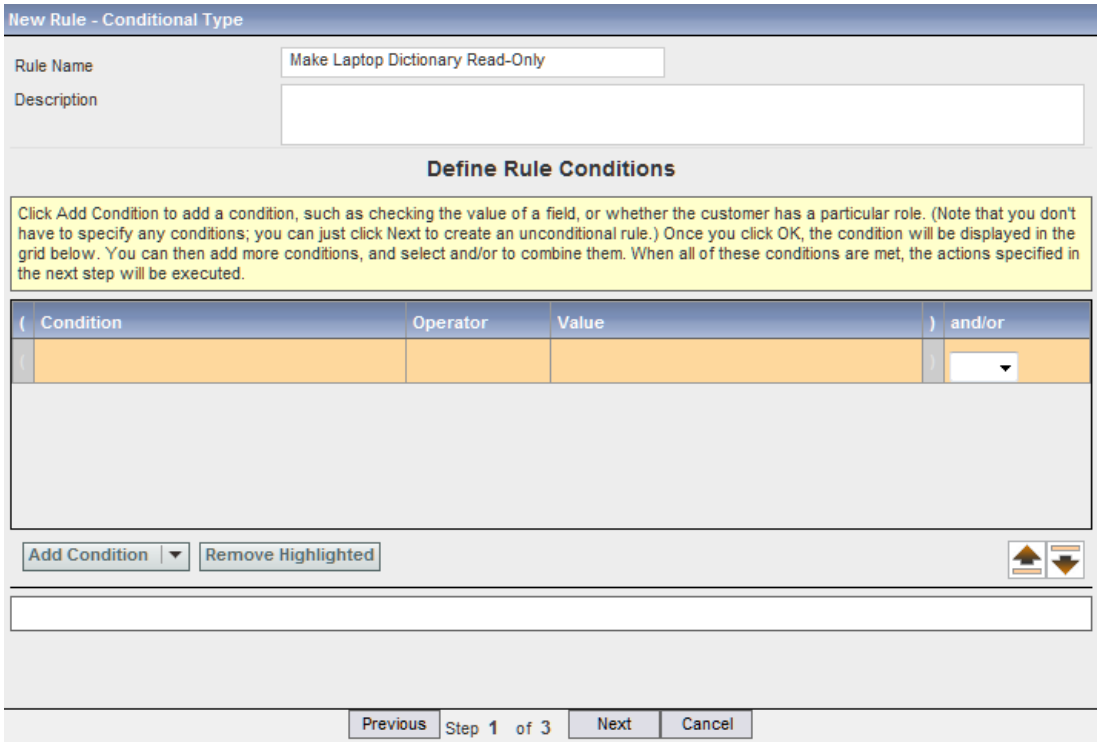
Step 24 Click **Next** to continue.

Step 25 Step 7 of the Data Retrieval Rule Wizard appears.

This final page of the wizard allows you to review and then save your rule. Check to make sure you have entered everything correctly. Click **Previous** if you need to go back to a Step and make changes. You can always go back and edit the rule after you have saved it.

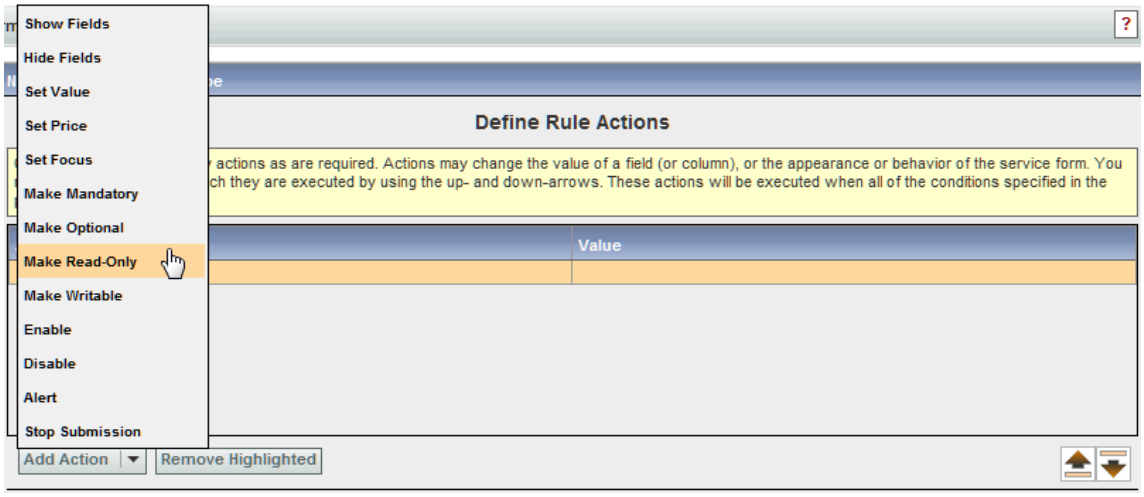
Step 26 To make the grid read-only, you can create a conditional rule to make the grid dictionary read-only. In the bottom left, choose **New Rule > New Conditional Rule**.

Step 27 Step 1 of the Conditional Rule Wizard appears. Enter **Make Laptop Dictionary Read-Only** in the Rule Name field. Enter a description for the form, if desired.

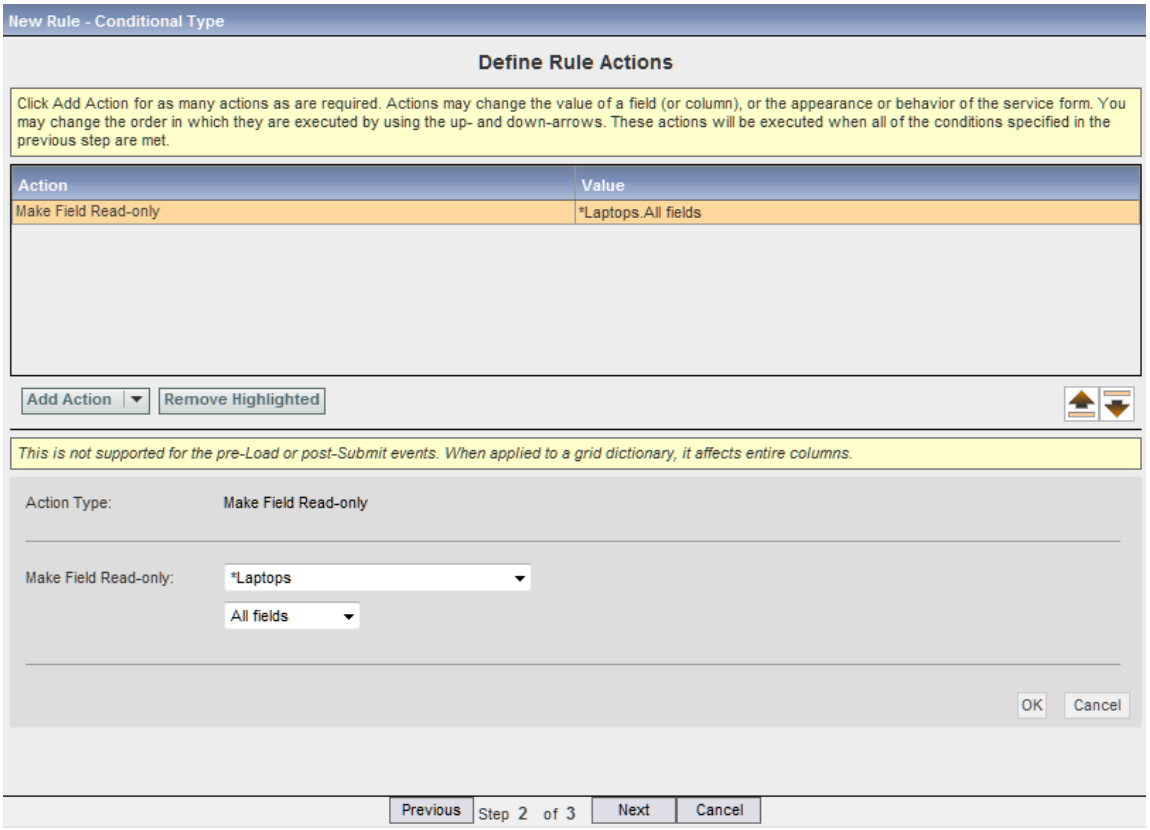


Step 28 We are not adding any conditions, so just click **Next** to continue.

Step 29 Step 2 of the Conditional Rule Wizard appears. Click the **Add Action** drop-down menu and choose **Make Read-Only**, as shown below.



Step 30 In the Make Read-Only drop-down menus, choose ***Laptops** as the dictionary and **All fields** as the field, as shown below. Click **OK** to add the action.

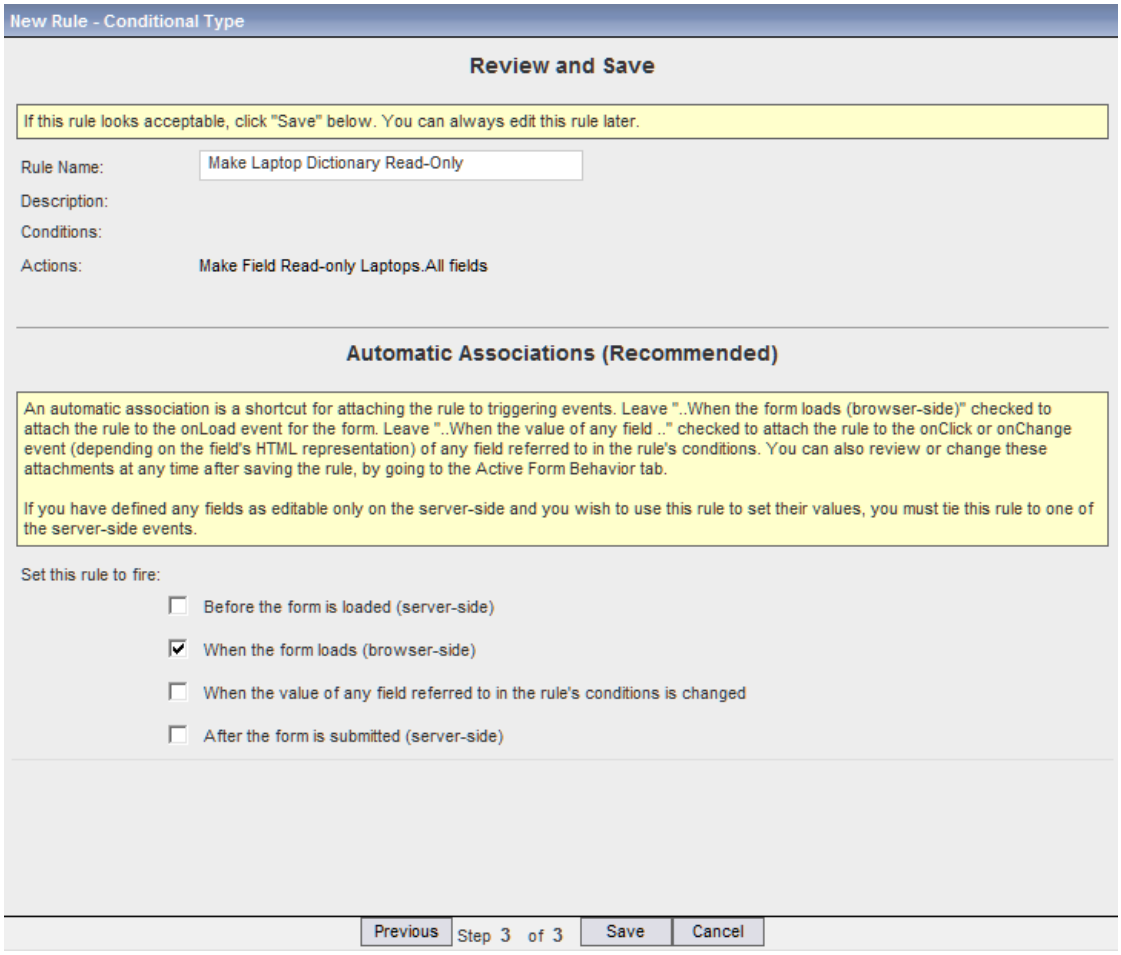


Step 31 Click **Next** to continue.

Step 32 Step 3 of the Conditional Rule Wizard appears.

This final page of the wizard allows you to review and save your rule. Check to make sure you have entered everything correctly. Click **Previous** if you need to go back to a Step and make changes. You can always go back and edit the rule after you have saved it.

In the "Automatic Associations (Recommended)" section, check the "**When the form loads (browser-side)**" check box. Deselect the other check boxes.



Step 33 Click **Save** to complete the Conditional Rule Wizard.

Step 34 Click the **Services** component of the Service Designer module.

Step 35 Choose **New > New Service**.

Step 36 In the Name field of the resultant New Service window, enter **View All Laptops** for this example. Enter "**View all laptop information in grid**" for a description. Choose **Dave Service Group** from the Service Group drop-down menu.

Step 37 Click **Add This Service**. The General Tab of the View All Laptops service appears.

Step 38 Click the **Form** tab of the View All Laptops service.

Step 39 Click **Add Forms** in the bottom left of the window. The Add Form popup window appears.

Step 40 In the Search field, enter **View All Laptops** to search for the View All Laptops form you previously created.

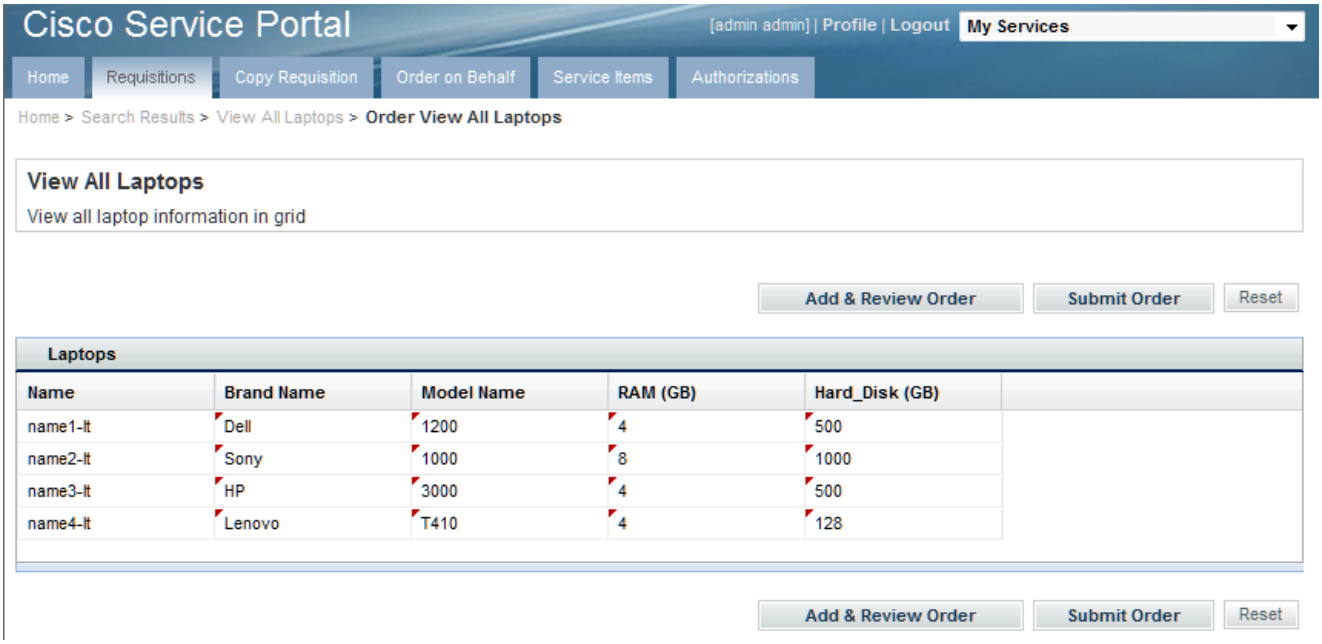
Step 41 Click the **View All Laptops** form and then click **Add**.

The View All Laptops form is added to the View All Laptops service.

Step 42 Now, let's see what the form looks like to the user. Click the **My Services** module of Service Portal.

Step 43 Enter **View All Laptops** in the "Search for Services containing:" field and then click **Search**. The View All Laptops service appears.

Step 44 Click **Order** to see the grid, as shown below.



The grid is read-only and appears with all service item instance data filled in.

Conditional Rules

Conditional rules allow designers to alter the behavior and appearance of a service form. A conditional rule has the following components:

- The rule name and description
- Conditions under which the actions specified by the rule should be executed
- Actions to be taken if all conditions are true

Once the rule has been defined, it is attached to one or more triggering events. The rule may fire, for example, when the page containing the service form is initially loaded into the user's browser; when the user changes the value of a particular field; or in response to other user-initiated actions. The most likely triggering events for the rule may initially be specified via the Automatic Associations listed when you define the rule; you may review or modify these attachments, or add others, by using the Active Form Behavior tab.

Conditional rules provide a powerful mechanism for applying logic both during the “conversation with the end-user” (that is, during the browser session), and before and after that “conversation” (that is, server-side). Using them effectively, however, requires an understanding of what can be done within the browser vs. what can be done server-side. Additionally, you must understand how dictionary fields are represented and stored, and how fields in a grid dictionary differ form their nongrid counterparts.

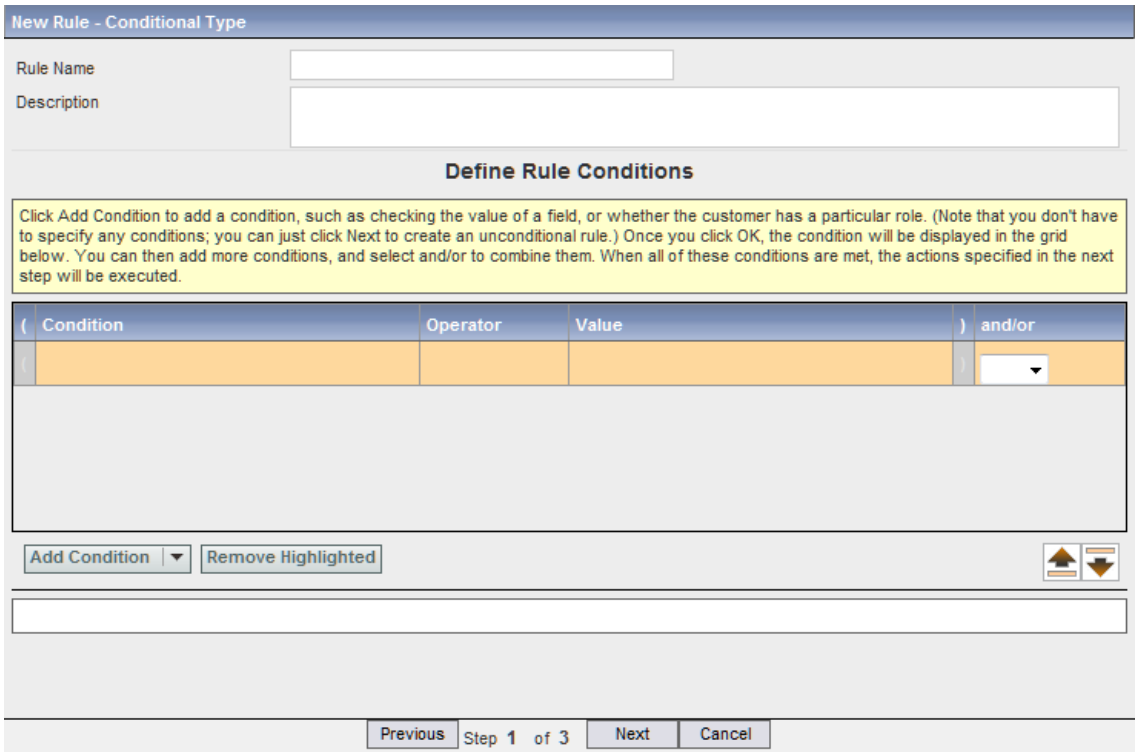
The Conditional Rule wizard (detailed in this section) helps you construct effective rules by explaining, for example, what actions are limited to browser-side or server-side, and what you can and cannot do with grids.

Conditions

The first page (**Step 1**) of the Conditional Rule wizard allows you to formulate a condition by specifying **zero or more** conditional clauses. If no conditions are specified, the actions are unconditionally executed in the triggering events that are specified using the Active Form Behavior tab.

Each component condition evaluates to true or false. Multiple conditional clauses can be combined, using standard relational rules of precedence and the AND and OR operations. If a condition evaluates to true (that is, the combination of all the conditional clauses is true), any actions specified will execute in the triggering events.

Note Grid dictionaries and their fields cannot be chosen as the "Triggering Condition" for conditional rules.



For example, some conditions that might be useful in a service form, and the actions they might entail, include:

Condition	Actions
Is the current value of the DatabaseType field in the Database dictionary equal to "Other"?	If so, display the Description field in the same dictionary, and require the user to enter additional information.
Are you ordering this service on behalf of someone else?	If so, display additional information on the customer (the recipient for the service).
Is the current user a task performer working on the "Order Memory" task?	If so, display the MemoryDetails dictionary and make entry of all fields required.

Click **Add Condition** to build each conditional clause. Depending on which "Condition" you choose, a slightly different Condition Builder dialog box appears, prompting you to formulate the complete condition.

Conditions are summarized in the table below:

Condition	Description/Usage
Dictionary Field	Compare the current value of a field in a dictionary on the service form to a literal; to the value of another dictionary field; or to a blank value or null. For details on the available operators, see the section on Operators below.
Dictionary	Determine the usage of the specified dictionary in the service form at the current time. A dictionary may be viewable, editable, or hidden (previously hidden by a conditional rule or ISF).
User Name	Compare the login name assigned to the current user to the specified value.
User Role	Check whether the current user has been assigned or not assigned the specified role.
Moment	Check whether the request is in a specific moment in the requisition life cycle. Requisition life cycle moments are: • Ordering • Departmental Authorizations • Departmental Reviews • Service Team Authorizations • Service Team Reviews • Financial Authorizations • Service Delivery • Pricing • Service Complete
Task Name	Compare the name of the current task in Service Manager to the specified Task Name. The Task Name is set for all authorization and service delivery moments. The task name is blank for moments when a task is not relevant ("Ordering" and "Service Complete") and when the service form is viewed in My Services, regardless of the current status of the requisition.
Context	The module in which the service form is currently displayed. Valid values are: • My Services • Service Manager
Service Name	The name of the service
Does unsubmitted requisition exist	A Boolean that returns **true** once the service request has been saved. This is **false** only when the service is initially requested, in the ordering moment. Once the request has been "Added" or Submitted", it exists.
Is Order on Behalf	A Boolean that returns **true** if the service has been ordered on behalf of another user, and **false** if the requestor is ordering the service for him/her self.
Customer Role	Check whether the customer for the service has been assigned or not assigned the specified role.

Best Practices/Guidelines for Constructing Conditions

The Context Condition: In some cases, the Context may seem redundant—for example, a request is only visible in the Ordering moment in My Services. However, a service form may be viewed in the Service Delivery moment within Service Manager by task performers and within My Services by the user who originally submitted the request. Similarly, a form may be viewable by multiple participants in the Service Delivery moment. You may want to vary the form's behavior or appearance depending on who is doing the viewing, and where.

The Does Unsubmitted Requisition Exist Condition: When a requisition is initiated, it is in the Ordering Moment. At this time any rules or ISF functions that “prefill” form fields with default values are typically executed. The user may complete data entry and immediately click Submit. In that case, assuming all data in the form is valid, the requisition goes from the Ordering Moment to next moment defined for that service.

In certain circumstances the user may save the requisition without submitting it by clicking “Add and Review”. For example, a user may save a requisition if not all data for the requisition is available; if additional services must be added to the same requisition; or if an attachment must be added. The user may edit an unsubmitted requisition by choosing it in My Services.

When an existing requisition entry is opened for edit before the requisition is submitted, the “Does Unsubmitted Requisition Exist” condition is true; it is false at all other times. Use this condition, for example, to ensure that a rule that prefills default values is executed only for new requests, not for any requests that have not been saved. This ensures the data the user has previously supplied is not overwritten.

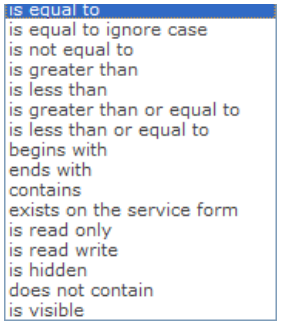
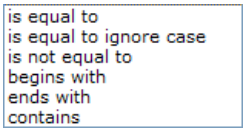
The Moment = Service Delivery Condition: The Service Delivery Moment may encompass many tasks. If a rule is to be conditionally executed for a particular task, use the Task Name condition.

The Task Name Condition: Be careful when referring to a Task Name, since this is a descriptive field that can freely be changed in Service Designer. This problem is minimized if service design guidelines are developed and strictly enforced for naming tasks.

Service designers sometimes include namespaces in task names. For example, the namespace #Name#, referring to the service name, is used to differentiate the same task when it occurs within multiple services. The Task Name used by a conditional rule has all Namespace references properly evaluated. String operations (only checking the beginning or end of the task name, or checking for a phrase contained within the task name) can be used to compare against the portion of the task name that is not derived from the namespace.

Operators in Conditions: The available operators are context sensitive, displayed in a drop-down list that depends on the condition you have chosen. For example, since a field may contain numeric, alphanumeric, or date data, both arithmetic and string operations are allowed, as well as operators to determine the usage of the field. Task and service names are text, so only string operations are appropriate. For conditions which are either true or false, or for which only a limited number of options are possible (such as the current context), radio buttons are available.

Some sample operations are shown in the table below.

Dictionary Field	Task and Service Name	Is Order on Behalf/Does Unsubmitted Req. Exist
		

Most of the operators that can be applied to dictionary fields check the value of that field, comparing it to the value of the specified field or literal. Of these operators, only “is equal to ignore case” is case-insensitive. All other operators are case-sensitive; be sure to take this into account when writing rules such as “starts with” and “contains”, which operate on alphanumeric data.

Other operators, such as “exists on the service form” and “is read only” refer not to the value of the field but to its usage. You do not need to use these rules under most circumstances, since the runtime rule framework automatically checks for the presence of a field on the form and its usage before attempting to execute a rule.

Variables in Conditions: The value of any field in any dictionary may be used in a condition. Since the dictionary may not be included in the current form, the service designer must ensure that forms containing all dictionaries referred to in the rule are included in the service.

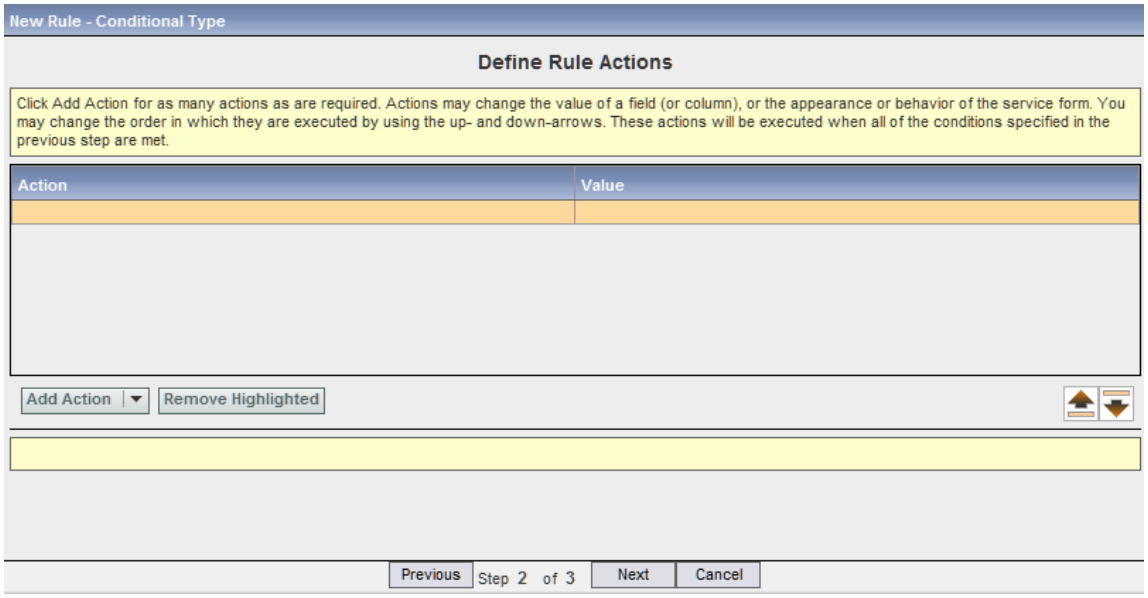
Combining Conditions within One Rule: Multiple conditions can be combined and evaluated to determine if the rule’s actions should be executed. In this case, the rule must include Boolean operators (AND, OR) and may include parentheses to alter the normal precedence of these operators.

The conditions essentially build an “if” clause. The Rule Builder does not currently support an “else” clause. If you need to apply if/then/else logic, define two rules with conditions that are mutually exclusive:

Condition	Actions
Rule 1:DatabaseType_Other Database.DatabaseType is equal to Other	Display the Description field in the same dictionary. Make the Description field mandatory.
Rule 2: DatabaseType_Defined Database.DatabaseType is not equal to Other	Hide the Description field. Make the Description field optional. Set the value of the Description field to blank.

Actions

The second page (**Step 2**) of the wizard shows the rule actions. Click **Add Action** to add each action. Any actions associated with the rule are executed in the order in which the actions are specified.



Actions are summarized in the table below:

Action	Description	Grid Use
Show	Show the specified field, grid column, or all fields in the specified dictionary. The elements are displayed with the representations defined in Service Designer.	Applies to entire column
Hide	Hide the specified field, grid column, or all fields in the specified dictionary.	Applies to entire column
Set Value	Set the value of a target field equal to the value of the specified source field, literal, null/blank or the result of an expression.	Not currently available; use ISF instead
Set Price	Set the service price or the price of the specified child service within a bundle equal to the value of the specified source field, literal or the result of an expression.	Not supported
Make Mandatory	Make the specified field mandatory.	Not supported
Make Optional	Make the specified field optional.	Not supported
Make Read-Only	Make the specified field, grid column, or all fields in the specified dictionary read-only; the field or column cannot be changed by the user, but its value can be changed by a rule or ISF. In the case of a grid, there is no difference between “Make Read-Only” and “Disable”.	Applies to entire column
Make Writable	Make the specified field, grid column, or all fields in the specified dictionary writeable. This action is identical to “Enable”.	Applies to entire column
Enable	Make the specified field, grid column, or all fields in the specified dictionary writeable. This action is identical to “Make Writable”.	Applies to entire column

Action	Description	Grid Use
Disable	Disable the specified field, grid column, or all fields in the specified dictionary. Disabling a field or column makes it read-only, like the Make Read-Only action; however, unlike Make Read-Only, it also dims the field or column, and ignores any changes applied to the field via a conditional rule or ISF. In the case of a grid, there is no difference between "Disable" and "Make Read-Only".	Applies to entire column
Set Focus	Move the cursor to the specified field.	Not currently available; use ISF instead
Alert	Display an alert box with the specified message; only a literal message can be used—namespaces are not supported. The action does not apply to server-side events in general. The only valid use of alerts on the server side is when it is coupled with the Stop Submission action in the post-Submit event.	N/A
Stop Submission	Do not allow the form to be submitted or updated. Should be used only in conditions whose triggering event is to submit the form. When the action is executed on the browser side, it does not stop execution of any other actions in the rule, regardless of its sequence in the actions. When the action takes place on the server side, all subsequent actions and rules are skipped and an error message is presented to the end user. An alert action can be defined optionally prior to the stop submission action to provide the appropriate error message. If no alert message is defined, a generic error message is shown to state that the request cannot be submitted.	N/A

Showing and Hiding Fields and Dictionaries. A frequent use of conditional rules is to show or hide fields or dictionaries based on the current value of another field on the form or any other conditions specified. If you need to hide most, but not all fields in a dictionary, you can hide the dictionary then show only the desired fields. If a field is hidden by a rule, its value is still accessible in other conditional rules.

Making Fields Mandatory or Optional. Making a field mandatory has the same effect as designating a field as mandatory via Service Designer.

- The mandatory symbol is displayed to the left of the field label.
- The user must enter a value in order to successfully submit the form.
- If the administrative settings include the form monitor, the monitor will show the dictionary as incomplete until a value is supplied.

If you are hiding a field (or dictionary), you must ensure that no fields in the dictionary are set to mandatory; if they are, an error message appears when users try to submit the form. If a rule (or ISF) is used to toggle the field between mandatory and optional, the field's HTML representation should define the field as optional. Using a conditional rule to override a field already marked mandatory in a HTML representation may result in unexpected behavior.

If a mandatory field will be toggled between read/write mode, the field should be made optional at the time when it is set to read-only, or disabled to avoid possible confusions to the users.

Setting Focus to a Field. A field on a form is said to **"**have focus**"** when the cursor is in the field, ready for data entry. A field automatically acquires focus when you mouse click in the field, or if you tab from field to field in the form. You may explicitly set the focus to a specified field via a Set Focus action. This is typically used after an alert or when the form submission is stopped, to direct the user's attention to a problematic field.

Setting a Field Value. When you set a field value, you need to take into account the field's data type and HTML representation.

HTML Representation	Set Value Rules
Text, Textbox	Set value to any other field or a literal.
Date Date Time	A literal must have the date and time formats specified in the user's preferences.
Checkbox, radio button, single-select	The value specified must match one of the options listed for the element.
Person	The HTML representation of a Person element actually consists of two items: the person's name (which is visible on the form) and the Person's Id (which is hidden). The SetValue command must set both of these.

If the value is set to a literal or the result of an expression, the literal or expression can include any numeric or text fields used in the current service. (Date arithmetic is not supported.) The field is represented in the expression using lightweight namespace syntax, that is, *#DictionaryName.FieldName#*.

If the expression is invalid (for example, it includes division by zero), the expression is skipped; the value is not updated; and execution of remaining actions or rules for the same event, if any, continues. No error message is displayed to the user.

The "Editable only on server-side" setting for a field also affects the behavior of Set Value. A field marked as "Editable only on server-side" is secured against any attempt to intercept its value during the browser session. In the case of person-based dictionaries and service item-based dictionaries with automatic retrieval enabled, the attribute values retrieved from the database will always override the values sent from the browser when the "Editable only on server-side" setting is enabled. For that reason, any Set Value action you wish to apply to such a field must be in a rule tied to the post-Submit event—in other words, you can use Set Value to edit the field, but only on a server-side event.

Setting a Service Price. The Set Price action supports dynamic pricing of a service. The price may be set to a constant, the value of another field, or the result of an expression formulated using the same rules as for the Set Value action. Although the Set Price action can be included in a rule triggered by any event, the new price actually takes effect only when the service form is submitted; if the transaction is cancelled, the new price is not recorded.

Because the new price takes effect only when the form is submitted, the most appropriate event you should choose for executing a rule containing Set Price is post-Submit.

The Set Price action is recorded in the requisition's System History.

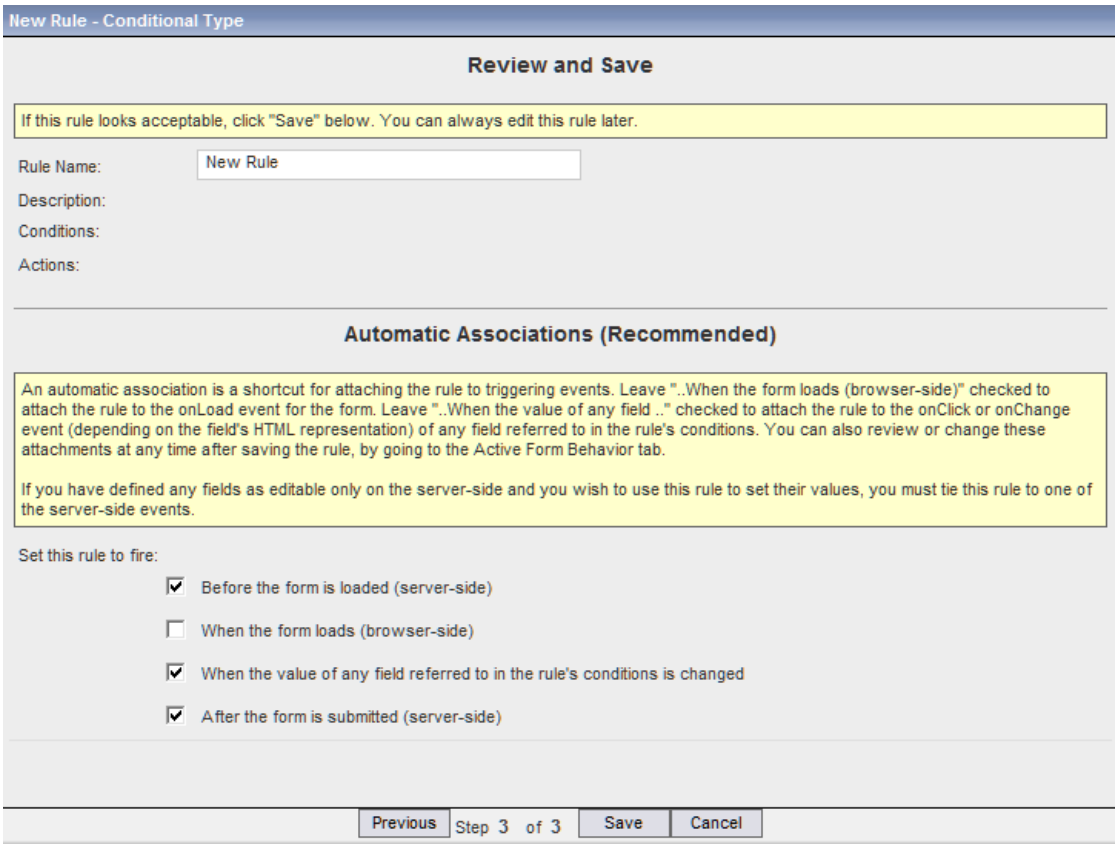
Review and Save

The last page (**Step 3**) of the wizard shows the rule definition. You may click **Previous** to change the conditions or actions.

The Automatic Associations portion of this page allows you to attach this rule to one or more triggering events (Set this rule to fire:):

- **Before the form is loaded (server-side)**: This server-side rule is executed on the server prior to sending it to the browser. (See the "Server-Side Events" section on page 2-32.)
- **When the form is loaded (browser-side)**: This rule is executed when the service form for any service containing this Active Form Component is initially displayed (loaded) in the browser.
- **When the value of any field referred to in the rule's conditions is changed**: In this case, the rule is attached to either the "When the field is changed" or "When the field is clicked" event, depending on the HTML representation of the field.
- **After the form is submitted (server-side)**: This server-side rule is executed on the server after the form is submitted on the browser and sent to the server for processing. (See the "Server-Side Events" section on page 2-32.)

These check boxes are shortcuts for having to attach the rule to the appropriate events via the Active Form Behavior tab. When you edit an existing rule, these check boxes are not shown. You need to review or modify the events to which the rule has been attached on the Active Form Behavior tab. (See the "Active Form Behavior" section on page 2-75.)



Modifying Dictionaries, Fields, and Rules

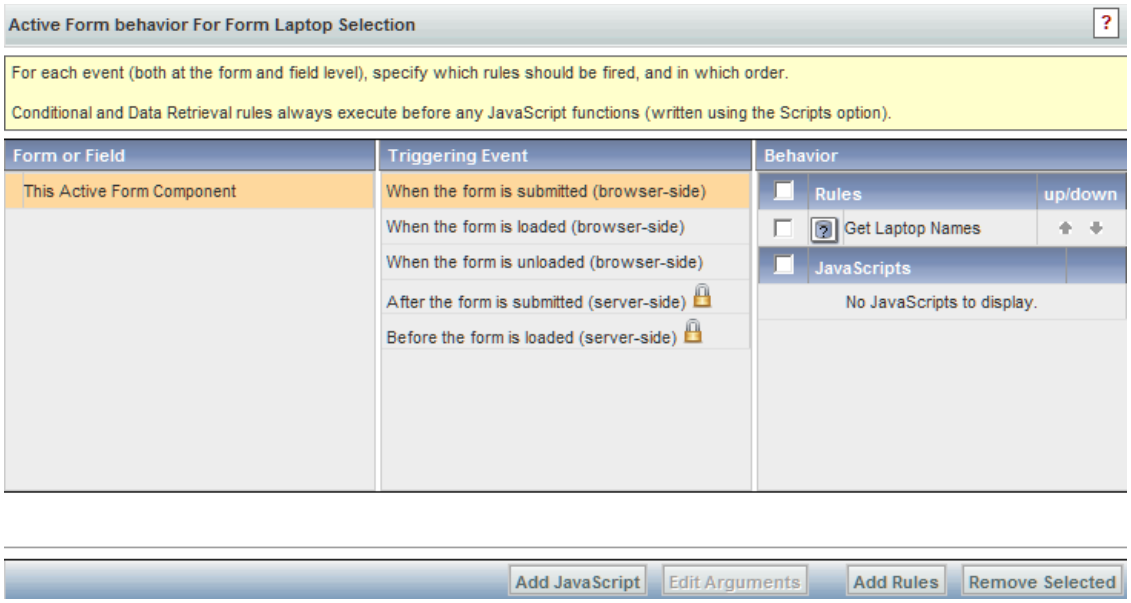
In order to modify an existing data retrieval or conditional rule, you must edit the rule, apply the desired changes, and navigate through all pages of the Rule Wizard. The Save button is available only on the last page of each wizard. This process ensures that all aspects of the rule are internally consistent.

Changes to the dictionaries and fields may not automatically be propagated to a rule which uses the corresponding dictionary, field or lightweight namespace. In case you change the name of a dictionary or field, you must edit the rule, navigating through all pages. For conditional rules and table-based data retrieval rules, references to the dictionaries or fields are updated automatically as you proceed through each page. For SQL entry data retrieval rules, you must re-enter the SQL, using correct lightweight namespaces.

If you delete a field or dictionary, you must edit rules to remove references to the deleted object. If a rule that previously worked suddenly stops working, a renamed or deleted object still referenced in the rule is a probable cause.

Active Form Behavior

The Active Form Behavior tab allows service designers to attach conditional rules and ISF functions (JavaScripts) to events that occur within the processing of a service form; to detach rules or functions from a form; and to change the order in which rules are executed.



Choose the triggering event by highlighting the form (“This Active Form Component” as shown above) or drilling down to the particular dictionary and field and then choosing the appropriate event.

The Form- and Field-Level Triggering Events are:

Form Event	Description
When the form is submitted (browser-side)	The code or rule is executed **after** the Submit or Update button is clicked, and **after** any validations specified in the form's Display (such as checking for numeric or mandatory data) but **before** the form is submitted to the server. This usually provides for a window to do extra validation, formatting, or any kind of processing before the data is sent to the server.
When the form is loaded (browser-side)	The code or rule is executed when the service form is initially displayed in My Services or Service Manager. This is the **preferred** place to add code that populates fields in dictionaries, or that does some kind of form massaging before the form is rendered on the browser. This corresponds to the HTML event: body … onLoad.
When the form is unloaded (browser-side)	This event is called when the form is being closed. This event can be used to notify the user about data being lost if the window is closed. This corresponds to the HTML event: body … onUnload.
After the form is submitted (server-side)	The service form is being prepared on the server prior to being sent to the browser (see the "Server-Side Events" section on page 2-32). This is the best place to attach rules affecting server-only editable fields. It is also the most appropriate place to apply data validations, which is why you will see all Validating (and Implicitly Validating) data retrieval rules attached here automatically.
Before the form is loaded (server-side)	This server-side event is executed on the server prior to sending it to the browser. (See the "Server-Side Events" section on page 2-32.)

Note JavaScript functions cannot be attached to server-side triggers.

Field Event	Description
When the item is changed	The user types something into the specified form field.
When the item loses focus	The user exits from the specified form field, either by tabbing to another field or clicking the mouse in another field.
When the item is clicked	The user clicks on a check box, radio button, or item in a drop-down list.
When the item is focused on	The user clicks in the specified field.
When the mouse is moved off the item	The user moves the mouse off the specified field.
When the mouse is on the item	The user moves the mouse onto the specified field.

You may add as many rules or JavaScripts (functions) as required. Any rules that have previously been attached via the Automatic Associations option of the Active Form wizard also appear. The order in which the rules are executed can be modified by moving the rules up and down within the list. However, the order in which the functions are executed cannot be defined. Therefore, if you need functions to execute in a certain order, create one JavaScript function that contains all functions in the desired order.

The Customer Information form performs some implicit lookups (via the lightweight namespaces) of personnel data from the database. This retrieval is done before any rules or functions are executed in the form load event.

If you have multiple forms in a service, and each has rules or JavaScript functions on the Form Load event, then the rules are executed in order of the forms' appearance within the service, followed by the JavaScripts.

Service Bundles

Multiple services can be combined into one service bundle consisting of one parent service and one or more child services. When the requestor orders a service bundle/the parent service, adjustments are made to the service form's content, appearance and behavior. The behavior of the active form components remains unchanged except for the following:

- Any rules or JavaScripts attached to form-level events (When the form is loaded or submitted) of the child services are ignored. A form component specifying these rules must be included in the parent service.
- Dictionaries are automatically deduplicated; that is, if a dictionary occurs in multiple child services, it will appear only once in the service bundle. The dictionary's configuration matches the configuration specified in the first child service that includes an Active Form Component which, in turn, includes that dictionary. If the configuration of the dictionary is different in other services (for example, by the application of service-specific rules), those differences are ignored.

Service Form Performance and Security Considerations

Service Portal enables you to create service forms quickly and easily. However, the task of creating a form that meets the complex business and technical requirements of a service delivery process, while maintaining ease of use and responsiveness to the end-user, is not at all trivial. It can be a balancing act of ensuring a smooth "conversation" with the end-user while gathering all of the data needed down-stream, whether by third-party systems being orchestrated or by service delivery personnel who may need to perform manual steps.

With that in mind, here are some general guidelines and principles for designing optimal service forms.

Sending Minimal Data to the Browser

In previous releases of Service Portal, any dictionary fields you needed to manipulate had to be sent to the browser in order to be manipulated. This is no longer the case in Release 9.3.2 with the addition of server-side events.

Server-side events (explained fully in the "Form and Field Trigger Events" section on page 2-31) provide a powerful mechanism for updating the values of fields in dictionaries to which the end-user has no permissions (that is, neither Read nor Edit Access Control). Rules executed on server-side events are also able to manipulate the values of any field marked with the "Editable on server-side only" flag on its HTML Representation.

Using server-side events to manipulate data in dictionaries not loaded in the browser has two advantages. First, it results in less data being sent to the browser session, which can improve the perceived performance of the form as it's loaded. Second, it keeps sensitive data under the control of server-side execution and therefore incapable of being intercepted.

For these reasons, you should group fields into dictionaries with an eye toward whether you will ever need certain data present in the browser. Fields that don't need to be loaded into the browser session—at least not during the Ordering moment—should be grouped together into one or more "nonloaded" dictionaries (that is, dictionaries with neither View nor Edit Access Control permissions). A common example of a "nonloaded" dictionary is one supplying the parameters to an orchestration Agent. The orchestrator often needs more data than the user ever sees on the form. This data is typically derived through conditional rules or data retrieval rules that determine who the user is, what role he has, what OU membership he has, and so on; and all of this is best derived either before the form is loaded in the browser (during the pre-Load event) or once the user has clicked the Submit button (during the post-Submit event).

Seeing the effects of rules operating on "nonloaded" dictionaries will take a bit more testing, since you won't be able to see the values being set until the post-Submit event has occurred. Use of "nonloaded" dictionaries will therefore typically involve checking the form in a subsequent moment (for example, Service Group Authorization or Service Delivery), as a user who has the Manage Service Dictionaries permission and is therefore able to see all the dictionaries defined on the form.

Using the Pre-Load and On-Load Events

From the previous discussion of using "nonloaded" dictionaries, you may conclude that it is always preferable to use the pre-Load event rather than the on-Load event. Existing users of Service Portal may be tempted to move all their existing conditional rules and data retrieval rules from the on-Load event to the pre-Load event.

In reality it's not so simple. Generally speaking, conditional rules and data retrieval rules take action on objects that are present in the browser. So a "Hide fields" conditional rule that is tied to the pre-Load event, for example, has in fact nothing to hide, because the fields being hidden don't actually exist yet during the pre-Load event.

There is a notable exception to this general rule, as explained in the previous discussion of "nonloaded" dictionaries. Beginning in Release 9.3.2 it is possible to manipulate the values of fields that are not loaded into the browser, through either the Set Value conditional rule action, or a data retrieval rule. The key for both techniques is that the rules must be executed on the server—in other words, they must be tied to server-side events.

The conditional rule and data retrieval rule functionality is highly flexible, enabling you to combine multiple rule actions on multiple targets, and to tie any one rule to multiple events. Any actions that do not have access to their targets—for example, a Set Value action on a field that is not actually present on the service form—are said to "fail silently" or take no effect. If you have a conditional rule that performs a sequence of actions (say, hiding some fields, making others mandatory, and popping up an Alert message; as well as executing a Set Value), you may see most of those actions taking effect in the browser. The effect of a Set Value action, however, may not be visible because either the target field is currently hidden ***or*** it is not present in the browser. The conditional rule framework doesn't stop you from constructing such a rule; it merely ignores any actions that cannot be executed.

Thankfully, the conditional rule and data retrieval rule wizards spell out these behaviors as you construct the rules. Help text embedded into each step explains any limitations you may experience when using certain actions on the server-side events.

Comparing the Pre-Load and Post-Submit Events

Server-side events provide you with an opportunity to write data directly to the database—specifically, to the WDDX data stored in the database for each requisition entry. The post-Submit event in particular is your window of opportunity for writing the data that was collected or derived from actions in the browser into the database.

An important distinction between the pre-Load and post-Submit events is that pre-Load can write to the database ***only if the requisition data is already in the database***. In other words, if the user has not yet clicked the Order or Add & Review buttons, the data for the requisition doesn't exist in the database and the pre-Load rules have nothing to manipulate. While the pre-Load event gives you the ability to change the ***viewable*** value, it doesn't persist that value in the database. The post-Submit event provides the persistence.

Using ISF

The introduction of server-side events in Release 9.3.2 provides you with great flexibility for manipulating data that is part of the service form yet protected by the server. What makes this possible is that the "nonloaded" dictionaries are stored in the database and therefore accessible by server-side rules. Although the browser-side rules are generated as JavaScript on the form, the server-side equivalents of those rules are actually generated as Java code—so there are, in effect, two separate manifestations of rules possible.

The ISF framework (documented in the "ISF Application Programming Interface (API)" section on page 2-82) enables you to write JavaScript for more complex manipulations of the form data and appearance. This JavaScript can only execute in the browser. Therefore any ISF functions you write cannot be tied to the server-side events.

Using the "Editable on server-side only" Check Box

This new flag, available on the HTML Representation tab for any field defined as Input Type = read-only or hidden, helps you satisfy what are sometimes conflicting requirements: the first is to display helpful data to the requesting user or use "hidden variables" on the form that drive business logic; and the second is to secure data completely from user intervention (including malicious activity).

Two common use-cases for using this flag are:

- The person-based dictionary, a "template dictionary" you can create in the Dictionaries module; and
- The service-item-based dictionary, the structure of which is also automatically created in the Dictionaries module

The two are similar in that they can both exhibit "autopopulation" behavior. In the person-based dictionary, the form user chooses a person (using the Select Person control) and that person's details are automatically displayed on the form. In the service-item-based dictionary, the form user can choose a particular service item he owns or has access to, and the attribute values for that service item instance are automatically displayed.

It's a common service design technique to define most (if not all) of the attributes automatically populated as read-only fields (that is, configured with an Input Type on the HTML representation tab as "read-only"). For example, if your user base is large enough to contain multiple people with the same name, you may use attributes such as the email address to help the form user differentiate among those people and choose the right person. Of course you would not expect the form user to be updating the chosen user's email address. On the other hand, the form user may know this person well and therefore

know that he uses his mobile phone exclusively and not the desk phone number obtained through Directory Integration. If this is the case, you may want to make the Work Phone field editable and not just read-only, so that at least the service delivery personnel handling the request have the most reliable way of reaching the chosen person.

Using the "Editable on server-side only" check box ensures that any fields you mark as Input Type = read-only or hidden are entirely controlled by the server and that any manipulation of their values in the browser session is discarded in favor of the data residing in the Service Portal database. In fact the only mechanism to override the data residing in the database is a rule (either conditional or data retrieval) that executes during a server-side event, and in this case that rule is actually updating the database value. (In the case of the person-based dictionary, the Person record in DirPerson is not updated, but the WDDX data stored in the database for the requisition entry ***is***.)

In other words, you—the service designer—***can*** manipulate the value of a field marked as "Editable on server-side only," but you can do so only through rules executing during server-side events. This is an important factor to consider, because you may be tempted to think that all you have to do to use these "protected" fields is to check the "Editable on server-side only" flag. That is all that's required if you don't have any reason to manipulate the value returned from the database. But if your business logic requires that you set these values in response to selections the form user makes, you must be sure to tie any conditional rule (for a Set Value action) or data retrieval rule (for a data distribution from a query) to a server-side event.

Using Distributing vs. Validating Data Retrieval Rules

The first step of the data retrieval rule wizard offers you three choices for retrieval rule type:

- Distributing Rule
- Validating Rule
- Distributing Rule with implicit validation

Despite there being three choices in the wizard, there are in fact only two types of retrieval behavior: Distributing and Validating. The third choice (Distributing with implicit validation) is simply a convenient combination of the two.

Prior to Release 9.3.2, all data retrieval rules were of the Distributing behavior. The Validating behavior is new. The difference between the two can be summarized thus:

A Distributing rule is primarily for running a query and distributing the results to the form. A Validating rule is for checking the data on the form, to ensure that it conforms to certain standards such as security. In other words, a Validating rule's primary purpose is to prevent malicious users from intercepting the form and manipulating the content.

A Validating rule is effectively answering the question: does this value in this particular field match one of the values that I could have gotten through a particular query?—because if not, I know something is wrong and I need to stop the submission.

There may be times when you want to do both: that is, you want to retrieve data so that it can be distributed to the form; but you also want to ensure that the form user has not done anything untoward in the browser. For this reason, the "combination" rule (Distributing with implicit validation) is offered as a shortcut to creating both types.

The Validating behavior always operates only in the context of the post-Submit event. If you create a Validating rule and examine its event binding on the Active Form Behavior tab, you will see that you cannot choose it for deletion from the post-Submit event. Likewise, any implicit validation cannot be chosen for deletion from the post-Submit event. Removing either of these from the post-Submit event results renders the rule/behavior as meaningless.

There are also performance considerations that should be applied to choosing among Validating, Distributing, and Distributing with implicit validation; those are covered in the next section.

Performance of Data Retrieval Rules

In general, the most secure type of data retrieval behavior is the one checked by default when you create a new rule: Distributing with implicit validation. So, for example, if you wish to populate a drop-down control with a list of server provisioning locations, and the locations are limited by the form user's role, using the "Distributing with implicit validation" option means that a malicious user would not be able to specify "Seattle" when the rule excludes "Seattle" as a valid choice for his role.

Executing the implicit validation comes with a cost, however. Anything executing on the server-side uses cycles on the application server vs. being confined to the user's browser session. For most drop-down lists, there is no undesirable behavior that could result from a malicious user specifying a value other than one actually retrieved by the rule. You need to decide whether taking cycles on the application server is worth the extra security measure. In most cases the "extra load" on the application server will also be trivial, with no overall effect on the performance of the solution to all users. However, the data retrieval rule mechanism is so flexible that it allows even nonoptimal queries to be defined and included in service forms. Some customers, for example, write rules that return thousands of records. While this is not recommended (as the end-user is probably overwhelmed by seeing that many results in a drop-down), there is nothing in the application that prevents such a rule from being defined. It is this type of rule that can affect overall solution performance if executed during a server-side event.

One guideline that applies regardless of the data retrieval rule's triggering event (server-side vs. browser-side) or the type (Distributing vs. Validating) is this: you should refine your query as much as possible, to avoid returning too many results. This will result in the best end-user experience ***and*** the best performance.

Many data retrieval rules—particularly those that populate drop-down controls on the service form—make sense to be executed on the on-Load event. Indeed, this was the only appropriate choice available in releases prior to Release 9.3.2. With the new pre-Load event available in Release 9.3.2, however, you may be inclined to move all of your on-Load rules to the pre-Load event in an attempt to achieve better performance. While it is generally true that a query executing on the server are faster than one being called by the browser, you should consider some additional factors before moving a rule from on-Load to pre-Load.

One factor is the sheer number of results your queries return. If you have designed rules that return hundreds of rows (again, not something that is recommended), such rules executing on the application server may affect the overall performance of the solution, for all users.

Another factor is the perceived performance to the end-user of the form. A complex service form with tens or hundreds of fields will naturally take some time to fully load into the browser window. If that form also contains data retrieval rules that populate drop-downs on the on-Load event, the user is likely to see the form being painted first, followed by the drop-downs being populated. In between the form painting and the drop-downs being populated, the user may see what appear to be "empty" drop-down controls on the form.

This effect can be mitigated if you tie the rules populating the drop-downs to the pre-Load event instead. However, note the term *pre-Load*. Moving data retrieval rules to the pre-Load event means that the form does not get loaded—and therefore the user doesn't see it loaded into the browser—until the rules finish executing. So although the overall performance of the complete loading of the form may be improved, the user's perception *may* be that the application is slow to respond to clicking the Order button.

You, as the designer and tester of the service form, are ultimately the best judge of how much logic to push to server-side events. Luckily it is very easy to tie a set of rules to the on-Load event and subsequently change them to be tied to the pre-Load event, for comparison purposes. If you make this comparison while the application is under load, you are better able to see the effects and choose the approach that is most desirable.

ISF Application Programming Interface (API)

What is ISF?

ISF stands for **Interactive Service Forms.** ISF is a set of interfaces and techniques to add JavaScript programming to a service form. The key word here is **Service Form**; JavaScript programming can only be executed when the service data is displayed—when the service is being ordered in My Services or on the Task Details tab in Service Manager.

When Should You Use ISF and JavaScript?

JavaScript programming, including ISF, is meant to supplement the capabilities provided by active form rules. The most common behaviors associated with enhancing the interactivity of a service form—dynamically showing and hiding fields or dictionaries; setting field values based on the context or on data previously entered; making fields read-only or editable, mandatory or optional—can be provided by the active form rules. The server-side events (pre-Load and post-Submit) cannot, therefore, execute JavaScript.

Writing ISF and JavaScript requires technical (programming) expertise as well as the use of additional tools to debug your code, access the application server and maintain source code control. Consequently, ISF code is more expensive both to develop and maintain than equivalent active form rules. This technology should be used primarily if the desired functionality cannot be implemented via the declarative rules. Some examples are given in the following sections. These use case typically fall into the following areas:

- Manipulating objects on the service form not accessible to the rules, such as dictionary captions, field labels, instructional (help) text, and (currently, at least) cells in a grid dictionary.
- Performing date or numeric arithmetic—for example, computing a scheduled start date based on the date the service was ordered, or computing a service price, based on components chosen.
- Accessing commercially available, freeware-distributed or custom developed JavaScript libraries for specialized functions such as encryption or decryption, or accessing another application via a web service or server-side (AJAX) code.

Some ISF components duplicate functionality available via the active form rules. If your application's requirements can be completely met by using the form rules, that is usually the most effective way of coding. However, in order to ease maintenance of many, complex rules you might decide to implement some equivalent functionality using ISF in the following scenarios:

- Since the rules do not fully support if/then/else logic (they only support the “if” part), two rules are required for every “if” condition with an “else” clause. Once the conditions get more complicated, for example, you need nested if statements, the number of rules required to cover all cases would increase rather markedly. Rather than writing and trying to keep track of all those rules, it might be easier in the long run to write one ISF function, which can include nested if statements.

- Rules are bound to one particular Active Form Component (AFC), while JavaScript functions/ISF are callable from any AFC. ISF could be used for a complex piece of code that needed to be callable from many AFCs, for example, if the same dictionary (for whatever reason) was used in two different AFCs or the same piece of code needed to apply to two fields.
- If ISF needed to be performed in conjunction with some actions that could be done in rules, for example, change a field label or help text, you might consider coding the entire thing in ISF. This is because of the difficulty in ordering ISF and rules—the rules can be explicitly ordered, but the ISF must follow all the rules for the same event.

You can freely combine ISF and the declarative rules in the same form, even on the same event. The most critical limitation to be aware of in using ISF to replace or supplement actions available in the rules is that JavaScripts must be run after any rules that are attached to the same event.

ISF Components

ISF includes global identifiers as well as a set of public functions.

Global Identifiers

ISF global variables and their possible values are summarized in the table below and discussed in more detail in the following paragraphs. When these identifiers have an equivalent condition in the conditional rules, that equivalence is noted. More details may be found in the preceding sections on Active Form Components.

Table 2-4 ISF Global Identifiers

Global Variable	Description
Context	The module in which the service form is currently displayed. Equivalent to the “Context” condition.
EditRequisitionBeforeOrdering	A Boolean that returns **true** in the ordering moment when an existing (previously saved) Requisition Entry is being edited and **false** under all other conditions. An alternative way to discover this condition is to evaluate (ReqID==0), since a RequisitionId is assigned when a requisition is saved. Equivalent to the condition “Does unsubmitted requisition exist?”
Moment	The current moment in the requisition life cycle. Equivalent to the “Moment” condition.
ReqCustomerID	The unique identifier of the customer for the service. It may be different from the user that is making the request. The ReqCustomerID value is available in all moments. This is a reference to the Person’s unique identifier in Service Portal.
ReqEntryID	The Requisition Entry (also known as Service Request) ID. The ReqEntryID is zero (0) until the requisition has been saved.
ReqID	The Requisition ID (also known as Shopping Cart). The Requisition ID is zero (0) until the requisition has been saved.
ReqInitiatorID	The unique identifier of the person that initiated the service request. The ReqInitiatorID value is available in all moments. This is a reference to the Person’s unique identifier in Service Portal.

Table 2-4 ISF Global Identifiers (continued)

Global Variable	Description
ServiceID	The unique ID of the service; included for backwards compatibility; should not be used for services deployed via Catalog Deployer, which does not preserve entity IDs between sites.
ServiceName	The name of the service. Equivalent to the "Service Name" condition.
TaskID	The ID of the task being viewed in Service Manager. The TaskID is set for all authorization and service delivery moments. The TaskID value is zero (0) for moments in which no task is active, typically before the authorization or service delivery begins.
TaskName	The name of the task being viewed in Service Manager, with all Namespace references properly evaluated. Equivalent to the "Task Name" condition.
UserID	The person that is making the request, or, when the context is Service Manager, the person that is working with the request.

Person References

All users must be registered in Organization Designer. The application identifies these users by assigning a unique identifier to their records in Organization Designer. The application tracks the initiator of the current requisition (ReqInitiatorID); the customer for the current requisition (ReqCustomerID); and the user currently working with the requisition (UserID).

In the ordering moment, the ReqInitiatorID is always equal to the UserID. In service delivery and authorization/review moments, the UserID identifies the task performer or reviewer. The ReqCustomerID is different than the ReqInitiatorID if the Order On Behalf (OOB) capability is used; otherwise, these values are identical.

JavaScript Functions

JavaScript functions are built into the ISF framework. ISF is an object-oriented framework. The ISF JavaScript functions are actually methods which are based on the base object serviceForm. To hide a dictionary, for instance, the user calls

```
serviceForm.DictionaryName.setVisible(false).
```

To hide a field the user calls

```
serviceForm.DictionaryName.FieldName.setVisible(false).
```

In the following sections, bold and italicized typeface means that the programmer should substitute the name of the item.

Note JavaScript functions cannot be assigned to any event of grid dictionaries and their fields.

Dictionary-Level Functions

Functions that are applicable to dictionaries are summarized in the table below and explained in more detail in the following paragraphs. For grid use, see the "Conditional Rules and ISF for Manipulating Grids" section on page 2-24.

Table 2-5 Dictionary-Level Functions

Function	Usage
serviceForm.***dictionaryName***	Returns the dictionary object if the dictionary exists in the form, and undefined otherwise.
serviceForm.***dictionaryName***.setVisible(Boolean)	Hides or makes visible the dictionary or grid, and all the fields in the dictionary; has no effect on a dictionary which is hidden in Service Designer.
serviceForm.***dictionaryName***.isVisible()	Returns true if the dictionary or grid is visible and false otherwise.
serviceForm.***dictionaryName***.setShowCaption(Boolean)	Hides or displays the caption for the dictionary.
serviceForm.***dictionaryName***.getCaption(stripTags)	Gets the caption text for the dictionary, or the title text for the grid, optionally stripping any HTML (stripTags is Boolean).
serviceForm.***dictionaryName***.setCaption(newCaption)	Sets the caption text for the dictionary, or the title text for the grid (newCaption is a string).
serviceForm.***dictionaryName***.setReadOnly(Boolean)	Sets all the fields or grid columns in the dictionary to be read-only or read-write; has no effect if the dictionary's Access Control is set to Edit only (not View) in Service Designer, or if the columns in Service Designer were already set to read-only. See details of the field-level **setReadOnly** for implications on specific field-types.
serviceForm.***dictionaryName***.isReadOnly()	Returns true if the dictionary is read-only and false otherwise.
serviceForm.***dictionaryName***.getGridSize()	Returns the number of rows in the grid.

Dictionary Permissions and ISF Dictionary-Level Functions

The appearance (and HTML representation) of a dictionary specified as read-only via Service Designer (for a specified moment or set of participants) is different from the appearance (and HTML representation) of a dictionary which is set to read-only via the ISF ***dictionaryName*.setReadOnly()** function.

- When the dictionary is set to Edit only via Service Designer, no HTML input tags are generated for the fields which comprise the dictionary; they are rendered on the service form as text.
- When the dictionary's Access Control includes Edit capability and it is set to read-only via ISF or a conditional rule, the dictionary fields are displayed as input objects; however, they are not enterable.

When a dictionary is read-only at design time (that is, does not have Edit permission for the current participant and moment as specified in the Access Control tab for the Active Form Components in Service Designer) it cannot be made writeable through ISF or rules. This is true because when the read-only dictionary is rendered, the resulting HTML includes text with <span ../> tags; HTML <input ..> tags are not present.

Dictionary Permissions and Administrative Users

Dictionary permissions are ignored for a user who has been assigned the "Manage Service Dictionaries" capability. (This capability is automatically granted to any users who are in the "Site Administration" organization and may be included in user- and Service Portal-defined roles as well.) The dictionary will appear as if it were Edit-able. Care should be taken to not test any ISF when logged in as an administrative user, since the "Manage Service Dictionaries" capability overrides the designated dictionary permissions.

Checking for the Existence of a Dictionary

The ISF expression:

```
serviceForm.dictionaryName
```

is not a function which returns a Boolean. The ***dictionaryName*** is an attribute of the serviceForm object. The ***dictionaryName*** attribute has a value of true if the dictionary exists in the service in which the ISF is executed, or undefined if the dictionary does not exist. Therefore, robust code that checks for the existence of one or more dictionaries and takes action only if the dictionaries are present in the current service might be coded as follows:

```
AIT_Server_onLoad() {
 if (serviceForm.RC_CUSTCODES != undefined)
    {RC_CUSTCODES_onLoad();)
  if (serviceForm.RC_PERFORMWORK != undefined)
     {RC_PERFORMWORK_onLoad();)
    }
```

Field-Level Functions

Functions that are applicable to fields are summarized in the table below and explained in more detail in the following paragraphs.

Table 2-6 Field-Level Functions

Function	Usage	Grid Use
serviceForm.***dictionaryName.fieldName***	Returns the field object if the field exists in the form, and undefined otherwise.	No
serviceForm.***dictionaryName.fieldName***.getValue()	Returns the current value of the field.	No
serviceForm.***dictionaryName.fieldName***.setValue(inputValues)	Sets the value of a field.	No
serviceForm.***dictionaryName.fieldName***.getCellValue (RowIndex)	Returns the cell value of the grid column at specified RowIndex.	Yes
serviceForm.***dictionaryName.fieldName***.setCellValue (RowIndex, inputValue)	Sets the cell value of the grid column at the specified RowIndex. inputValue takes a single value instead of an array.	Yes

Table 2-6 Field-Level Functions (continued)

Function	Usage	Grid Use
serviceForm.***dictionaryName.fieldName***.setValue(inputValues, defaultValue)	Sets the value of a field. inputValues has to be an array, the first value of inputValues is assigned to any single-option field. The defaultValue is applicable only for a field with an input type of text or textarea. For any other field type the defaultValue is ignored.	No
serviceForm.***dictionaryName.FieldName***.setSelection(inputValues)	Set the value of a field with an HTML representation of select (single), select (multiple), checkbox or radio button.	No
serviceForm.***dictionaryName.fieldName***.setMandatory(Boolean)	Makes the field mandatory or optional. If the field is mandatory via Service Designer settings, it cannot be made nonmandatory. When a field is made mandatory, validation is automatically assigned; an error message, "Required: Please fill out this field before submitting" is displayed if the form is submitted with this field blank.	No
serviceForm.***dictionaryName.fieldName***.isMandatory()	Checks it the field has been marked mandatory. Returns true if the field is mandatory, either via the ISF setMandatory() function, a conditional rule, or a Service Designer Display setting.	No
serviceForm.***dictionaryName.fieldName***.setReadOnly(Boolean)	Makes the field or grid column read-only or read-write.	Yes
serviceForm.***dictionaryName.fieldName***.isReadOnly()	Returns true if the field or grid column is read-only and false otherwise.	Yes
serviceForm.***dictionaryName.fieldName***.setVisible(Boolean)	Makes the field or grid column hidden or visible (displayed). If the column is hidden in Service Designer settings, it cannot be made visible.	Yes
serviceForm.***dictionaryName.fieldName***.isVisible()	Returns true if the field or grid column is visible and false otherwise.	Yes
serviceForm.***dictionaryName.fieldname***.setFocus(Boolean)	Sets focus to the field; applicable to all input types except for radio buttons and checkboxes. True focuses the field; false blurs the field. This function does not apply for fields that are in dictionaries set read-only via Service Designer or for hidden fields—the call is ignored and no error is shown.	No
serviceForm.***dictionaryName.fieldname***.setFocusViaValidation(Boolean)	Sets focus to the field, applicable to radio buttons and checkboxes only.	No

Table 2-6 Field-Level Functions (continued)

Function	Usage	Grid Use
serviceForm.***dictionaryName.fieldName***.getInstructionalText(stripTags)	Returns the instructional text for a field or grid column, optionally stripping any HTML from the text based on the Boolean stripTags argument.	Yes
serviceForm.***dictionaryName.fieldName***.setInstructionalText(text)	Sets the instructional text for a field or grid column.	Yes
serviceForm.***dictionaryName.fieldName***.getPrompt(stripTags)	Returns the prompt for a field or grid column header—optionally stripping any HTML from the prompt based on Boolean stripTags argument.	Yes
serviceForm.***dictionaryName.fieldName***.setPrompt(prompt)	Sets the prompt for a field or grid column header.	Yes

Checking for the Existence of a Field

The ISF expression:

`serviceForm.`***`dictionaryName.fieldName`***

is not a function which returns a Boolean. The ***fieldName*** is an attribute of the serviceForm.***dictionaryName*** object. The ***fieldName*** attribute returns undefined if the field does not exist in the current service. (The field may be hidden, and it is still considered to exist.) Therefore, robust code that checks for the existence of one or more dictionaries and takes action only if the dictionaries are present in the current service might be coded as follows:

```
RC_REQUESTEDBY_onLoad() {
  if (serviceForm.RC_REQUESTEDBY.FirstName != undefined)
    {serviceForm.RC_REQUESTEDBY.FirstName.setReadOnly();)
  if (serviceForm.RC_REQUESTEDBY.LastName != undefined)
    {serviceForm.RC_REQUESTEDBY.LastName.setReadOnly();)
}
```

Checking for field existence is not necessary if code has previously confirmed that the dictionary "exists".

```
commonOnLoad() {
  if (serviceForm.RC_REQUESTEDBY != undefined) {
    RC_REQUESTEDBY_onLoad();
  }
  …
}

RC_REQUESTEDBY_onLoad() {
  serviceForm.RC_REQUESTEDBY.FirstName.setReadOnly();
  serviceForm.RC_REQUESTEDBY.LastName.setReadOnly();
}
```

Getting the Value of a Field

The getValue() method (serviceForm.***dictionaryName.fieldName***.getValue()) always returns an array. If there is only one item then it is an array of one. Use .getValue()[0] to access the first element. The getValue() method works on all fields, regardless of whether the field is read-only, read-write, or hidden in the current moment, provided that the dictionary is defined with edit access.

- For input types like checkbox and Select (Multi), getValue() is processed incorrectly if the values contain tabs (for example, if an attempt was made to import data from an external source where tab is used to delimit distinct values in lists). The tab character is represented within the value as \t.
- For complex controls (radio, checkbox, multi-select, single-select/drop-down) the value returned is the set of "selected values"—meaning only the highlighted values are returned and the "Value" property is used rather than the label or text often seen by the user.

  For security reasons, this method does not work for a field with an input type of password. It returns a blank string—no error is shown.

Setting a Field to Read-Only

The setReadOnly() method (serviceForm.***dictionaryName.fieldName***.setReadOnly()) toggles a field between read-only or read-write.

- Fields with the input types of person, date or datetime have an associated button that the user clicks to choose a value for the field. Setting these fields read-only disables the Select button next to the field so it cannot be clicked. The text field containing the descriptive information (person's name, date, or datetime) is always read-only.
- When a dictionary is marked as read-only for a particular moment in Service Designer, fields appear on the service form as boilerplate text; no HTML input object is generated. Such a dictionary (or fields in the dictionary) cannot be made read-write through ISF. Attempts via ISF setReadOnly() to make the dictionary or a field in the dictionary writeable silently fail. Similarly, attempts via ISF to make the field or dictionary read-only have no effect and do not generate an error.
- If a dictionary or field is made read-only through ISF, the HTML input box is still displayed but field contents cannot be edited and the field is removed from the tab sequence.

Setting a Field to Visible

The setVisible() method (serviceForm.***dictionaryName.fieldName***.setVisible()) toggles a field between being hidden and visible on the service form.

- When a dictionary is marked as hidden for a particular moment in Service Designer, fields in that dictionary cannot be made visible via ISF. Attempts via ISF setVisible() to make the dictionary or a field in the dictionary visible silently fail.
- When a dictionary has been hidden via ISF, attempts to make fields in the dictionary visible silently fail.

Setting the Value of a Field

Two methods are available for setting the value of a field:

- serviceForm.***dictionaryName.fieldName***.setValue())
- serviceForm.***dictionaryName.fieldName***.setSelection())

A third method is the equivalent of the setValue() method, but for cells in a grid:

- serviceForm.***dictionaryName.fieldName***.setCellValue())

See the "Conditional Rules and ISF for Manipulating Grids" section on page 2-24 for more details.

The setValue() method sets the value of the specified field to the specified inputValues. inputValues has to be an array; the first value of inputValues is assigned to any single-option field.

- For security reasons, this method does not work for fields with an input type of password.
- inputValues for Select (single), Select (multiple), checkbox and radio button input types are set to the display values and this function selects the same. Checkbox type fields can take an array of inputValues. The function returns without selecting when invalid display values are passed that are not in the field. On saving the service form, the selection is persisted.
- No validation is performed on the inputValues. It's possible for wrong values set through this function to be persisted when the service form is submitted, even for field types like Person, SSN, URL, or Date.
- For Person type fields, see the "Specialized Field-Level Functions" section below.

The setSelection() method should be used for fields with multiple options—select (single), select (multiple), checkbox, and radio. The argument is matched to the values for the various elements in the field. For a radio field or checkboxes, this function allows you to set the selection to one or none of the controls. For example: .setSelection(['']) clears all the radio buttons and restores the "pristine state".

For a multi-select or a single-select input item, setSelection marks the requested items selected. Items that are not found in the list are ignored.

This function does not do anything for fields with an input type of password, text, or textarea, or for fields in read-only dictionaries.

Specialized Field-Level Functions

Some field-level functions are applicable only to fields of particular types. These are summarized in the table below.

Table 2-7 Special Field-Level Functions

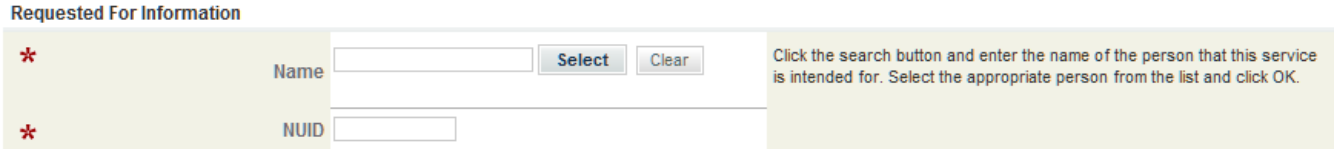
Function	Usage
serviceForm.***dictionaryName.fieldName_disp***.getValue()	Applies to Person type fields only. ***fieldName***.getValue(inputValues) gets the PersonID of the specified field. The display value of the Person field (generally the person's name) can be accessed by suffixing '_disp' to the fieldname: …***fieldName_disp***.getValue().
serviceForm.***dictionaryName.fieldName_disp***.setValue(inputValues)	Applies to Person type fields only. ***fieldName***.setValue(inputValues) sets the PersonID of the specified field. Use ***fieldName_disp***.setValue(inputValues) to set the value displayed in the text box.

Table 2-7 Special Field-Level Functions (continued)

Function	Usage
serviceForm.***dictionaryName.fieldName_saved***.getValue()	For single- and multiple-select fields, the normal fieldName.getValue() always returns the value that is currently selected. The saved values for Select type of fields can be accessed by suffixing "_saved" to the actual field name –. ***fieldName_saved***.getValue() This function can be useful to determine the values that was saved in previous moments for Select lists. This function also returns values previously saved in the same moment. Although ***fieldName_saved***.setValue() executes without error, it is nonfunctional, and does not change the saved value of the field. **Note** This function is not available for grid dictionaries.

Person-Based Fields

The person data type and HTML representation are designed for the display and validation of person profile data stored in Organization Designer. The field appears on the service form as a single-line text box with an associated control labeled "Select", as shown below.



Clicking the "Select" button opens a popup window allowing the user to search for people in the Service Community, and choose the person of interest. The person's name and email address (fields configured in Site Administration) are displayed on the service form.

ISF functions operate as follows when applied to a person-type field:

- ***personFieldName***.getValue() returns the ID of the person, the unique identifier for the person record in Service Portal. That ID can be used, with appropriate, customized, server-side code to lookup additional person profile information for display in other fields on the service form.
- ***personFieldName_disp***.getValue() returns the name of the person as currently displayed on the service form.
- Using the Select control automatically updates both the PersonField and PersonField_disp fields, so they are kept in sync.
- ***personFieldName***.setValue() may be used to set the value of the PersonID. This function can be used in conjunction with ***personFieldName_disp***.setValue(), so that the correct name is always displayed for the current ID.

The PersonField.setValue () function expects a valid Person ID, but no validation is done by the client—so assigning an invalid ID does not cause the submit/update action to fail. The display value of the Person field can be accessed by suffixing '_disp' to the fieldname:

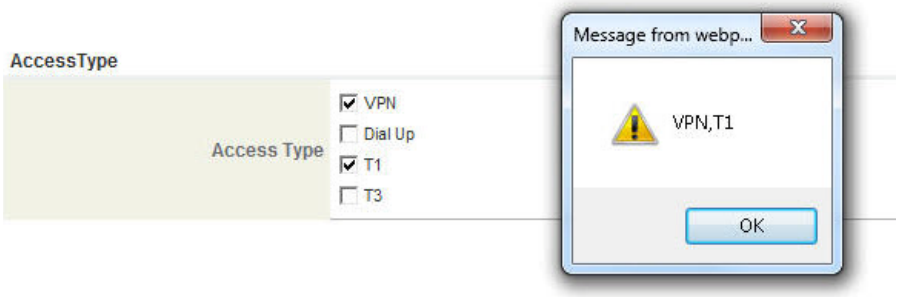
fieldName_disp.setValue(inputValues).

- When the service form is submitted, the Person ID value that was set using this function is retained—the display value is ignored.
- When the Person name changes, the service form does not synchronize the Person display value; that is, it stills show the value that was set using this function.

Fields allowing Multiple Values/Selections

The HTML displays for multi-select fields and for checkboxes allow multiple values to be chosen for the same field. The values chosen are represented as a comma-separated list of values, as shown in the following example:

```
alert (serviceForm.EUIT_RemoteAccessDetails.AccessType.getValue([0]));
```



A JavaScript split() method can be used to parse the field value into its distinct elements.

Integrating ISF Code into Service Forms

ISF and service forms implement an event model which is similar, but not identical, to the Document Object Model (DOM) event model. That is, customized JavaScript functions may be invoked to handle events which occur during the processing of a service form.

JavaScript functions are typically invoked as event handlers for processing events which occur as an HTML form is displayed and the user enters data in the form's input fields. Since service forms are generated dynamically, based on the dictionary and form definitions previously stored in the repository, it is not possible for programmers to simply type ISF code into an HTML file. They must rely on the user interface provided by Service Designer to write their functions and to attach these functions to the appropriate event. Therefore, any JavaScript function to be accessed as an event handler must also be defined within the repository. Such functions are defined via the Scripts option of Service Designer, and associated with the appropriate event via the Active Behavior tab for the form.

JavaScript functions written as event handlers can, in turn, call other JavaScript functions. These functions (if they are to have a public scope) cannot be defined as Scripts within Service Designer. Instead, they must be written in a JavaScript library, a text file comprising one or more functions, which resides on a file system accessible to the application server. The Service Designer interface is then used to include these libraries in forms in which their functions are required.

The Service Designer options which are used to write ISF code and to integrate that code into a form are:

- **Scripts > JavaScripts**: Create an individual ISF function, optionally include a library in the function, and monitor function usage.
- **Scripts > Libraries**: Create a reference to a JavaScript library which can contain multiple functions.
- **Active Form Components > select a form > Active Form Behavior tab**: Attach a JavaScript as an event handler for a form or a field on a form.

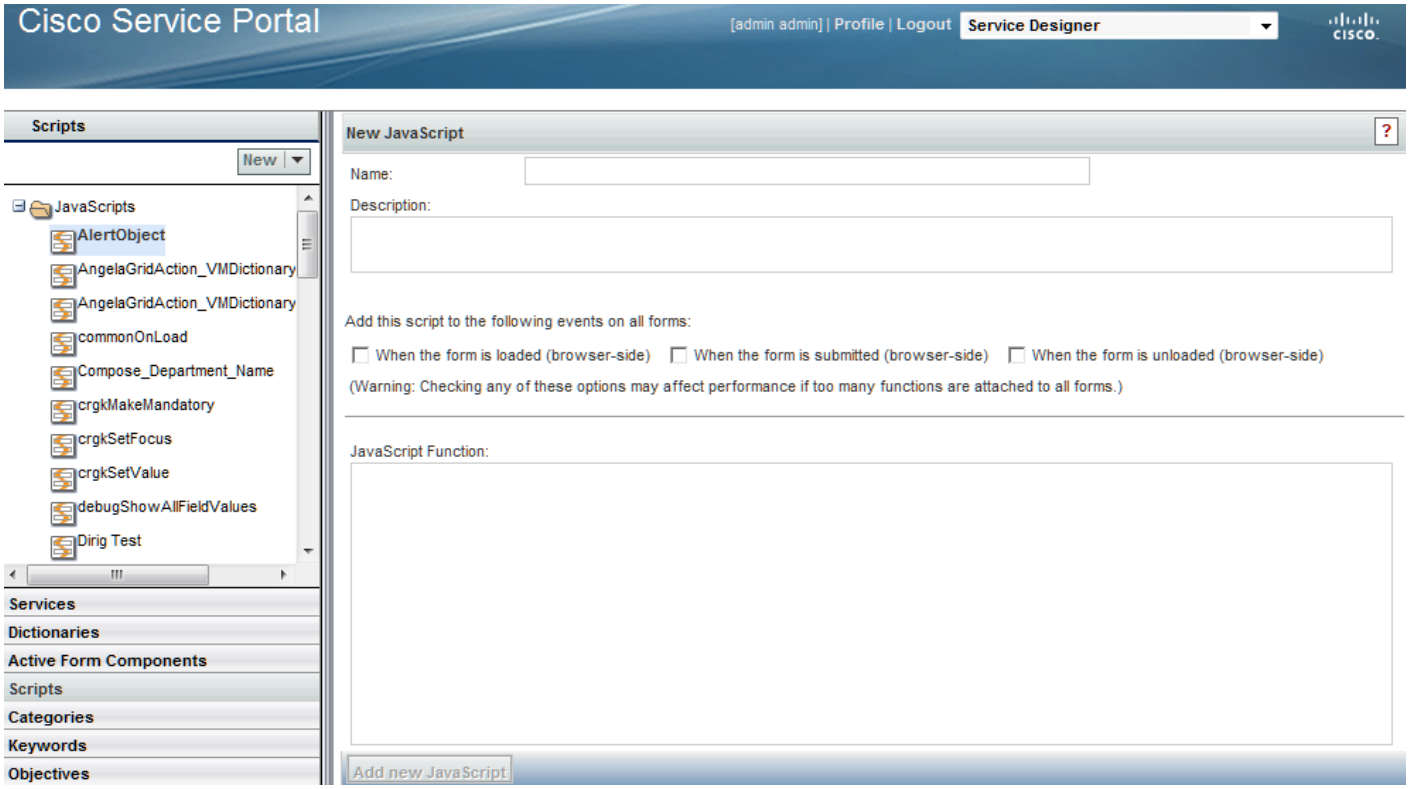
JavaScripts

When choosing a **JavaScript** function under **Service Designer > Scripts**, three tabs appear:

- The General tab allows you to create and maintain a JavaScript function.
- The Libraries tab allows you to include a JavaScript library in the current function and, by extension, in the form to which the function is attached.
- The Active Form Components tab displays the forms to which the current function is attached.

Creating and Maintaining JavaScript Functions

To create a new JavaScript function, choose **New > New Function** under **Service Designer > Scripts**. Once the function has been created, choose it for maintenance from the tree structure on the left.



- **Name**: The name of the function. Although this name is just used within Service Designer, to refer to the JavaScript function, it is best practice to use a JavaScript identifier which will identify the function within the generated code. As a JavaScript identifier, the name is a single case-sensitive word consisting of letters and numbers and starting with a letter. For guidelines on naming functions, see the "Best Practices for Using Active Form Components" section on page 2-111.
- **Description**: Optional, but highly recommended description of the function.
- **Add this script to the following events on all forms**: These check boxes provide a way to specify the function as the event handler for the checked event. This "global" attachment would replace the "local" attachment of the function to the event via the Active Form Behavior tab. Global attachment is not recommended for most projects, as discussed in the the "ISF Coding and Best Practices" section on page 2-104.
- **JavaScript Function**: The actual code for the function which includes the ISF code. The function signature must not include the "function" keyword (this is automatically added when the function is included in the generated service form), but function contents otherwise follow standard JavaScript rules. For example:

Write this code block in the JavaScript Function	To generate this code in the form:

`CER_Name_onChange() {`
 	
`}`
	`function CER_Name_onChange() {`
	
	`}`

Add the JavaScript function by clicking **Add new JavaScript**. Once a JavaScript function has been created, you may edit the function.

Adding Arguments to a JavaScript Function

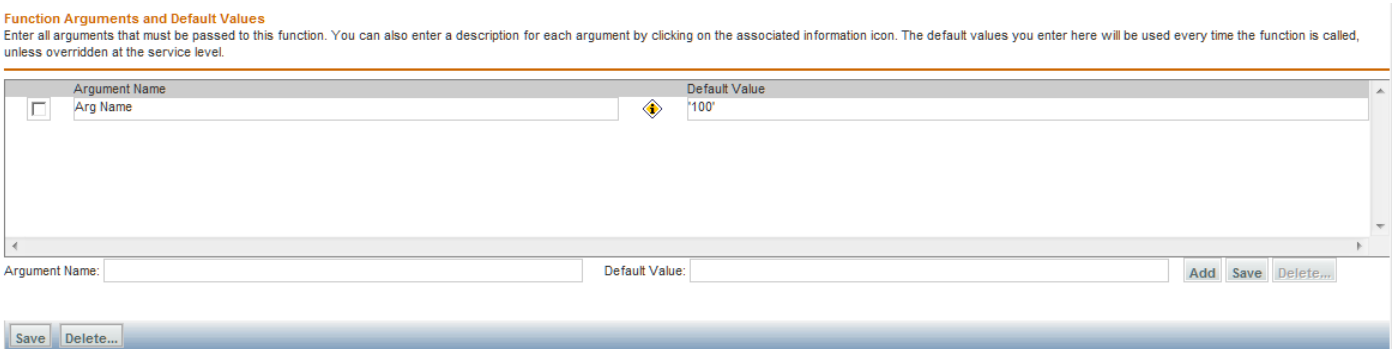
You may add arguments to the function by choosing the **Function Arguments** tab that is now available at the bottom on the "General" page for the function definition.

To add arguments to a JavaScript function:

Step 1 In the Argument Name field, enter a name for the argument (follow naming standards for JavaScript identifiers).

Step 2 In the Default Value field, enter a default value.

Step 3 Click **Add** to add the new function argument, as shown below.



Step 4 (Optional) Enter a description by clicking the Information icon button ( ). A Parameter Description popup window appears where you can enter a description and then click **Set Description**.

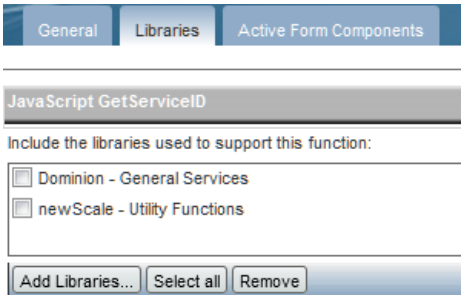
Step 5 In the bottom left of the window, click **Save**.

Follow the guidelines below in defining arguments:

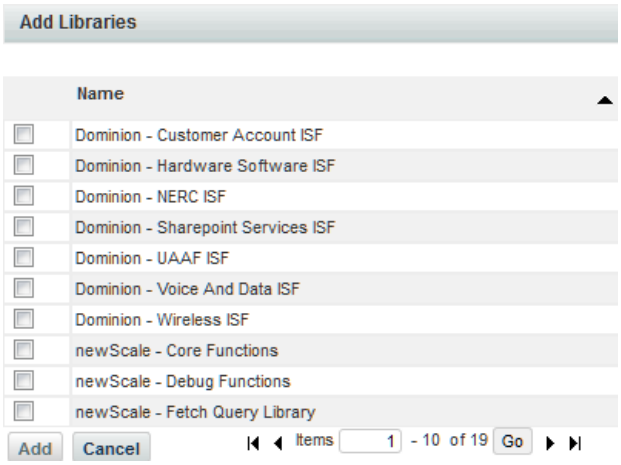
- Set an unused default value (that is. some default value that will never be overridden) for each of the JavaScript arguments.
  - For example, MyUniqueFunction(arg1, arg2) { .. }: The default value for these two arguments should be - arg1 = 'unused value 1', arg2 = 'unused value 2'.
- You MUST enter a default value for each argument. Otherwise, you will encounter errors.
- You must enclose default values intended to be strings in single quotes. Do not use double quotes and do not use two single quotes with no value in between. Otherwise, you will encounter errors.
- There is no way to mark an argument as a particular data type because JavaScript is type-less.
  - If the intent of an argument is a string value then the default dummy value (the value which will never be possible) can be enclosed in single quotes. For example: 'AAABBBCCCDDD'.
  - If the intent of an argument is a number value then the default dummy value (the value which will never be possible) can be some negative value. For example: –999999999.
- Always remember to edit the function arguments after adding the JavaScript function into an Active Form Component.

Associating Libraries with JavaScript Functions

If the function calls additional functions which reside in a library, use the Libraries tab of the JavaScripts option to specify the library.



- Click **Add Libraries** to bring up a window that allows you to choose one or more libraries.

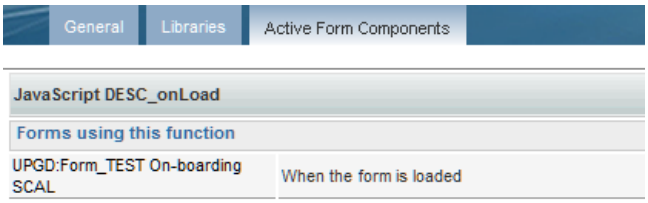


- Check the check boxes corresponding to the libraries you want to add and click **Add** to include the chosen items in the JavaScript function.

All libraries included in the function will appear on the Libraries page. You may delete one or more by checking the corresponding check box and clicking **Remove**.

Forms which Use a Particular JavaScript

The Active Form Components tab lists the forms to which the current JavaScript has been attached, and the triggering event, as shown below. Click on the form name to review the form definition.

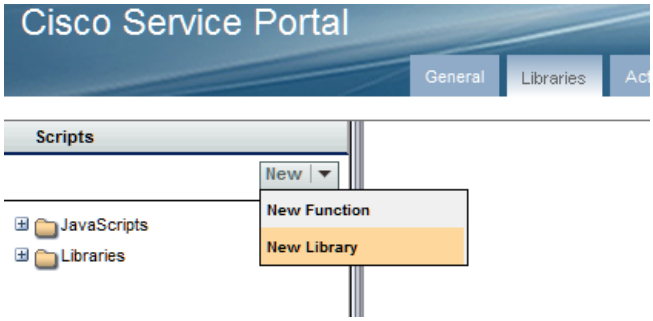


This cross-reference reflects only “local” attachment of functions. It does not include any JavaScripts which have been “globally” attached to a service by checking the “Add this script to the following events on all forms” check box on the General tab of the JavaScript function.

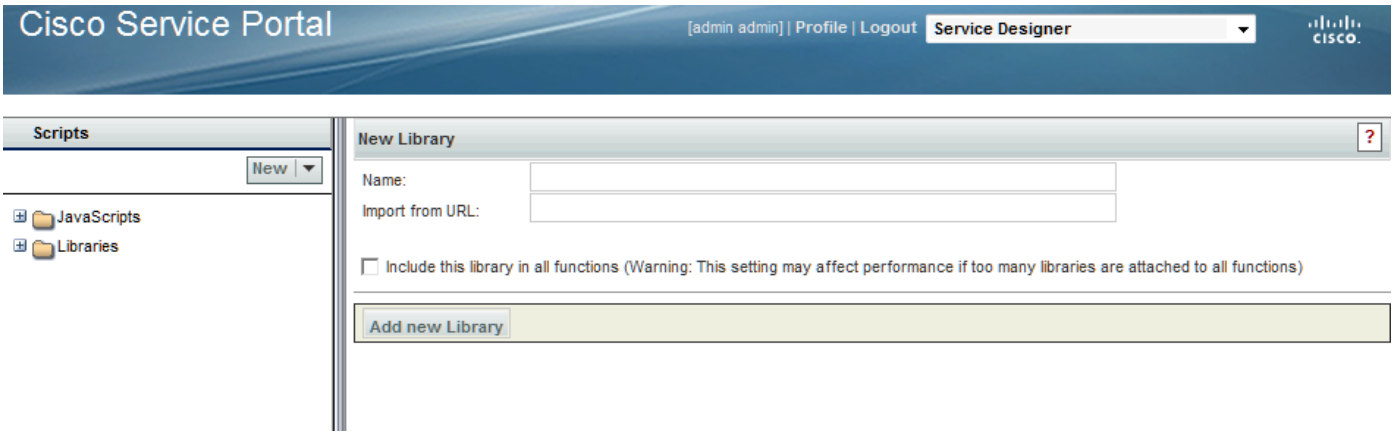
Libraries

A library is an ASCII text file which contains the code for one or more JavaScript functions.

Choose **New > New Library** under **Service Designer > Scripts** to create an entry for your library in the repository.



Calling this a “New Library” is a bit of a misnomer; it is actually a “New Reference” to a library which you have created (or will create) external to Service Portal.



- **Name**: Any name may be specified for the library, since this is a descriptive field. This name is used within Service Designer to refer to the library.
- **Import from URL**: The location of the library. The library must be located within the web deployment directory (designated as /RequestCenter/). By convention, ISF libraries are placed on the isfcode directory, directly beneath the root directory /RequestCenter.
- **Include this library in all functions**: This, too, is a slight misnomer, as the check box actually includes the library in all service forms, rather than in all “functions”. Checking this check box includes the library (by reference) in the service form, so that any functions defined in the library may be invoked. (A <script> tag for the library is generated into the service form.) Details on this and other options are discussed in the “ISF Coding and Best Practices” section on page 2-104.

Add the library to the repository by clicking **Add new Library**.

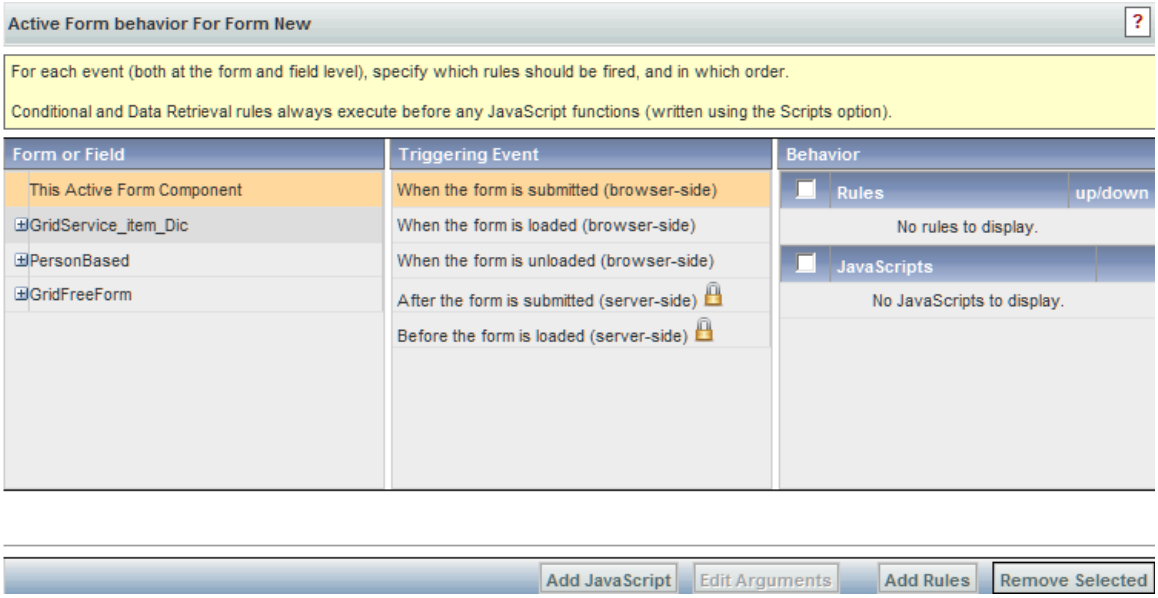
Adding JavaScript Functions to a Form

To add JavaScript functions to a form:

Step 1 Edit the form in the Active Form Components component of Service Designer.

Step 2 Click the **Active Form Behavior** tab.

Step 3 Click the first row in the "Form or Field" column, **This Active Form Component**. The Triggering Event column will list all form-level events.



Step 4 Choose the form-level triggering event to which the JavaScript functions are to be attached.

Available form-level triggering events for JavaScript functions are:

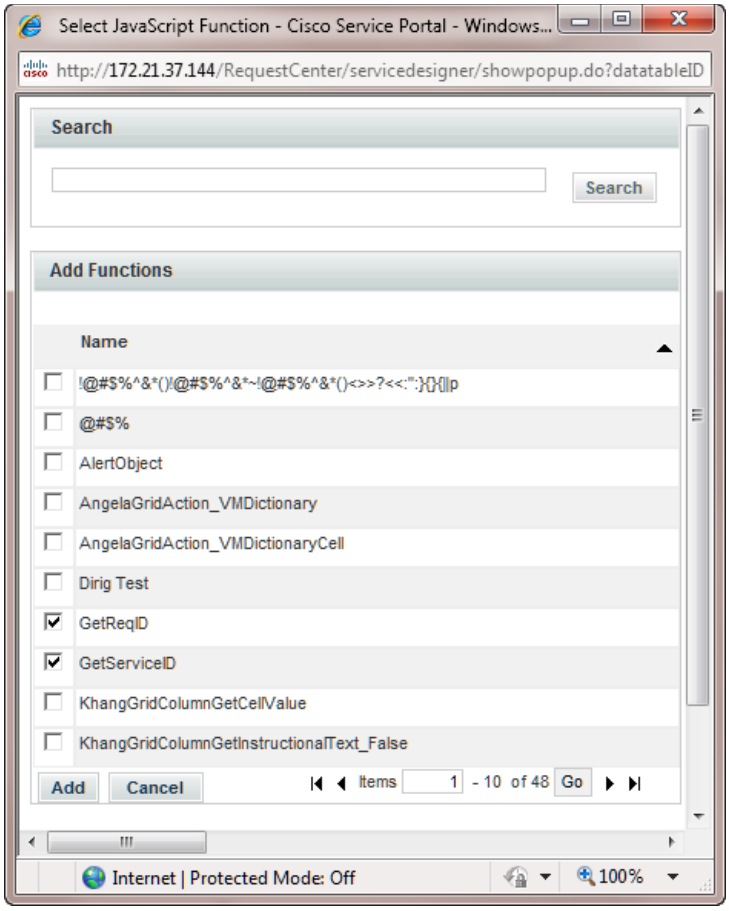
Form-Level Triggering Events	Description
When the form is submitted (browser-side)	The code is executed **after** the Submit or Update button is clicked, and **after** any validations specified in the form's Display (such as checking for numeric or mandatory data) but **before** the form is submitted to the server. This usually provides for a window to do extra validation, formatting, or any kind of processing before the data is sent to the server. The onSubmit function **must return a Boolean**, true if the submission can proceed, and false to stop it. This corresponds to the HTML event: form … onSubmit.
When the form is loaded (browser-side)	This is the **preferred** place to add code that populates fields in dictionaries, or that does some kind of form massaging before the form is rendered on the browser. This corresponds to the HTML event: body … onLoad.
When the form is unloaded (browser-side)	This event is called when the form is being closed. This event can be used to notify the user about data being lost if the window is closed. This corresponds to the HTML event: body … onUnload.

Note The server-side triggering events, "After the form is submitted (server-side)" and "Before the form is loaded (server-side)", are only available for form rules.

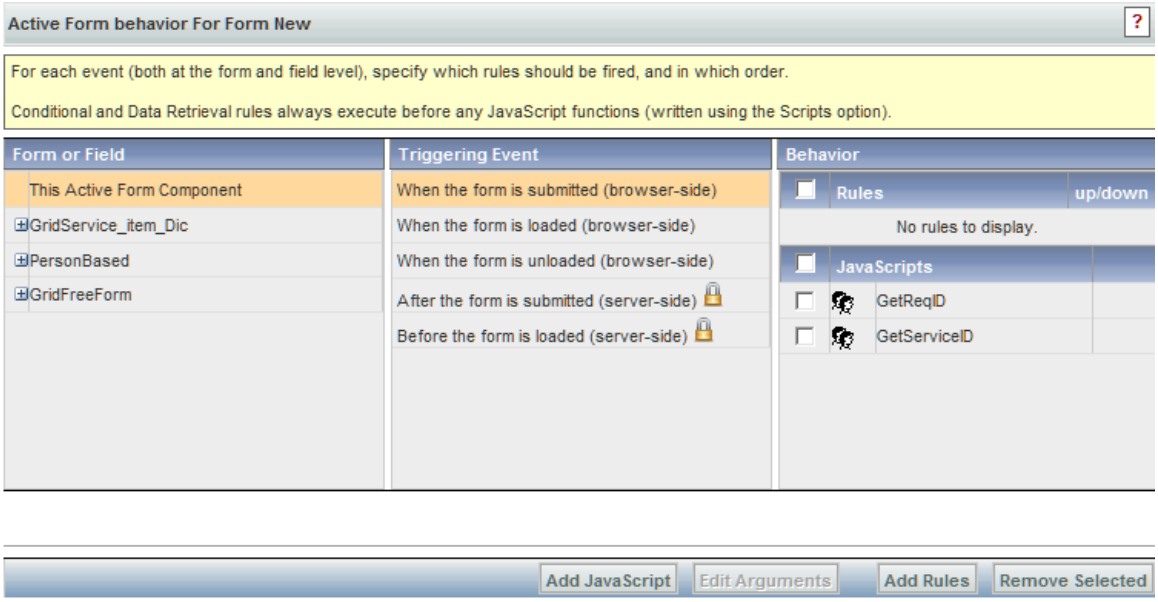
Step 5 Click **Add JavaScript**.

An Add Functions dialog box appears listing JavaScript functions previously defined with the Scripts option.

Step 6 Choose the JavaScript functions you want by checking their check boxes, and then click **Add**. You can use the Search box to search for JavaScript functions, if needed.



The functions appear on the Behavior column below the JavaScripts section, as shown below.



Note Although you can attach multiple JavaScript functions to the same event in a single form, this practice is best avoided, because the order in which the functions are specified (and executed) cannot be defined. Therefore, if you need functions to execute in a certain order, create one JavaScript function that contains all functions in the desired order.

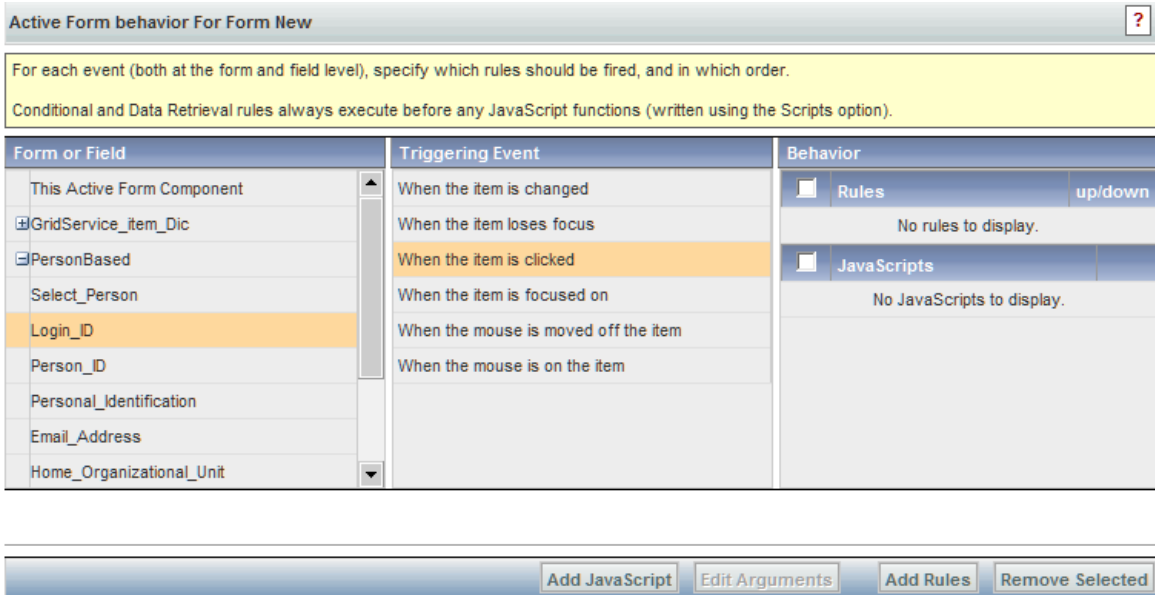
To add a JavaScript function to a field-level event:

Step 1 Edit the form in the Active Form Components component of Service Designer.

Step 2 Click the **Active Form Behavior** tab.

Step 3 In the "Form and Field" column, expand the dictionary node and choose the field to which the function is to be attached.

Step 4 In the Triggering Event column, choose the field-level triggering event that corresponds to the timing at which the function is to be executed.



Available field-level triggering events are:

Triggering Event Description	HTML Event Name
When the item is clicked	onClick
When the item is changed	onChange
When the item is focused on	onFocus
When the item loses focus	onBlur
When the mouse is on the item	onMouseOut
When the mouse is moved off the item	onMouseOver

The **onChange** event is best used to detect changes to Text/Single-select fields. The onChange event for a Person, Date or DateTime field is always triggered, even if the same Person is chosen from the "Select Person" popup window, or the same Date is chosen from the Calendar popup. Similarly, any typing in a Text field triggers the onChange event, whether the value in the field is actually changed or not.

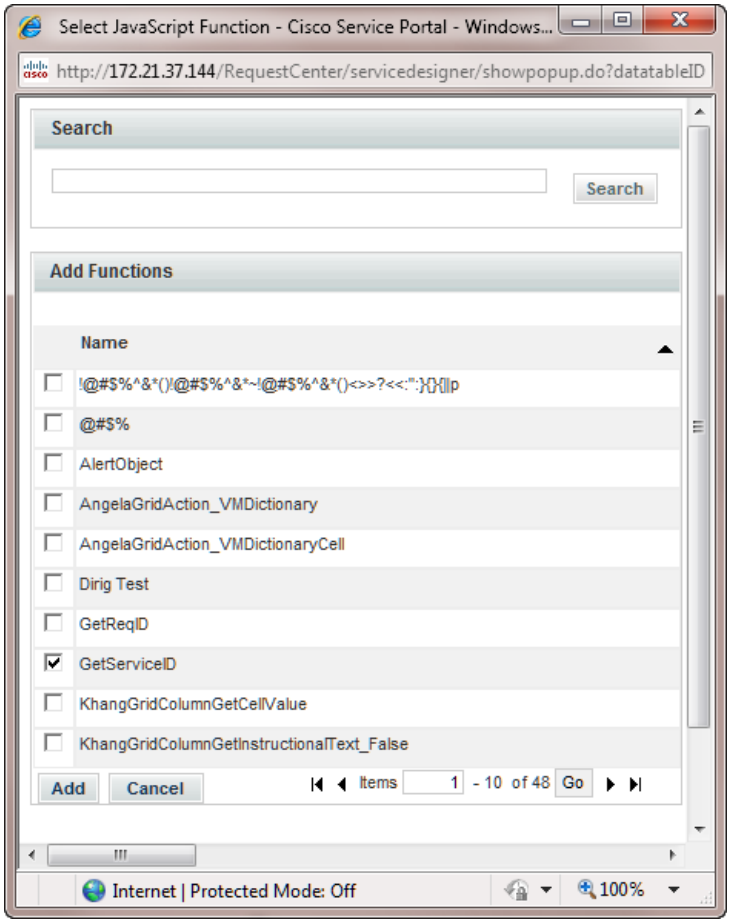
- Fill a field based on the value of the current field (for example, set a flag in a service based on the value of another field entered by the user).
- Choosing "Other" from a drop-down requires displaying an extra text box on the form.

The **onClick** event works best with Radio/Checkbox controls. The onClick event does not trigger for Person, Date and DateTime fields as text boxes for these field types are always read-only and selection is only possible through the popup window.

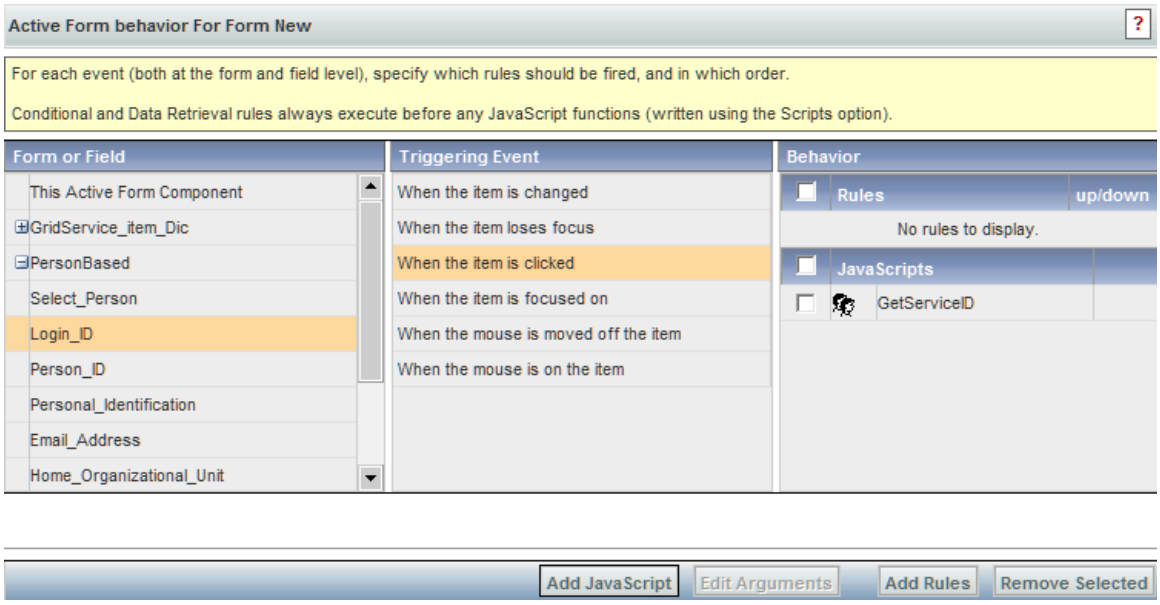
Step 5 Click **Add JavaScript**.

An Add Functions dialog box appears listing JavaScript functions previously defined with the Scripts option.

Step 6 Choose the JavaScript function you want by checking its check box, and then click **Add**. You can use the Search box to search for JavaScript functions, if needed.



The function appears in the Behavior column below the JavaScripts section, as shown below.

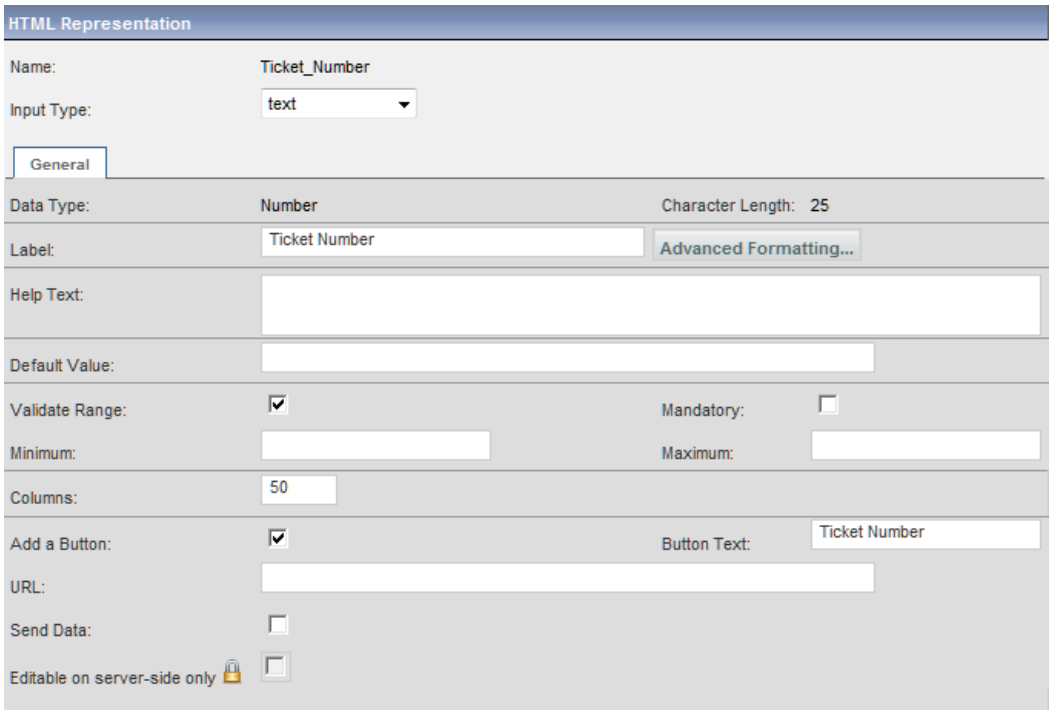


Note Although you can attach multiple JavaScript functions to the same event in a single form, this practice is best avoided, because the order in which the functions are specified (and executed) cannot be defined. Therefore, if you need functions to execute in a certain order, create one JavaScript function that contains all functions in the desired order.

Step 7 If required, you can edit arguments for a function by clicking on a JavaScript function in the Behavior column and then clicking **Edit Arguments**. (See the "Adding Arguments to a JavaScript Function" section on page 2-94 for more information.)

Associated Controls (Buttons and Links)

Any dictionary field may have an associated button, as specified in the "Add a Button" section of the Display Properties for the field:



Designers can specify a caption (text to appear on the button) and a URL. The URL can point to a JavaScript function or to an external page accessible elsewhere on the network.

When referring to an external web page, use the fully qualified URL.

- The URL may point to a JavaScript function, available in a library, or embed JavaScript code. Form data cannot be included in the parameters, if any, included in a function call. However, the function can include ISF.
- The Send Data option is applicable only to an external web page. This will POST a response to that page including all of the data on the current form.

The disadvantage of using an associated button, rather than placing code in an event handler, is that the user must use an extra keystroke (clicking on the button) to invoke the code. Consequently, buttons are best reserved for events that need to occur on demand, not in the course of regular form processing, or for extended dialog boxes or browsing of external sites.

Simply specifying a URL brings up the specified web page in a separate window. However, the window is not resizable and does not include scroll bars. Therefore, for some applications, it might be preferable to use JavaScript to display the web page in a window that you explicitly specify. Sample code is shown below:

```
javascript:window.open ("http://www.coder.com", "mywindow","status=1,toolbar=1,
scrollbars=1, width=500, resizable=1");
```

The JavaScript all has to be on one line (no carriage returns). See sample output (primitive, but you get the idea) below. The above line is copied and pasted from the URL entry for the button with the “URL with JavaScript” label.

ISF Coding and Best Practices

Using ISF adds an additional level of complexity to a service catalog project. This section outlines some methodological and technical tips that may be helpful in developing in this environment.

JavaScript/ISF Development Environment

Developing, testing, and debugging JavaScript and ISF efficiently may require installing additional tools on your client workstation or reconfiguring software previously installed.

JavaScript Debugging

JavaScript has a terrible pitfall: **It does not warn of errors**. Some syntax errors, errors that other languages catch, are ignored by most implementations of JavaScript. For instance, an extra parenthesis in a function call leads to **ALL the JavaScript code in the page failing to execute**. There are no warnings but an “Error in page” message may appear at the bottom left of your browser. By clicking on that message, you can sometimes see the line number of the page on which the error was detected.

In order to have more helpful error messages and debugging facilities, you need to install a JavaScript Debugger. The Microsoft Script Debugger is available as a free download from Microsoft (find it via a an internet search). Some web development environments, such as Visual Studio, include debuggers which could be used.

In order to use the debugger, you must reconfigure Internet Explorer to enable script debugging. You can access this setting via the **Tools > Internet Options > Advanced** option.

JavaScript-Aware Editor

It is helpful to have access to a text editor that provides syntax highlighting for JavaScript programs. Several such editors are available as freeware or trial versions. Some Java integrated development environments (IDE) also offer support for editing JavaScript files.

Maintaining Libraries on the Application Server

ISF developers need read-write access to the directory on the application server (typically, the isfcode directory underneath the RequestCenter.war web archive) on which JavaScript libraries reside. They will also need software for transferring the files between their workstation and the application server.

Library files are served from the application server. Therefore, page caching must be disabled, to allow revised versions of the libraries to be loaded.

Architecture/Storing ISF Scripts

Service Designer allows JavaScript functions to be stored in Scripts (within the repository) or Libraries (as external files on the file system).

Advantages of Using Libraries

Storing JavaScript in an external JavaScript (library) file has the following benefits:

- Code for many functions is maintained in one place rather than requiring the user to navigate to different screens, as would be the case if each event had a function attached in Script Manager.
- Maintaining the code in a file makes it easy to version control the source code.
- Maintaining the code in a text file makes it easy to do global search-replace and execute searches on the file.
- A JavaScript library is simply an ASCII file which resides on the application server. As such, you can use a powerful editor to maintain file contents.
- Since the same piece of code is referenced from one or more functions, changes need to be made only at one location and tested only once.

Prerequisites for Using Libraries

The library approach has the following prerequisites:

- The JavaScript libraries must reside on the web deployment directory (RequestCenter.war) on the application server. By convention the libraries are placed in a directory named “isfcode”.
- The programmer must therefore have access to the file system of the application server. This may entail creating additional logins on that server or providing additional client software (for example, a Remote Desktop service).
- The directory on which the ISF code resides needs to be set with read-write permissions for the ISF programmers.

A decision regarding where to store ISF scripts must be made before detailed design is attempted, as it will affect the design cost. It is significantly more efficient to use the library approach than to embed all ISF in the repository.

Structuring and Using Libraries

In principle, all client code could be included in one library. However, under some circumstances it might be advisable or required to divide the code into multiple libraries.

- One or more JavaScript libraries can be used, grouping the ISF functions so that those that are likely to be used in the same service or group of services are in one library, and those used by another set of services in another. This allows multiple sets of developers, potentially working on independent projects, to keep their work separate.
- If using multiple, independent libraries, the libraries should never be attached globally to the service forms (using the “Include this library in all functions” check box on the Library page of the Scripts option). Instead, the relevant library (libraries) should be included in a function attached to the onLoad event of a form (using the Libraries tab of the Scripts page).
- Similarly, the onLoad event to which the relevant library is attached should not be included in all forms (using the “Add this script … when the form is loaded (browser-side)” option on the Scripts page). Instead, it must be associated with the event using the Active Form Behavior tab for the form in the Active Form Components.
- There is a theoretical advantage in placing ISF functions likely to be used in the same service in one library and those functions that are rarely used, or used under well-defined circumstances, in another. The advantage would be a reduction of memory used (functions not required for a particular service are not loaded). However, since memory is relatively cheap and plentiful these days, and the amount used by ISF functions is minor compared to that used by other components, no practical advantages have been observed.

Recommended Naming and Coding Standards

If your organization has developed any JavaScript coding standards, you should adapt these. Although scripts written to support ISF are not generally too long or complex, applying naming and formatting standards helps programmers understand and read one another's code.

JavaScript is case-sensitive so any naming standard needs to specify case to be used for object names.

ISF Function Names

The ISF you write will be in JavaScript functions. If the functions apply to a specific dictionary or field within that dictionary, the function names should reflect this hierarchy. This makes it much easier to track function usage. Function names follow the convention:

- dictionaryName_event
- dictionaryName_fieldName_event

For example,

- RC_REQUESTEDBY_onLoad
- ST_HighProfile_onSubmit
- ST_UserLocation_FieldOffice_onChange
- SVC_Phone_PhoneType_onClick

In addition to dictionary- and field-specific functions, Cisco Advanced Services may create the following site-wide functions, which may be modified as required:

- rc_CommonService_onLoad

- rc_CommonService_onSubmit
- siteRC_REQUESTEDBY_onLoad
- siteRC_REQUESTEDFOR_onLoad

This naming convention makes it easier to understand how each script is used, and to find the appropriate place to create any new functionality. It also allows site-specific code to interface correctly with ISF code that may have been installed in conjunction with a standard service library.

Code Placement

It is recommended that ISF JavaScript functions be placed in an external JavaScript library. It is important to order the functions within that file, to facilitate finding a particular function.

Code Formatting

All code should be stored in ANSI text files without tabs and use 2-space indentation for clarity of reading and understanding the code. Lines should not be longer than 76 characters each.

ISF-Specific Best Practices

Be very careful changing the names of fields in a dictionary. If a field is referred to in a function, that function will stop working.

Every standard property and method in JavaScript starts with a lower case character, and the next word starts with an upper case character. For instance the property "**readOnly**" or the method "**onSubmit**" follow this convention. Although it is not enforced in ISF, following similar conventions is recommended. That way one does not have to think about it trying to remember how to spell a property name, either in the code or as part of the JavaScript language. The examples here use this convention.

Writing the Code

Service forms are generated automatically based on the specifications you enter interactively in Service Designer. Because you don't manually write HTML pages, the HTML tags for including libraries in a particular page or pages, or assigning a piece of JavaScript to a particular event embedded in a page, must also be generated for you. You use Scripts to specify how to include libraries in the generated HTML pages that contain service forms, and the Active Form Behavior tab to instruct the application which Scripts are event handlers for particular events. You must work outside of Service Portal to maintain code within the libraries.

Create a Library

As discussed previously, you want to place most custom ISF code in one or more JavaScript libraries. Use the Libraries option under **Service Designer > Scripts**, as described in the previous section.

Since the library is a text file, external to the application, you need a text editor to maintain the library contents. A JavaScript-aware editor or development environment is highly recommended.

Third-party JavaScript libraries may also be used in conjunction with ISF. Simply register the library using the Libraries option.

Copy the Library to the Application Server

To be accessible by the service form, the library must reside within the directory structure on the application server, mapped to the URL /RequestCenter. By convention, the library is placed on a directory named isfcode directly under the root. The physical location of the /RequestCenter site may vary according to how Service Portal is installed on your application server and the application server in use. Please consult your system administrator to determine the physical location on which the ISF libraries need to reside. You also need to ensure that ISF developers have read/write access to this directory and a tool for transferring files to the directory.

Include the Library in Service Forms

A reference to the library (implemented as an HTML script tag) must be included in the generated service form in order for functions in the library to be used in the service form. This reference may be generated in one of two ways.

- Use the Libraries tab of the JavaScripts node of the Scripts option to associate the library with a particular function. The library is then available in all forms which use the function.
- Use the "Include this library in all functions" check box on the Library option of Scripts to include the library in all service forms.

Loading the Library by Including it in a Function

This is the recommended approach to including a library reference in a service form. There should be a function which is invoked as an onLoad event. Libraries are then included in that function. One or more onLoad functions may be coded. Each may have a different set of libraries attached. In this way different teams of developers have control over which libraries are available to their service forms.

Loading the Library via the Library Check Box

The "Include this library in all functions…" check box on the Library page is a misnomer; it should be "Include this library in all forms…" since the library is only loaded once per form. Using this check box puts the <script> tag for the library at the beginning of the generated HTML page, ensuring that the library and its functions are present when the page-level ISF calls it.

This methodology is not recommended for complex projects.

Check Your Work

At this point you have a library (which possibly contains no code) and a specification to use the library for all service forms. You can verify your work by running a service form, looking at the source code, and searching for the name of your library. To view the source code of a service form, you cannot use the "View Source" option of your browser. Because the service form is displayed as a frame within an HTML page, its generated source is not displayed. Instead, move the mouse pointer to within the service form (not within a field) and use the right mouse button option to "View Source". A search shows the name of the library.

Write Custom JavaScript Functions

Ok, now it actually is time to write the code. Because of the difficulty of debugging in an HTML and JavaScript environment, it is highly recommended that your write, debug, and test one function at a time. You may, of course, edit the library file containing your JavaScript locally; but to test it, it must be copied back to the designated directory on the application server. When initially developing code, or if your access to the application server is limited, it may be easier to write the code within a Script and refactor when you are done, moving the function to the library and leaving only a wrapper function, which calls the library function, under the Scripts. It goes without saying that you should save often and keep backup copies in case you need to revert to a previous version of the code.

Attach the Function to the Appropriate Events

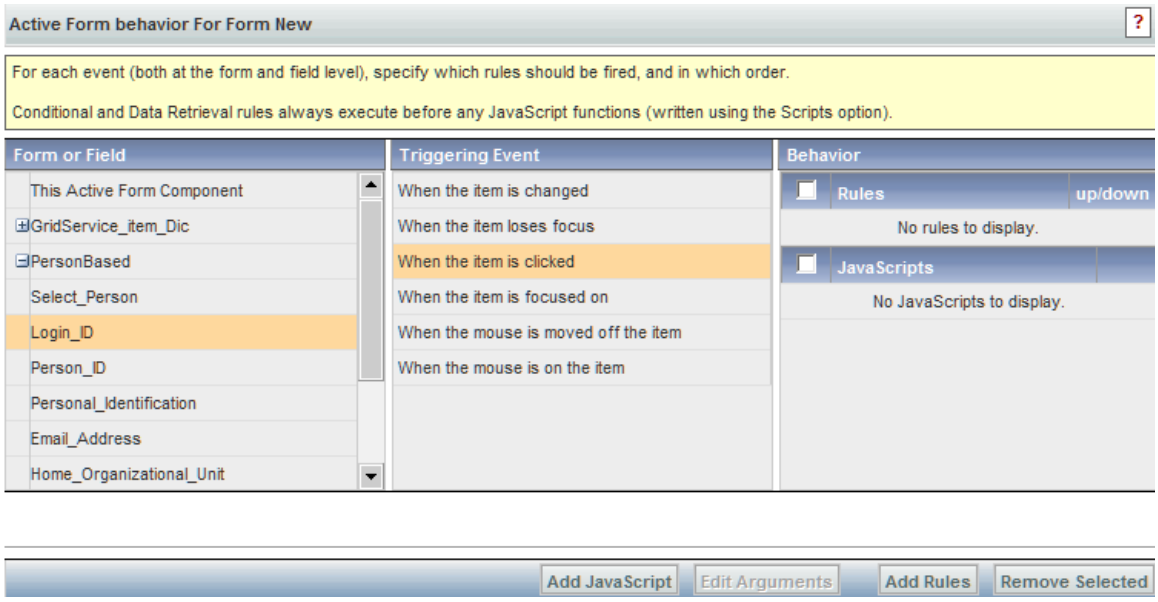
Once the code is written, it needs to be attached to an event in the form. Since the service form HTML page is generated dynamically, you must use Service Designer to do this. See the "Adding JavaScript Functions to a Form" section on page 2-97 for more information.

To attach a field-level event to a dictionary field:

- Write the function in the JavaScript library. Name it ***dictionaryName_fieldName_event***, for example, RC_REQUESTORLOCATION_LocationName_OnChange.
- Name the function with a site-specific code prefixed to the name of the function you previously included in the JavaScript library. Service Portal-provided functions (for example, those used in services included in the Service Portal libraries) use an "rc" prefix, so the code would appear as shown here:

```
rcRC_REQUESTORLOCATION_LocationName_onChange () {
 RC_REQUESTORLOCATION_LocationName_onChange();
}
```

- Use the Active Form Behavior tab of the Active Form Components option to attach the function to the appropriate field-level event. Choose the field and triggering event, then add the function by clicking **Add JavaScript**. (If the dictionary is used in multiple forms, the function must be attached to the dictionary field in every form. This is not recommended.)



Testing

ISF code is attached to an active form which can be reused in many services. However, it is not sufficient to test just one service. For example, you may have code that assumes that a field in a dictionary in another form is also present in the service; if that is not the case, your ISF code will fail with a JavaScript error. Therefore, you need to set up a testing matrix based on the different combinations of forms that can be used.

Especially in the initial phases of testing, it is useful to run multiple sessions of Service Portal simultaneously.

- Start Service Designer in one, with the option set to Scripts (to make changes to function code) or Active Form Components: Active Form Behavior (to change JavaScript attachments) displayed.
- Run My Services or Service Manager in the other, to test the service form in the moments for which rules or ISF have been defined.
- If you are testing code that resides in a library you will, of course need another window: to edit the library file and upload your changes to the application server.

If a JavaScript error is encountered, the JavaScript debugger is displayed. After you have fixed the error (by editing the function or library code) and dismissed the debugger, there is no reason to exit the current service form and start a new request. Simply refresh the page to cause the current service form to be reloaded with the new ISF code.

If the Browser Cache setting is enabled in the Administration Settings, changes made to the JavaScript libraries will not take effect until the browser cache has been deleted. Therefore in a development environment, it is best to disable browser caching. When modifications to JavaScript libraries are deployed to the production environment where browser caching might be enabled, application users will need to delete their browser cache. To prompt the application users to do so, follow the instructions in the *Cisco Service Portal Configuration Guide* to increment the browser cache version.

Design Guidelines

Create onLoad Code at the Granularity of the Dictionary

All JavaScript code can be thought of as specific to a dictionary so while authoring code ensure that each function is stand-alone and does not depend on any other function or dictionary. Hence, when a new dictionary is added or existing dictionaries are removed, the other code continues to function as before.

For example, assume that two dictionaries used in a particular service have code that must be executed in an onLoad event. Rather than writing one monolithic function, write two dictionary-specific functions, place them in a library, and call them from a wrapper function which is defined as a Script and attached as the onLoad event to the form:

```
Common_Service_onLoad () {
 IT_Dictionary1_onLoad();
 IT_Dictionary2_onLoad();
}
```

Create Service-Independent Code

Although this might not be possible in all cases, strive to create service-independent code. Then testing the code in one service suffices for all services in which that code is used.

The code in the previous example is not service-independent. It would fail if the Common_Service_onLoad() function were executed in a service that was missing one or both of the dictionaries. However, this can easily be modified:

```
Common_Service_onLoad () {
 if (serviceForm.ITDictionary1 !- undefined) {
    IT_Dictionary1_onLoad();
  }
  if (serviceForm.ITDictionary2 !- undefined) {
  IT_Dictionary2_onLoad();
}
```

Just as the above code tests for the presence of a dictionary in a service before attempting to apply dictionary-specific code, you may need to test for the presence of a particular field before attempting to apply field-specific code. It is best practice to use the dictionary in only one Active Form Component; however, service-specific rules may affect the dictionary's appearance. For example, displaying the supervisor information for the person requesting a service may only be required for those services that require supervisor approval. Therefore, code which attempts to manipulate the supervisor-related fields should be included in a code block such as:

```
FirstApprover_onLoad () {
  if (serviceForm.FirstApprover.SupervisorName !- undefined) {
//    code goes here;
  }
}
```

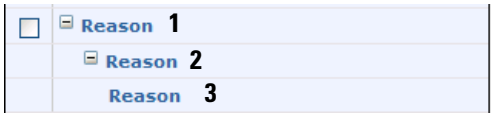
A field may be used in a form, but conditionally hidden by a rule or ISF code previously executed. In cases like these, code like the following might be more appropriate, and more robust:

```
  if (serviceForm.FirstApprover.SupervisorName !- undefined) {
    if (serviceForm.FirstApprover.SupervisorName.isVisible()) {
//    code goes here;
    }
  }
```

Best Practices for Using Active Form Components

Naming Conventions

Although it is perfectly legitimate, you should not give a dictionary, form, and field the same name. It results in a very confusing display, like:



1	Dictionary
2	Form
3	Field

Dictionaries should ideally use a prefix notation to denote their usage and perhaps the dictionary group in which they have been placed. A “Reason” dictionary sounds like it is fairly generic, used in many forms. Therefore, it makes sense to place it in a dictionary group named, for example, “Common”, and

to prefix the dictionary name with a code designating this dictionary group, for example, CMN_Reason. Dictionary groups could reflect the service group in which a service-specific dictionary is used, or perhaps the company department or division for which the dictionary and service have been developed.

An easily understood naming convention for forms is also needed. This is especially critical if you need to differentiate between multiple forms that include the same dictionary or group of dictionaries.

Form-Dictionary Relationship and Form Granularity

A key design decision is how many and which dictionaries to include in one form. Multiple dictionaries should be included in the same form if:

- These dictionaries will all be required in the same services.
- The order in which the dictionaries are presented is the same in all services.
- No additional dictionaries need to be displayed between the dictionaries in this form. On a service form, all dictionaries in one form component are displayed, in the order specified in that component; then, dictionaries in the next form component included in the service are displayed, in the order specified, and so on.

Simple Example

Assume that a group of services developed for the Facilities department have a chain of approvals consisting of up to three approvers, based on the customer's position or department. In this case the Facilities_Approval form should include three dictionaries: FirstApprover, SecondApprover, and ThirdApprover, as well as rules to determine how many of the three approvals actually need to be collected.

Not-So-Simple Example

Assume that most of the services developed for the Facilities department require the standard up-to-three approvers, but a rather expensive service requires an additional approver, at the VP level. You have two options:

One Form	Modify the Facilities_Approval form so that it includes the VPApprover dictionary in addition to the FirstApprover, SecondApprover, and ThirdApprover, and write an additional rule or rules to display and process the VPApprover dictionary only for that one service.
Multiple Forms	Create another form, Facilities_VP_Approval, which includes the VPApprover dictionary and duplicates the HTML, access control, and rules needed to process the FirstApprover, SecondApprover, and ThirdApprover dictionaries.

In most cases, the first solution (one slightly more complex form) should be preferred. You don't have to duplicate work to configure three dictionaries and their associated rules in more than one form. And, if any of those dictionaries or their rules needs to be changed, there is only one place to change them. You should only consider the second solution (two partly redundant forms) when including the additional dictionary would also cause extensive changes to the way the other dictionaries are displayed or processed.

Different Renderings for the Same Dictionary

What if the form's behavior or appearance needs to vary greatly, based on the services in which the form is used? Various scenarios are available. Only you, the service designer, can decide which scenario is preferable, based on the criteria outlined below.

For example, virtually every service needs to include that “Reason” dictionary. But sometimes the field label should read “Reason”, sometimes “Justification”, sometimes “Explanation”. Further, the help text associated with the field needs to be substantially different for each service or group of services. In this case, since the dictionary is simple and easily configured, a decision is easy: Create a separate form for each rendering of the dictionary, and include the appropriate form in the corresponding services.

But what if the field or fields that need to be customized on a service-by-service basis were part of a large dictionary with potentially complex rules? You still have the same two options outlined above: one form or multiple forms. However:

One Form	Create one form and customize its appearance using ISF functions.
Multiple Forms	Create a different form for each rendering of the dictionary.

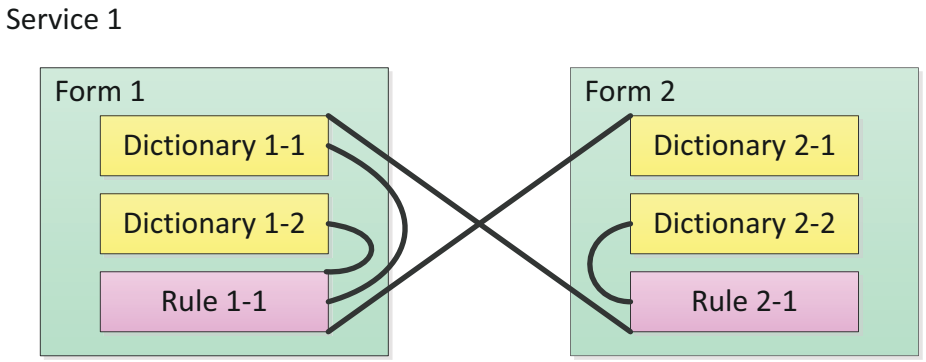
The One-Form option is now complicated by the fact that you cannot customize the dictionary's appearance using conditional rules—conditional rules only modify a field's appearance and behavior, not the contents of the field's label and help text. The Multiple-Forms option has the same drawbacks as before—you need to do the work upfront to create the forms, and the maintenance work to maintain rules in multiple places.

There are two additional options:

- Get the requirements changed. This is not likely nor a good idea, if it results in the users getting less information about how to supply information in requesting a service.
- Put the problematic field in a separate dictionary that is used in multiple forms, and keep one form for the rest of the dictionary. This is only possible if the field can be displayed before or after all the other fields, not in the middle.

Dictionaries, Forms, Rules and Services

All the discussion so far has assumed that dictionaries, forms, and rules are structured like this:



Key takeaways from this diagram are:

- A form typically consists of a set of dictionaries and a set of rules.

- The rules may affect not only those dictionaries (and their fields) in the same form but dictionaries in other forms.
- It is very important to test the rules thoroughly in the context of a particular service, to ensure that all referenced dictionaries are in forms included in that service. The Best Practices Reports include a "Dictionaries in Services" report that will facilitate doing this testing and doing an impact analysis to assess potential effects of any proposed changes to dictionaries or rules.

It is also perfectly reasonable to structure Form 2 so that it contains only rules to be applied to a particular service. Many services could include only Form 1, but only the service with extra requirements includes Form 2. This greatly simplified (or potentially eliminates) conditions that would be needed in the rules in Form 2, to ensure they only fire in the appropriate services.

This also eliminates having to write a conditional rule that tests for the service name—a textual element that is subject to change. You would merely have to include the rules-only form in the service affected, sequencing it after the form whose dictionaries are affected. Rules are executed in the order in which the forms are included in the service, and then in the order in which the rules are arranged within the form.

Loosely Coupled vs. Tightly Coupled Rules and Dictionaries

When rules and the dictionaries they affect and refer to are in the same AFC, the rules and dictionaries are said to be "tightly coupled". The example above, with a rule affecting the appearance of a dictionary in another form, shows "loosely coupled" rules and dictionaries.

In general, a rule can affect a dictionary in any form, provided both forms are included in the same service. There is, however, one significant limitation on loosely coupling rules and dictionaries—a rule defined in one form cannot be triggered by an field-level event attached to a field in a dictionary in another form. Therefore, loosely coupled rules typically need to be triggered by a form-level event, when the form is loaded or submitted.

Rules, ISF and the Requisition Life Cycle

When a request is submitted, all of the definitions for the dictionaries, HTML representation, access control, and task plan for that service are stored (in a compressed format) with the request data. This ensures that a request can be processed throughout the authorization and delivery moments without any subsequent changes to the form's definition affecting the behavior or appearance of the service form for the in-flight request.

If you change a dictionary, all forms using that dictionary automatically inherit the change and all services which use those forms also inherit the change. Any request for one of the changed services that has been saved but not submitted are marked as obsolete. The user will need to cancel the request or remove the service, readd it, and fill in the service form data. Submitted requests are not affected by changes to dictionaries or active form components.

However, rules and ISF functions (whether in a Script or an external library) are always loaded dynamically when a service form is displayed in My Services or Service Manager. Therefore, you must ensure that a current version of the rule/code does not refer to a field no longer on the form or, conversely, that a field on previous versions of the form, but not longer used, can have an associated function. This flexibility is easily accomplished by always checking for the existence of a dictionary or field before attempting to manipulate it via ISF, as discussed previously. If desired, a different version of a library could be attached to the newer versions of the form so that, for example, the names of functions would not need to change, but their contents could be updated to comply with revised requirements.

Changing a Dictionary or Active Form Component

Geek Alert: Dictionaries and Active Form Components show true “inheritance”. As noted in the “Rules, ISF and the Requisition Life Cycle” section above, if you change the definition of a field within a dictionary, all active form components will reflect that change. Similarly, if you change any aspect of the active form component's definition, all service definitions that use that form will reflect that change.

You cannot delete/change a dictionary or dictionary field to which a conditional rule is associated, or that triggers a data retrieval rule; you must remove the associations first. This check is not performed for JavaScript functions.

Each time the form is changed, the application automatically updates the version number for all service definitions which incorporate that form. You may see a brief delay in saving such changes if the form is used within many services. This may also affect any Requisition API (RAPI) programs implemented for ordering the service, since the RAPI SubmitRequest operations requires the version number of the service to be specified.

Coding SQL Entry Data Retrieval Rules

If you are not familiar with SQL, it may be a bit intimidating to code a SQL Entry data retrieval rule. A “trick” to help familiarize yourself with SQL is to start by configuring a simpler version of the rule as a Database Table Lookup. After you have saved the rule, the rule summary displays the generated SQL. You can copy the SQL statement, paste it into a SQL Entry data retrieval rule, and modify it as required.

Using Customer and Initiator Lightweight Namespaces

The #Customer# and #Initiator# lightweight namespaces are automatically supplied as the default values for fields in the Customer_Information and Initiator_Information dictionaries used in the Customer-Initiator form component. However, these namespaces could be used as default values for any form component.

The design of dictionaries to hold information about customer location could use this technique. Fields about the customer’s location could be included in the Customer_Information dictionary. However, such fields may be required on very few services—only those where a service must be delivered to the customer’s physical location. A more flexible design may be to configure a separate dictionary (and form component) for the customer location, and include this form component only in services where customer location is relevant.

Multiple namespaces can be specified as a default value. For example, the expression #Customer.FirstName# #Customer.LastName# concatenates the customer’s first name and last name, separated by one space. This expression duplicates the appearance of the first field of a person-based dictionary.

Note The use of grid dictionary fields for lightweight namespaces is not currently supported.

Putting It Together

Overview

In a complex service catalog you can easily have hundreds of instances where active form rules or ISF functions are used to enhance the service form's usability and interactivity.

1. Perform a use case analysis in order to determine what rules are required and when they will fire.
2. Perform a detailed design analysis to determine what dictionaries and fields are needed to support the specified use cases; how the dictionaries should be combined into forms; and what tools you will need to implement the requirements.
3. Define the dictionaries and forms. Specify the display and access control properties of the forms.
4. Write active form rules and sequence these to meet the detail design criteria. If required, write JavaScript functions and libraries; attach the functions to the appropriate HTML event or events.
5. Define the services which use the reusable form components previously defined.
6. Test and make fixes as required.

This section includes some examples of both rules and ISF code integrated into a service form, and the sorts of implementation decisions made to meet the requirements. Hopefully, these examples illustrate some common design patterns, such as:

- Adjust the appearance or behavior of dictionaries based on a specific task
- Show/hide dictionaries or fields when a service form is loaded or when the user changes the value in another field
- Manager lookup on change for Person fields
- Drill-Downs
- Ensure that Select Lists are populated during the delivery moment
- Hide dictionaries and clear fields if their value is not relevant to the current context

Use Case Analysis

Review user requirements to determine when a service form's default behavior and appearance need to be supplemented by rules or ISF. While you may document these requirements discursively, a better format might be a table which explicitly defines the behavior and its triggering events. An example is shown below.

Service Name	Use Case Description	Moment
All	If a person is high profile, display his status as read-only; if not, hide the status	Ordering

Where:

Service Name is either "all" or a list of the services which are affected by the requirement.

Use Case Description specifies, in language accessible to business users, the desired behavior of the service form.

Moment is either “all” or one or more of the moments that occur during the fulfillment of a requisition. A list of valid moments is given in the “Service Form Framework” section on page 2-4.

Try to collect all the use cases for the current project, so you can estimate the scope of the work involved.

Detailed Design

In the detailed design phase the programmer must review the use cases previously defined and specify, at a high level, the rules or JavaScript components that need to be written, and the triggering events for these rules/functions. The design incorporates a detailed specification of the forms, dictionaries and fields involved. In particular, the analysis should cover:

- Will you need to use ISF to supplement the form rules? If so, ensure that your development environment is set up to support ISF (JavaScript) development, testing and debugging, and that personnel are available with the requisite skill set.
- Will you need to use data retrieval rules or SQL option lists? If so, ensure that the development environment includes a means to test and debug SQL queries; that datasources are defined to allow you to access the desired data; and that personnel are available with knowledge of the structure of the source data and the requisite skill set.

Scenario #1: Dynamically Adjusting Form Appearance and Behavior

Functional Requirements

In requesting a database to be created, the user must choose whether the database server type is Oracle, SQLServer, or some “Other” database. If some other, nonenterprise standard database is chosen, additional information must be gathered from the user; otherwise, the additional fields are hidden.

Scenario:

Is the current value of the DatabaseType field in the Database dictionary equal to “Other”?	If so, display the Description field in the same dictionary, and require the user to enter additional information.

Dictionary/Form Design

Define a dictionary called, for example, NewDatabase.

- The dictionary includes a field named DatabaseType, rendered as a radio button which allows the user to designate whether the database is Oracle, SQLServer, or Other.
- If the user selects “Other”, the type of database must be provided and is mandatory. Otherwise, this field is hidden.

Detailed Rules Design

This use case can be implemented entirely through the use of conditional rules.

The AIT_DATABASE.DatabaseServer field needs rules which fires when the field's value is changed.

- If the value which the user selects is 'Other', a rule must display the “Other”-dependent dictionary fields and ensure that all such fields are mandatory.

- If the value which the user selects is not 'Other', a rule must ensure that the “Other”-dependent fields are not visible.

Unfortunately, this release of Service Portal does not include if/then/else logic in the rules, so you will need two rules to implement this design.

Conditional Rule Implementation

Define two conditional rules as follows:

Rule Summary - DatabaseStandard	
Rule Name	DatabaseStandard
Description	
Conditions	NewDatabase.DatabaseType is not equal to Other
Actions	Hide NewDatabase.OtherDatabaseType

Rule Summary - DatabaseOther	
Rule Name	DatabaseOther
Description	
Conditions	NewDatabase.DatabaseType is equal to Other
Actions	Show NewDatabase.OtherDatabaseType Make Mandatory NewDatabase.OtherDatabaseType

The Triggering Events

When should this rule be applied? In more technical terms, what is the “triggering event”, during the course of a user entering data in the service form, when this rule should “fire”?

It seems obvious that the rule needs to be executed when the user selects a value from the DatabaseServer radio button. So, use the Active Form Behavior tab for this form to associate these two rules with the “When the field changes” event. In this case, it really doesn't matter what order the rules are applied in—either one or the other will fire, but not both.

Build the Service Definition

To test this scenario, you need to complete the definition of the service which contains the active form you have just defined. You could initially create a service containing just that form, for a quick-and-dirty test, but this is clearly just a first step. Forms and their rules can interactive with other forms and their rules, so the best test is the most realistic.

Test

It's easiest to test this scenario by using several browser windows. Keep Service Designer open in one, with the Active Form Components option displayed. Then, start a new session, log in as a My Services user who has permission to order the service. See what happens.

Testing Follow up and Results

The rule as previously defined should work correctly. However, the implementation is incomplete, as the behavior is only triggered when the customer changes (or initially selects) the operating system. If the form is saved and reviewed, or submitted and displayed in a subsequent system moment when the dictionary is editable, the saved value of the DatabaseServer field must be used to adjust the appearance of the form. Therefore, an additional triggering event is required, to fire when the form is loaded.

Scenario #2: Manipulating Customer and Initiator Information

Functional Requirements

Usage needs to take into account two different scenarios that affect the one Customer dictionary and the one Customer-Initiator Form:

Scenario:

If the service is delivered electronically …	Display the standard set of fields required for collecting Customer information.
If the service needs to be delivered to a specific physical location …	In addition to the standard fields, display fields pertaining to the Customer's location, as stored in the Customer profile, but allow him/her to specify an alternate service location, for example, if the location on file is out of date or the customer is temporarily at a different location.

Dictionary Design

The design can be implemented using three dictionaries:

- Design the Customer dictionary so that it includes all of the fields that must be available in both types of services—those fields that are always required and those that are required only when the requestor must specify or confirm the location where the service must be delivered.
- Design a Perform Work dictionary that will only be displayed for services to be delivered in person. The form has one field, which asks if the work is to be performed at the customer's location or a different (service) location.
- Design a Service Location dictionary, to be filled in, by default, with the location information from the Customer's profile, but which can be overridden if the requestor indicates that this is different than the location in the profile.

Form Design

The Customer and Initiator dictionaries are typically read-only in all moments. That is, data is displayed from the person's profile and cannot be changed by either the customer or any task performer.

There are at least three ways to make all fields in a dictionary read-only:

- Use the Access Control tab for the form to specify that the dictionary is viewable (not editable) for appropriate moments and participants. This control cannot be overridden by either rules or ISF—a viewable dictionary can be hidden, by it can't be made editable. And field values display as boilerplate, rather than enclosed in input fields. (Q: Would lightweight namespaces work?)

- Make the dictionary editable in the Access Control tab, but define the default fields as having an HTML input type of read-only and the location-related fields as having an input type of hidden. Both hidden and read-only fields are supplied values from an associated lightweight namespace.
- Make the dictionary editable in the Access Control tab, assign appropriate input types to the standard fields, but create a rule, applied when the form is loaded, to make All Fields in the dictionary read-only. This rule would not affect any hidden fields.

Option #2 makes the most sense in this case. Fields could potentially be made writeable, for example, if users are allowed to update out-of-date data from their person profile. And there is no extra rule to keep track off. That takes care of the Customer-Initiator form.

You also need a form in which the PerformWork and ServiceLocation dictionaries are used. Call this form ServiceLocation. (Q: Naming conventions?) The field in the PerformWork dictionary can be implemented as a check box? (Is work performed at the Customer Site?) The ServiceLocation dictionary would have fields corresponding to the person fields included in the Customer dictionary.

Detailed Rules Design

Now you need a rule for the services where a service location is required. The rule is needs to:

- Copy the default location values from the Customer dictionary to the ServiceLocation dictionary. This can be done in an onLoad event.
- If the user says that a different service location is needed, make the ServiceLocation fields writeable. This needs to be done in an onChange event for the PerformWork field. (Q: Nomenclature, again).

The rules are included in the ServiceLocation form.

Conditional Rule Implementation

Does anybody else out there remember COBOL? Most of the time, coding in COBOL was painfully verbose, but it had one really neat command: COPY CORR(esponding). The COPY CORR command copied all values, by name, from one structure to values with corresponding names in a second structure. It would be great to COPY CORR Dictionary1 TO Dictionary2, but that is not possible. So, the first rule (ServiceLocation_onLoad) should be applied in the ordering moment, and Copy Value for all location fields.

The second rule (PerformWork_onChange) is straightforward, reminiscent of the rule written in the initial scenario.

The Triggering Events

The rule is triggered.

Build the Service Definitions and Test

You'll need two services to test this scenario—one that doesn't include the ServiceLocation form, and one that does.

Scenario #3: Securing Sensitive Data

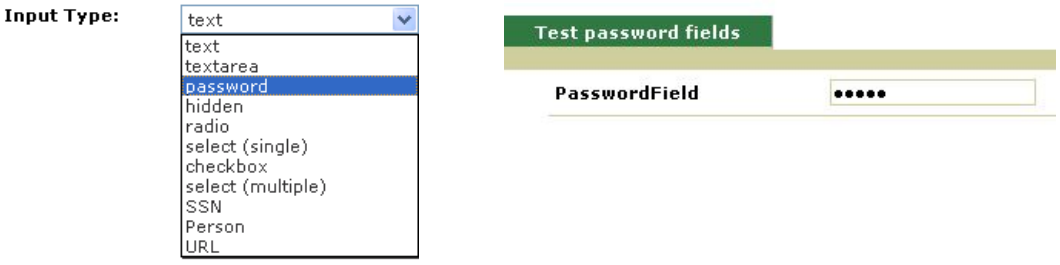
Functional Requirements

The service requires the user to specify a Social Security Number, credit card information, or other sensitive data that should be available only to people who “need to know”, not to every authorizer or task performer involved in fulfilling the service. The data needs to be protected from attempted hacks as well.

Approach 1 – Hide the Dictionary and Fields

By default, when a dictionary's display property is set to “none” in Service Designer, the dictionary and the values for any fields previously entered are not part of the generated service form seen by users. (This setting may be overridden for backward compatibility with behavior of Request Center versions prior to 2007. Be sure to check with your Request Center Administrator to verify the setting for dictionary.permission.none.show newScale property.) Therefore, the data in the field would not be visible, even to users savvy enough to view the page source from the browser.

To allow the field value to be initially provided by the user, the field must (obviously) be visible on the form, and the dictionary display setting set to Read/Write. The field's HTML representation can be set to “password”. The field value will then be displayed as a series of asterisks.



So, the value is protected from people doing shoulder-surfing. If you view the source of the page, however, the value of the field is visible. The value of the field is not accessible to ISF’s getValue() function—“undefined” is returned. The value of the field is available to the DOM, that is, via JavaScript methods such as:

```
document.getElementById('Dictionary.PasswordField').value.
```

The value of the field is also accessible to Service Link.

In any case, the value of the field is stored in clear text.

Approach 2 – Use Encryption

Encryption can be used in conjunction with some of the practices above to better secure sensitive data. Instead of (or in addition to) using a password field, use JavaScript functions to encrypt the sensitive data before you save it and decrypt it before displaying it on the form. An open-source algorithm on the web called “tiny encryption algorithm” could be used.

```
// Algorithm: David Wheeler & Roger Needham, Cambridge University Computer Lab
//    http://www.cl.cam.ac.uk/ftp/papers/djw-rmn/djw-rmn-tea.html (1994)
//    http://www.cl.cam.ac.uk/ftp/users/djw3/xtea.ps (1997)
//
// JavaScript implementation: Chris Veness, Movable Type Ltd
```

If you define the function to encrypt/decrypt code in a Script, it would be visible in the form if the user did a View Source. However, if you include the function in a library, it would not be visible to users who attempt to view the source, and not subject to reverse engineering.

Scenario #4: Computing a Value in a Form

Functional Requirements

The service requires the user to specify a "Quantity" and "Price" for an item to be ordered. The service should compute the "Extended Price", and show this value on the service form.

Dictionary/Form Design Requirements

A dictionary, call it SVC_PRICE, needs to be created, containing three Number fields. The fields may have decimal precision or not, according to detailed requirements. The dictionary is included in the service form and is writeable for the customer in the ordering moment. All fields are rendered as text fields.

ISF Detailed Design

The Total field needs to be read-only, since users are not allowed to enter a value—it must be computed. The computation needs to take place when the user changes either the Price or the Quantity.

This task requires three custom events:

- The SVC_PRICE_onLoad event sets the ExtendedPrice field to be read-only.
- The SVC_PRICE_Quantity_onChange event computes the ExtendedPrice.
- The SVC_PRICE_Price_onChange event also computes the ExtendedPrice.

JavaScript Code and Events

The first task is better accomplished when the form loads the first time. To do this, create a function in Script Manager called SVC_PRICE_onLoad:

```
SVC_PRICE_onLoad ()
{
  serviceForm.SVC_PRICE.ExtendedPrice.setReadOnly(true);
}
```

This code is associated to the "When the form is loaded (browser-side)" event in the Behavior tab.

The second task is to compute the ExtendedPrice based on the Quantity and Price. Use the "When the item is changed" event for both Quantity and Price. The code needs to verify that both fields have valid values, and then compute the ExtendedPrice. The function is called SVCPRICE_Price_onChange.

```
SVC_PRICE_Price_onChange ()
{
  serviceForm.SVC_PRICE.Total.setReadOnly(true);

  var Price = serviceForm.SVC_PRICE.Price.getValue()[0];
  var Quantity = serviceForm.SVC_PRICE.Quantity.getValue()[0];

/* Blank out current value (if any) of ExtendedPrice */
  serviceForm.SVC_PRICE.Total.setValue(['']);
```

```
/* Check is required, since check for Numeric data happens only on Submit. */
  if (isNaN (Price))
  {
    alert ('Price is not a number');
    serviceForm.SVC_PRICE.Price.setFocus(true);
    return;
  }

  if (isNaN (Quantity))
  {
    alert ('Quantity is not a number');
    serviceForm.SVC_PRICE.Quantity.setFocus(true);
    return;
  }

  var Total = Price * Quantity;
  serviceForm.SVC_PRICE.Total.setValue([Total]);
}
```

Refactored JavaScript Code

The code above, defined as a Script in Service Designer, could be attached to the onChange event for two different fields: the Price and the Quantity. In fact, this may be an efficient way to initially test the code. However, this approach, with a function name that doesn't reflect this usage and that doesn't use a library, is harder to maintain in the long run. Therefore, the following refactoring is recommended:

- Edit the function code, renaming the function something generic, like “ComputeExtendedPrice”, and extract the code from Scripts, placing it into a custom library for your application.
- Be sure the custom library is defined in **Scripts > Libraries** and that it is included in the service form when this function is required.
- Create two Scripts, which look like the following:

```
SVC_PRICE_Price_onChange ()
{
  ComputeExtendedPrice();
}

SVC_PRICE_Quantity_onChange ()
{
  ComputeExtendedPrice();
}
```

- Attach these functions to the onChange events of the Price and Quantity fields, respectively.
- Remember to upload the revised library to the application server.

Formatting

Requirements

Create a JavaScript function to format two fields in a form: A social security number and a phone number. The SSN must verify that there are 9 digits, and is formatted like “999-99-9999” Any formatting done by the user is ignored. The phone number must have 10 digits and it is formatted like (999) 999-9999.

JavaScript

Create two functions, one called **formatSSN** and the other called **formatPhoneNo**. Put both functions in a file called "isfprimerlib.js". As documented in the "Maintaining Libraries on the Application Server" section on page 2-105, this file may reside on any directory on the application server beneath the RequestCenter.war directory; by convention, ISF libraries are placed on a directory named "isfcode".

Create a library reference to your JavaScript file in Script Manager. Be sure to specify the library in the Libraries tab for a JavaScript function that are attached to your service.

The resulting code is:

```
function getOnlyDigits (inValue)
{
  var outValue = '';
  var aChar;

  for (i=0; i < inValue.length; i++)
  {
    aChar = inValue.charAt (i);
    if ('0' <= aChar && aChar <= '9')
    {
      outValue = outValue + aChar;
    }
  }
  return outValue;
}

function testValueLength (inValue, inLen, obField, fieldName)
{
  if ((inValue.length > inLen) || (inValue.length < inLen))
  {
    alert (fieldName + ' must have ' + inLen + ' digits and it has ' + inValue.length);
    eval('serviceForm.'+obField).setFocus(true);
    return false;
  }
  return true;
}

function formatSSN (obField)
{
  var SSNString = getOnlyDigits (eval('serviceForm.'+obField).getValue()[0]);
  if (testValueLength (SSNString, 9, obField, 'SSN'))
  {
     eval('serviceForm.'+obField).setValue([SSNString.slice (0,3) +
       '-' + SSNString.slice (3,5) + '-' + SSNString.slice (5)]);
  }
}

function formatPhoneNo (obField)
{
  var phoneString = getOnlyDigits (eval('serviceForm.'+obField).getValue()[0]);
  if (testValueLength (phoneString, 10, obField, 'Phone Number'))
  {
    eval('serviceForm.'+obField).setValue(['(' + phoneString.slice (0,3) + ') '
      + phoneString.slice (3,6) + '-' + phoneString.slice (6)]);
  }
}
```

For the field SSN create a function called **Customer_SSN_onChange** that calls **formatSSN**:

```
Customer_SSN_onChange ()
{
  formatSSN('Customer.SSN');
}
```

For the PhoneNo field, create another function called **Customer_PhoneNo_onChange**, that calls formatPhoneNo when the value in the field changes. Just to show an alternative implementation, these functions pass the name of the field and acts “by-reference” rather than “by-value”.

```
Customer_PhoneNo_onChange ()
{
  formatPhoneNo ('Customer.PhoneNo');
}
```

Both functions are called when the value in its respective field changes.

Server-Side Associated Controls

This section explains how to transfer service form data to and from an outside server through an HTTP request. The purpose of this functionality is to allow the Service Designer to use Web widgets that reside on Web servers separate from the application server. This section only deals with the actual transfer and handling of the service form data.

The service form data is sent to the outside Web widget via an HTTP form post. The data is passed in the **wddxdataform** form in the WDDXData variable as a WDDX packet. WDDX is an XML schema for storing data structures in a serialized packet. This allows the data to be passed into or out of Service Portal independent of the programming language used. You can use an HTTP request to a Service Portal page to place the resulting data set back into the service form.

CHAPTER 3

Lifecycle Center

- Overview, page 3-1
- Service Items and Service Item Manager, page 3-4
- Configuring Service Item Dictionaries, page 3-22
- Configuring Active Form Components, page 3-27
- Configuring the Delivery Plan, page 3-29
- An End User's View of Service Items, page 3-44
- Importing Service Items and Standards, page 3-47
- Best Practices, page 3-66
- Administration, page 3-68
- VMware Adapter Error Messages, page 3-73

Overview

A service item is a tangible result delivered in response to a service request. It may be physical, such as a physical hardware device (for example, a cellphone or server). Or it may also be virtual, such as software (for example, an application or a login ID), a software license, or a VMware virtual machine. A given service request may deliver multiple service items, particularly in the case of an employee on-boarding service, which may provide a cellphone, a laptop, a network login, email access, and multiple software applications.

Lifecycle Center is a module of Service Portal that allows Request Center users to manage the service items that have been ordered from the Request Center service catalog.

This chapter explains how to configure services to manage the lifecycle of a service item. That life cycle starts at the time the item becomes available as a corporate asset, and extends to when it is provisioned via a service request to a Request Center user; through updates or changes to the item; to when it is no longer allocated to the requesting user and becomes available to be reassigned.

The Case for Service Item Lifecycle Management

System and data architects or others familiar with Configuration Management Databases (CMDBs) might be tempted to ask, "Why should I use service item management within Request Center when I have a CMDB?"

One pain point is complexity of maintaining and using a CMDB. When delivering a service to an end-user, delivery performers often need the context of what other service items that user already has. Looking that up "from the CMDB" is potentially no small matter; it requires access to several disparate systems and the knowledge to traverse them to find this information.

With Lifecycle Center functionality, Request Center becomes the most easily referenced place to find out what service items have been provisioned to/for this end-user. The mechanism for doing this doesn't require access to, or knowledge of, the CMDB (and its constituent parts); it's just a matter of going into Request Center and "taking a look". End-users (and their managers) get to view the service items they have—and potentially resolve discrepancies with the items IT *thinks* they have (according to various asset systems or CMDBs).

Further, even companies with mature CMDBs have some items that are either not tracked at all (neither in a CMDB *nor* an asset system), or have incomplete information, making the job of managing them difficult. For example, software licenses aren't the types of items you'd typically store in a CMDB. Likewise, cell phones may very well be tracked, but perhaps in an existing system that doesn't record the business-context data that is truly needed for delivering follow-on services to end-users.

Designing Service Item-Aware Services

Integrating service item lifecycle management into the service catalog leverages the capabilities inherent in Request Center to design services.

- Analyze business requirements to identify the service item and functionality required to support managing the service item's lifecycle.
- Use Service Item Manager to define the service item or to review the definition of preconfigured service items such as a virtual machine.
- Use Service Designer to specify a service item-based dictionary. The service item based dictionary provides specifies the fields required to specify detailed requirements for creating or maintaining their service items.
- Use Service Designer to design an Active Form Component which includes one or more service item-based dictionaries and appropriate rules to support the use of those dictionaries. The Active Form Component provides the user interface whereby the user enters or reviews service item or other details of their service request.
- Use Service Designer to design a delivery **Plan** (workflow) that specifies tasks to create, update, or delete the service item.
- Sometimes the service item must be created, updated, or otherwise maintained by a third-party application, such as VMware's vCenter, for a Virtual Machine, or Amazon's Elastic Compute Cloud (EC2), for a virtual machine in the cloud. In such cases, use Service Link to design the integration between Request Center and the external system.
- Once the service has been designed, go back to Service Item Manager and associate it with a service item. This allows users and managers to quickly see what services are available for reconfiguring or otherwise adjusting service items previously provisioned for them.
- Optionally use Service Item Manager's Import feature to import externally defined service items and standards into Request Center, or use the Service Link file adapter to import data for service items or standards.

Service Item Manager

The Service Item Manager module allows service designers to create and manage service items. A service item definition specifies the attributes associated with that service item. For example, a Laptop service item may include model, make, and asset ID or serial number; a software license may include the application, version, level (Standard or Professional) and other vendor-dependent information.

In addition to user-configurable service items, Request Center includes one type of preconfigured service item—a Virtual Machine, as configured by VMware. This chapter includes detailed instructions on how to configure services to manage virtual machines. The *Cisco Service Portal Integration Guide* includes detailed instructions on how to configure a Service Link "agent" that implements the Request Center interface to vSphere 4.1 vCenter server.

The Service Item Manager module also enables you to further manage your communications with vCenter server and to customize the user interface of the VMware services with vCenter. You can use Service Item Manager to import data about your virtual hardware, virtual machine templates, and configuration options into a set of "Virtual Data Center Standards". VMware services can then use those standards as reference data to validate user data entry, for example, to ensure that the user selects a valid VMware virtual machine template or appropriate configuration options when provisioning or configuring a virtual machine.

Service Designer

Service Designer is the module used to define a service request, including its presentation; authorizations and delivery plan; data required, in the form of dictionaries, for users and task performers to supply on the order form; and optional rules which can dynamically adjust the form's appearance and behavior to fit the current context.

Service Item Tasks—a special kind of workflow—are used within a service's delivery plan to trigger the lifecycle phases of the requested service item.

Organization Designer

Like all capabilities available in Service Portal, the abilities to design, maintain, and view service items and standards are governed by Role-Based Access Control (RBAC). Preconfigured roles include Service Item Manager, Service Item Designer, Service Standards Manager, and Service Item Administrator.

Service Link

Through the use of a file adapter, service items and standards can be imported from an external system. This offers an alternative to importing the data interactively, via Service Item Manager's Import feature.

Service Link also provides the ability to configure agents to communicate with third-party systems such as VMware vCenter and Amazon EC2. These agents can then be used within a service's workflow to provision or configure such externally maintained service items as required.

The End User View of Service Items

Service items are automatically assigned to a person when they are provisioned via service requests. Service items can also be assigned directly to a Request Center user through the manual Assign feature in Service Item Manager. Manual assignment is particularly useful if users have been assigned service items in an external system and that data must be brought into Request Center.

As part of Lifecycle Center, My Services offers a **Service Items** tab and My Items portlet that enable users to see the service items that have been provisioned for them and to request further changes and additions to these items. The My Services Consumer role (the default role assigned to all Request Center users, to allow them to submit service requests) does not include the ability for users to view and search on their service items. This functionality is provided by the My Services 360-Degree Consumer role, which includes the capability to "View My Service Items", and a personal preference setting allowing each user to show or hide the My Items portlet.

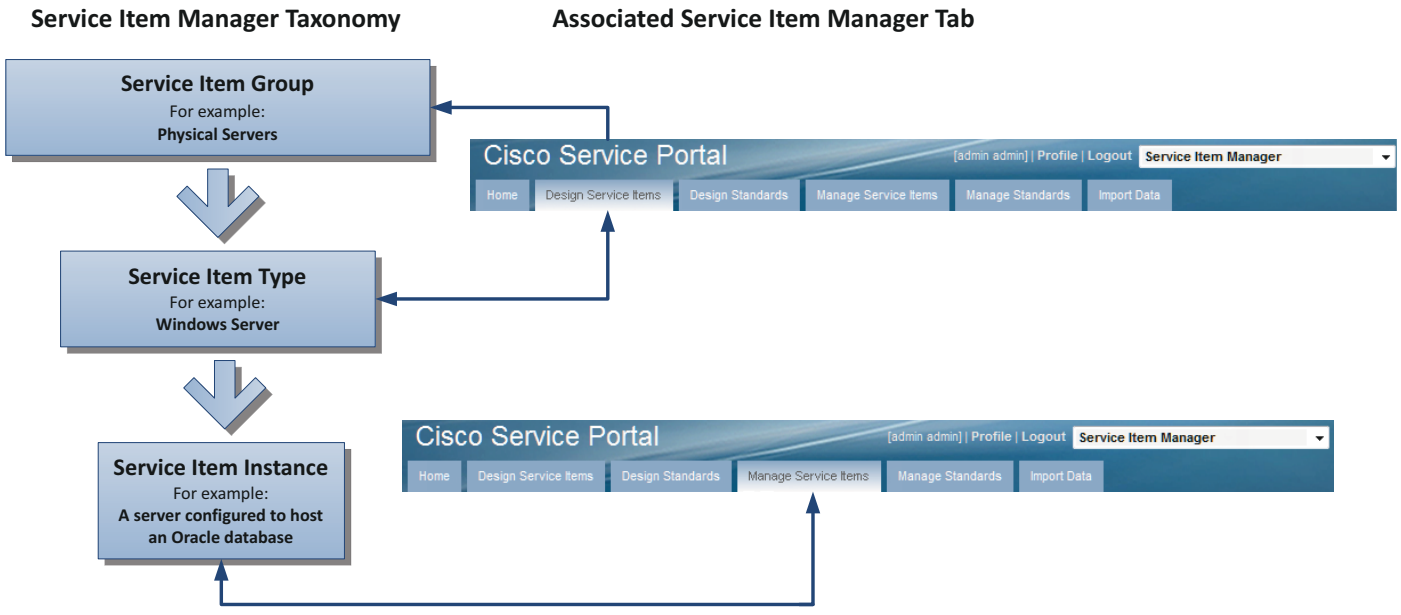
See the "Working with My Service Items" section on page 3-45 for more information.

Service Items and Service Item Manager

Service Item Manager is the Service Portal module that allows designers and administrators to design and manage service item types and instances; and to create, import, and manage the supporting data (or *standards*) which provide validation and reference data for the service (order) forms through which users will request service items.

The Service Item Manager Taxonomy

The Service Item Manager taxonomy is as follows:

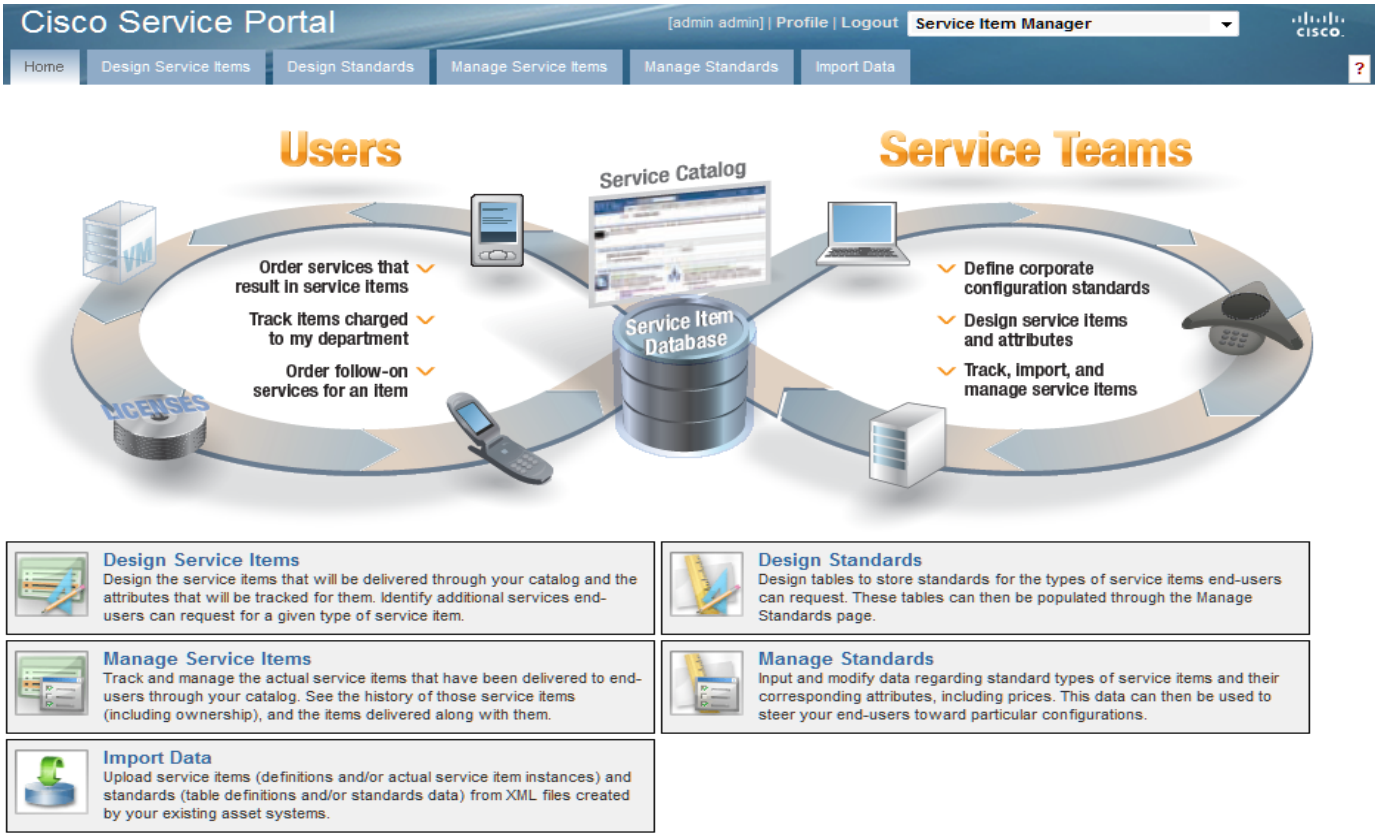


- Service Item Group – The top of the hierarchy, managed on the **Design Service Items** tab. The service item group Virtual Hardware is provided automatically and cannot be modified. User-configurable service item groups can be freely added.
- Service Item Type – A child of a service item group, also managed on the **Design Service Items** tab. The Virtual Hardware service item group contains one preconfigured service item, Virtual Machine, to support Virtual Data Center (VDC) services. This service item cannot be modified. You can create and manage additional service items in this group or within other service item groups.

- Service Item Instance – An instance of a service item type, managed on the **Manage Service Items** tab. For example, a virtual machine configured on a VMware server host is an instance of the Virtual Machine service item type, which, in turn, is a child of the Virtual Hardware group.

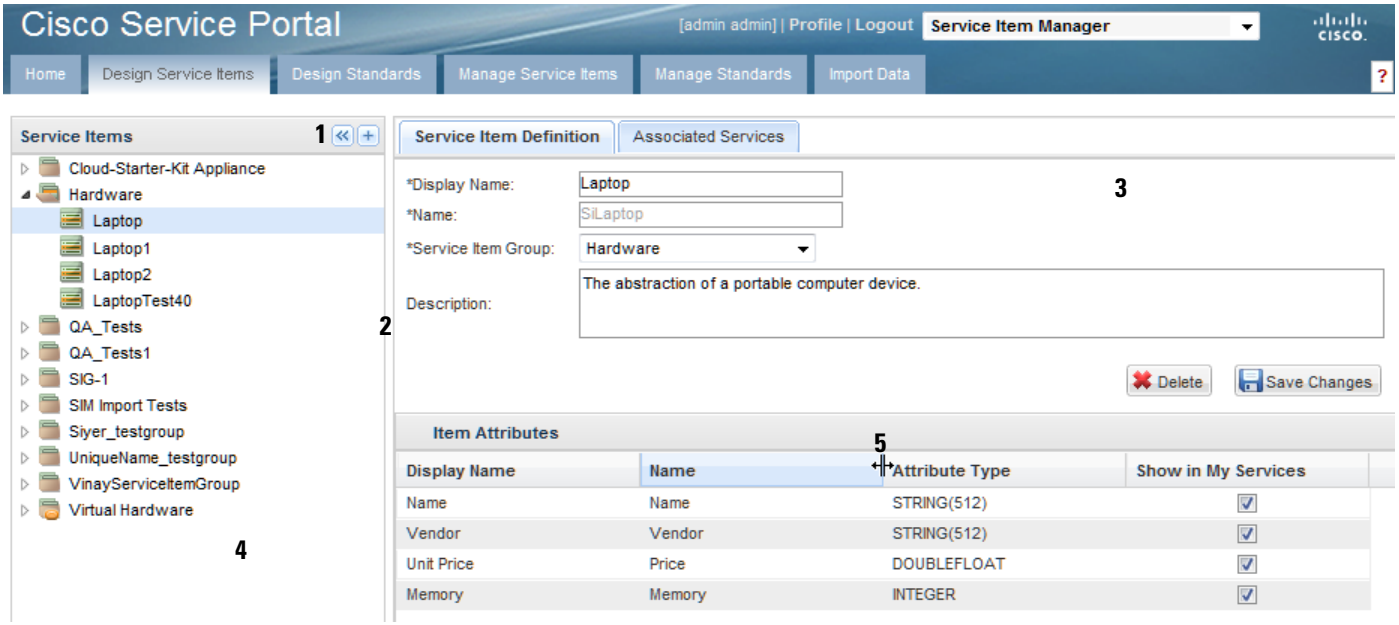
Managing the Service Item Manager Screens

When you choose the Service Item Manager module, the Service Item Manager home page appears.



You can access the Service Item Manager screens either by clicking on the tab corresponding to the option you want or by clicking on the option's description.

Most Service Manager pages consist of a list panel on the left, a content panel on the right, and a grid within the content panel, displaying service items or standards.



1	Collapse panel icon	**4**	List panel
2	Divider	**5**	Width cursor
3	Content panel		

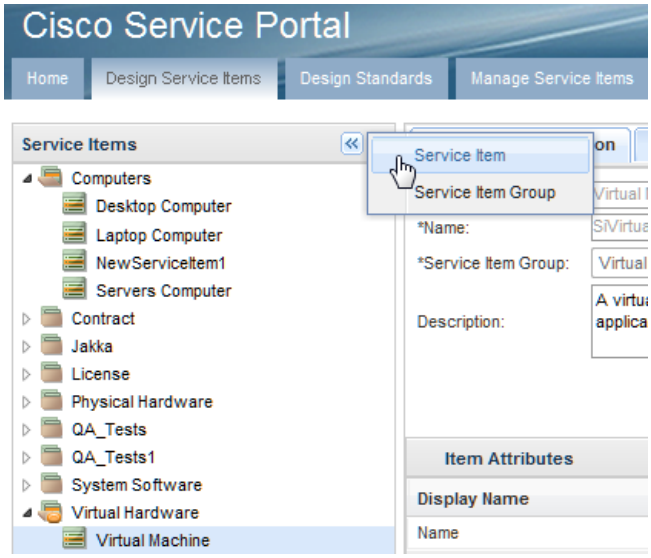
You can also control the appearance of the screen:

- The list panel can be collapsed and expanded by clicking the collapse and expand panel icons.
- The width of the list and content panels can be changed by dragging the divider.
- To change the width of a column in the grid, position the cursor on the line between that column and the next. The cursor changes to a double line with two arrows pointing in opposite directions. Once the new cursor appears, you can drag holding down the left mouse button to adjust the column width.

Designing Service Items

Use the **Design Service Items** tab to view the available service item groups and associated types and to create or modify groups or types.

You can create a new service item or service item group by clicking the plus sign (+) and then choosing **Service Item** or **Service Item Group**, respectively, from the resultant Create (+) menu, as shown below.



A group is defined by specifying the group name and optionally supplying a description. A service item definition consists of the item's name, a display name, optional description, and a set of attributes that describe the item.

- The Display Name is the user-friendly version of the name; it may contain spaces. The display name is used in the My Items Portlet and as the basis for the caption for a Service Item-based dictionary. The Display Name must be unique within a site.
- The item Name is the name by which the system references the service item and its data. The item name can contain only alphanumeric characters and the underscore (_), with no embedded spaces. It must begin with an alphabetic character. It corresponds to a table that is dynamically created and maintained in the Request Center transactional database. Service Item Manager creates the database table with the same name as the item name with a prefix of "Si". The item Name must be unique within a site.

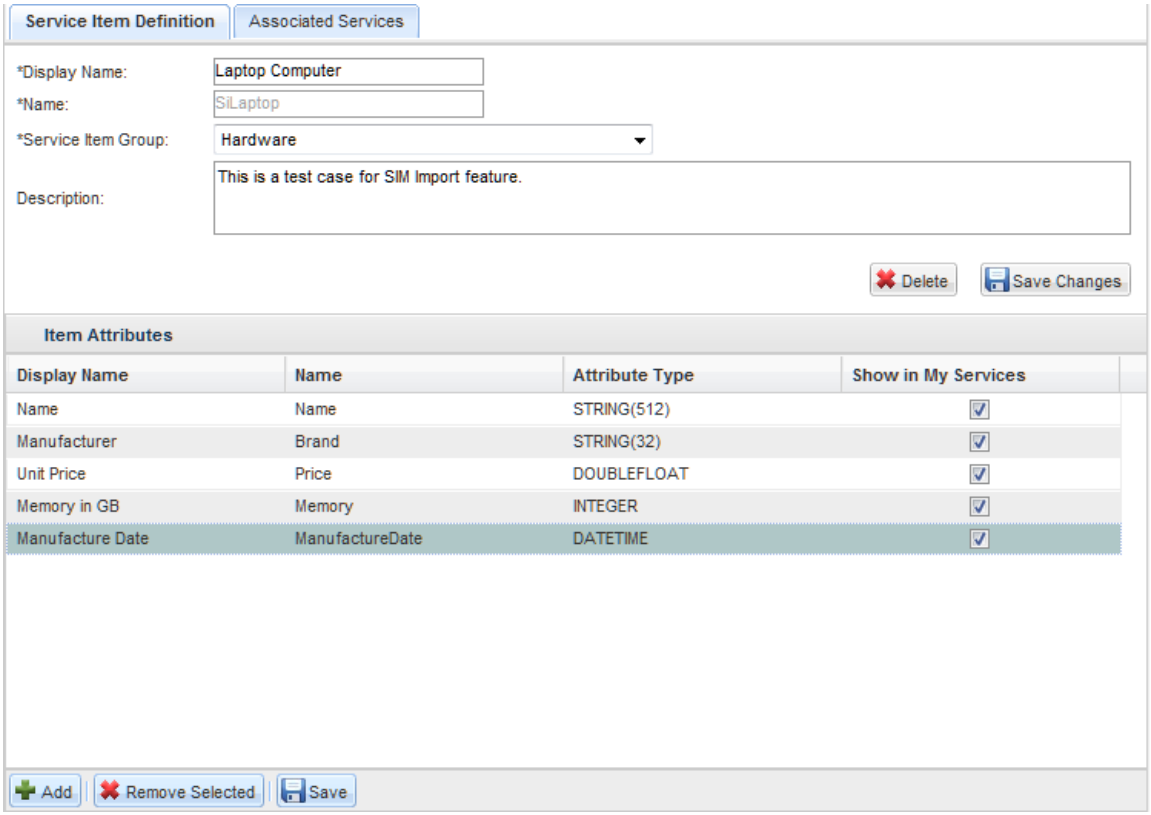
The item attributes specify the fields (data) that are maintained about the service item. All service items must have a Name, which provides a unique identifier for the item. Other attributes may be added as shown below and described in the following table.

To maintain the definition of a service item:

Step 1 Click **Add** to add a new attribute to the service item. An empty line opens at the bottom of the grid, where you can type the service item definition.

Step 2 To delete one or more attributes, choose the attribute to be deleted by clicking on it (use Ctl-Click or Shift-Click to choose multiple attributes), and then click **Remove Selected**. To remove an attribute from the selection, simply click on it again.

Step 3 Click **Save** to save your changes.



Field	Description
Display Name	The name of the attribute that will appear as the attribute label on all Request Center pages, including the My Items portlet; can include HTML formatting tags if desired.
Name	The internal name for the attribute; keywords and reserved words in the underlying database, for example, INTEGER, ORDER, or VARCHAR, cannot be used. Name length is limited to 27 characters.
Attribute Type	The data type for storing attribute data.
Show in My Services?	True if the value of the attribute should be displayed in My Services; false otherwise.

All attributes added or updated since the last save are marked by the red triangle at the upper left of the attribute's display name.

Attributes automatically maintained as part of the Service Item Subscription or Service Item History should not be added to the service item definition. Such attributes, including the Customer ID, RequisitionID and other aspect of service item usage, are described in the "Defining Service Item-Based Dictionaries" section on page 3-22.

Attribute Type

The attribute type specifies the data type used to store the attribute's value. The following data types are available:

Attribute Type	Description
INTEGER	A positive or negative whole number or zero; in the range: –32,676 to 32,676
LONGINTEGER	A positive or negative whole number or zero
DATETIME	A date and time
MONEY	A positive or negative number with up to three digits of decimal precision or zero
DOUBLEFLOAT	A positive or negative number with decimal precision or zero
STRING(32)	A string (alphanumeric) value up to 32 characters long
STRING(128)	A string (alphanumeric) value up to 128 characters long
STRING(512)	A string (alphanumeric) value up to 512 characters long

Changing an Existing Service Item Definition

Once a service item has been created, its name cannot be changed. The display name can freely be changed.

Attributes can be freely added to the service item definition. Once the revised definition is changed, existing service items will have blank values for the added attributes. Administrators can use the Manage Service Items option to provide values for these new attributes.

Attributes can be freely deleted from the service item definition. Corresponding data will be removed from any existing service items.

An existing service item attribute's Display Name can be changed, but the attribute cannot be renamed, nor can its data type be changed. To make either of these changes, you must delete the original attribute and add a new one. Any data associated with the original attribute will be lost.

If a Service Item-Based Dictionary has already been created, any changes made to the service item definition need to be applied manually to the dictionary. So long as the name of a field added to the dictionary exactly matches the name of the corresponding attribute in the service item, the relationship between the field and attribute is maintained.

Service Item History and Subscriptions

Request Center logs every time a service item is created, updated, or deleted by a service request. The history of these transactions is available in the Service Item History table. In addition, the current status of the service item is available in the Service Item Subscription table. These tables form the basis of the queries available in the My Items portlet. They can also be used in data retrieval rules.

The Service Item Subscription contains the following information about the item:

Field Name	Description
RequisitionID	The RequisitionID of the requisition (shopping cart) which includes the service request that created the service item subscription.
RequisitionEntryID	The RequisitionEntryID of the service request that created the service item subscription.

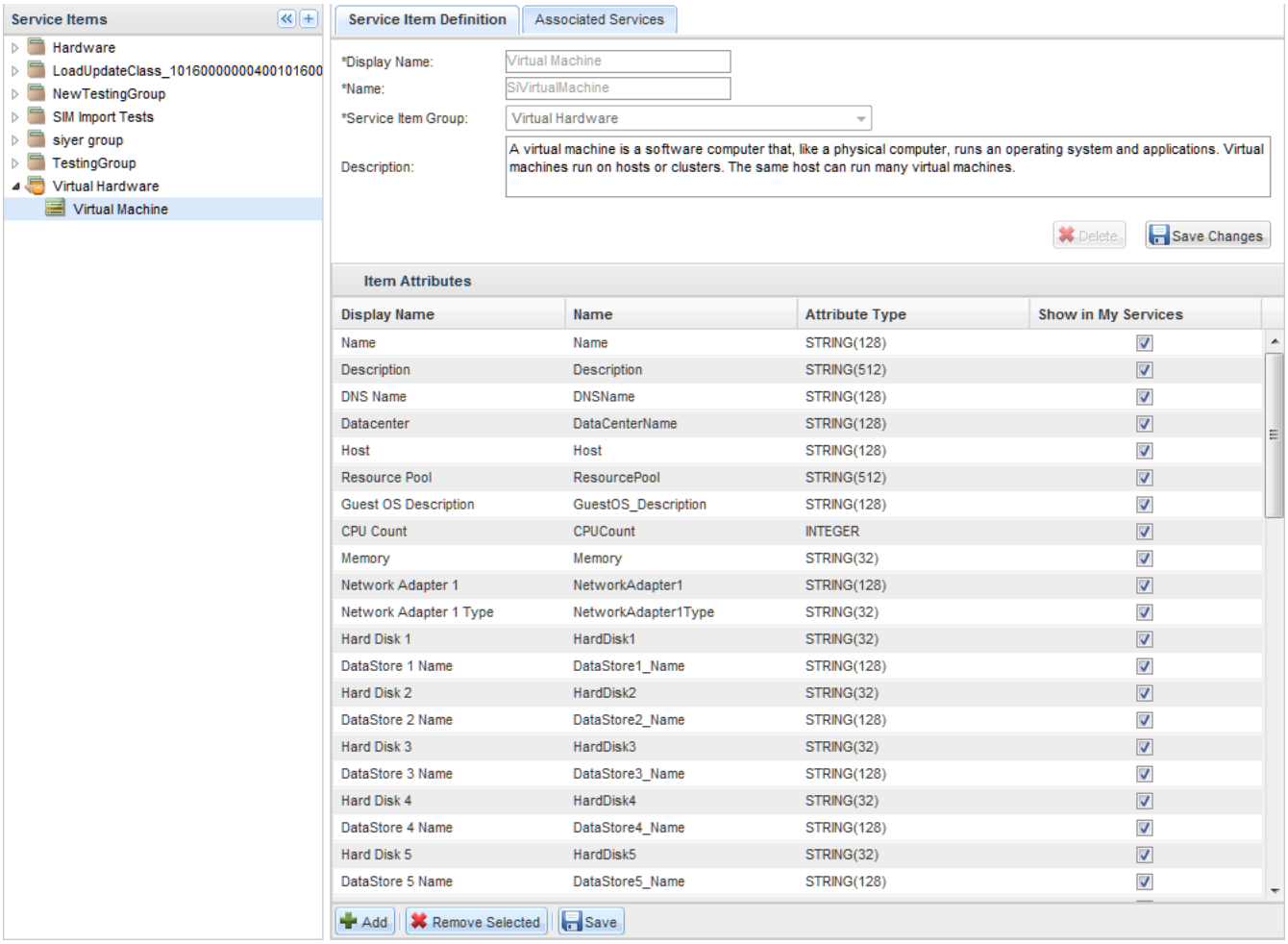
Field Name	Description
CustomerID	The ID of the customer to whom the service item is currently assigned; blank if the item is not currently assigned.
DisplayName	The name of the service item to be displayed in the Service Items tab and My Items portlet.
ServiceItemTypeID	The unique identifier of the type of service item—user-defined or preconfigured (internal use only).
ServiceItemTypeName	The type of service item—user-defined or preconfigured.
ServiceItemID	The unique identifier of the service item instance (internal use only).
ServiceItemClassificationID	The unique identifier of the service item group of the service item (internal use only).
OrganizationalUnitID	The Home OU of the customer to whom the service item is assigned. Its use is recommended if you want to query service items assigned to a particular OU as well as service items assigned to a specific person.
SubmittedDate	The date the request that includes the “Create Service Item” task was submitted.
AssignedDate	The date the item was assigned to its current owner.
Operation	The most recent operation performed on the service item which is one of Create, Update, or Delete.

The Service Item History contains the following information about the item:

Field Name	Description
RequisitionID	The RequisitionID of the requisition (shopping cart) which includes the service request that created the service item subscription.
RequisitionEntryID	The RequisitionEntryID of the service request that created the service item subscription.
CustomerID	The ID of the customer to whom the service item is currently assigned; blank if the item is not currently assigned.
OrganizationalUnitID	The Home OU of the customer to whom the service item is assigned. Its use is recommended if you want to query service items assigned to a particular OU as well as service items assigned to a specific person.
SubmittedDate	The date the request that included the “Create Service Item” task was submitted.
AssignedDate	The date the item was assigned to its current owner.
Operation	The operation performed on the service item.

Virtual Machine Service Item

As the screen shot below shows, attributes for the Virtual Machine include the Name, DSN Name, and other information that must be supplied for a particular VM to be valid.



The attributes for the Virtual Machine service item are summarized in the table below. More details on each of these attributes and their usage in VMware operations are provided in the "Configuring a VMware Operation" section on page 3-33.

Field Name	Description	Used in outbound requests to vCenter?	Captured inbound from vCenter?
Name	Mandatory for all VMware requests; cannot include the characters / (backward slash); \ (forward slash); or % (percent).	Yes	Yes
Description	Annotations of the VM in VMware.	Yes	Yes
DNSName	DNS name currently assigned to the VM.	No	Yes
DataCenterName	Mandatory for all VMware requests to indicate where the VM is located.	Yes	Yes

Field Name	Description	Used in outbound requests to vCenter?	Captured inbound from vCenter?
Host	The ESX host on which vCenter resides.	Yes (“create” and “clone” operations only)	Yes
GuestOS_Name	Short name of the operating system that the VM is provisioned for.	Yes	Yes
GuestOS_Description	Full name for the operating system that the VM is provisioned for.	No	Yes
CPUCount	Number of virtual processors to be allocated to the VM. Only integer values are accepted.	Yes	Yes
Memory	Amount of memory (in MB) to be configured on the VM. Only integer values are accepted.	Yes	Yes
HardDisk1	Size of the first hard disk of the VM. The size is an integer value in KB for the VMware adapter “create” operation, or in MB for the “clone” and “reconfigure” operations. Hard Disk 1 is required for any VM and the disk size provisioned should meet the minimum requirement for the operating system to be installed on the VM.	Yes	Yes
HardDisk2 HardDisk3 HardDisk4 HardDisk5 HardDisk6 HardDisk7 HardDisk8	The sizes of the second to eighth hard disks of the VM. The size is an integer value in KB for the VMware adapter “create” operation, or in MB for the “clone” and “reconfigure” operations.	Yes	Yes
CDDVDDrive1	ISO image file used for the first optical device.	No	Yes
CDDVDDrive2	ISO image file used for the second optical device.	No	Yes
FloppyDrive	Image file used for the floppy drive.	No	Yes
DataStore1_Name	The datastore in which HardDisk1 is located. This is also where the virtual machine definition is stored when the virtual machine is created.	Yes	Yes

Field Name	Description	Used in outbound requests to vCenter?	Captured inbound from vCenter?
DataStore2_Name DataStore3_Name DataStore4_Name DataStore5_Name DataStore6_Name DataStore7_Name DataStore8_Name	The datastores in which hard disks 2 to 8 are located, if they exist in the virtual machine.	Yes	Yes
DataStore1_Capacity	Deprecated.	No	No
DataStore2_Capacity	Deprecated.	No	No
NetworkAdapter1	The network connection (NIC) for the VM network. The ESX host may be configured with multiple network port groups, but only the port groups dedicated for virtual machines can be used for virtual machine operations. The value that you specify here is used to create or update the virtual hardware called “Network adapter 1” in Create, Clone and Reconfigure operations.	Yes	Yes
NetworkAdapter1Type	The adapter type for the Network Adapter 1. The adapter types supported are dependent on the guest operating system and the network devices available on the ESX host.	Yes	Yes
IPAddress	IP address of the VM (i.e., specifically “the first vNIC”).	No	Yes
NetworkAdapter2	VM network name for Network Adapter 2.	Yes	Yes
NetworkAdapter2Type	Adapter type for Network Adapter 2.	Yes	Yes
IpAddress2	Not used.	No	No
NetworkAdapter3	VM network name for Network Adapter 3.	Yes	Yes
NetworkAdapter3Type	Adapter type for Network Adapter 3.	Yes	Yes
IpAddress3	Not used.	No	No
NetworkAdapter4	VM network name for Network Adapter 4.	Yes	Yes
NetworkAdapter4Type	Adapter type for Network Adapter 4.	Yes	Yes
IpAddress4	Not used.	No	No

Field Name	Description	Used in outbound requests to vCenter?	Captured inbound from vCenter?
SCSIController0	First virtual hardware controller for the VM.	No	Yes
SCSIController1	Second virtual hardware controller for the VM.	No	Yes
VendorResourcePoolId	Internal ID of the resource pool in which the VM is located. This value has to be supplied when creating a VM within a specific resource pool. If the value is not provided, the VM is located at the cluster/host level.	Yes	Yes
ResourcePool	Name of the resource pool in which the VM is located.	No	Yes
ResourcePoolPath	Shows the inventory path of the resource pool including the parent resource pools above it.	No	No
CpuReservation	Guaranteed minimum CPU allocation for the VM.	Yes	Yes
CpuLimit	Maximum CPU allocation for the VM.	Yes	Yes
CpuSharesLevel	CPU share allocation setting for the VM; valid values are low, normal, high and custom, default setting is normal.	Yes	Yes
CpuSharesCount	CPU shares for the VM; applicable only when the shares level is set to “custom”.	Yes	Yes
MemoryReservation	Guaranteed minimum memory allocation for the VM.	Yes	Yes
MemoryLimit	Maximum memory allocation for the VM.	Yes	Yes
MemorySharesLevel	Memory share allocation setting for the VM; valid values are low, normal, high and custom, default setting is normal.	Yes	Yes
MemorySharesCount	Memory shares for the VM; applicable only when the shares level is set to “custom”.	Yes	Yes
AssetState	The running status of the VM.	No	Yes
VCenterURL	URL to the vCenter server that the VMware agent is configured to connect to.	No	Yes
VMwareID	Internal ID of the virtual machine within vCenter.	No	Yes
VirtualMachineTemplateName	The VM template on which the VM configuration is based; used to create a VM. A VM instance name can be also be used here in place of a template name.	Yes	No

Field Name	Description	Used in outbound requests to vCenter?	Captured inbound from vCenter?
SnapshotName	The name of the VM snapshot to be created/removed in a snapshot operation.	Yes	No
SnapshotDescription	The description of the VM snapshot to be created.	Yes	No
DeviceLabel	The VM device being modified in the VMware adapter "reconfigure" operation. This value has to be supplied for modify or delete action of hard disks, network adapters and CD/DVD drives.	Yes	No
Custom1	Memo field for capturing non-VMware data elements in the service form.	No	No
Custom2	Memo field.	No	No
Custom3	Memo field.	No	No
Custom4	Memo field.	No	No
Custom5	Memo field.	No	No
Custom6	Memo field.	No	No

Managing Service Items

A service item can be added to the Request Center repository in several ways:

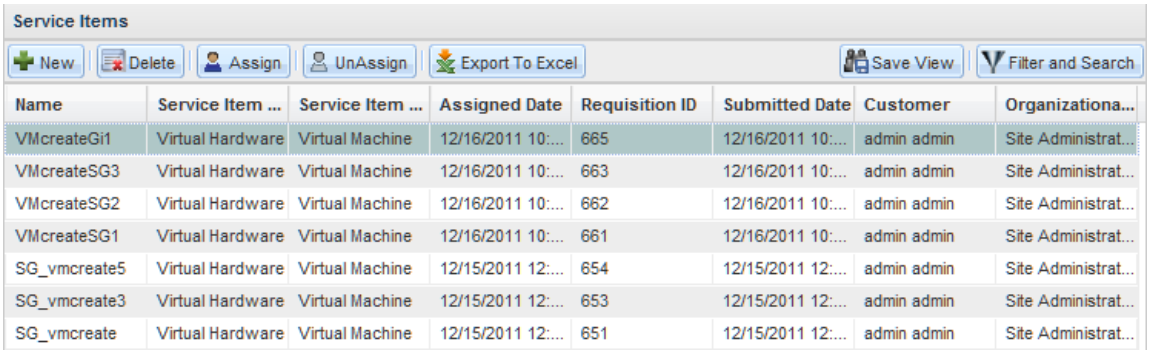
- A user requests a service which provisions a new service item via a Service Item Task
- The Service Item Import utility is used to import service item definitions and instances
- A Service Link file adapter is used to import service item definitions and instances
- A service item administrator manually adds the item through the **Manage Service Items** tab

Manage Service Items

The **Manage Service Items** tab allows service item administrators to:

- Review the service items currently tracked by Request Center, no matter how they were added to the system.
- Add new service item instances.
- Update information about individual service items.

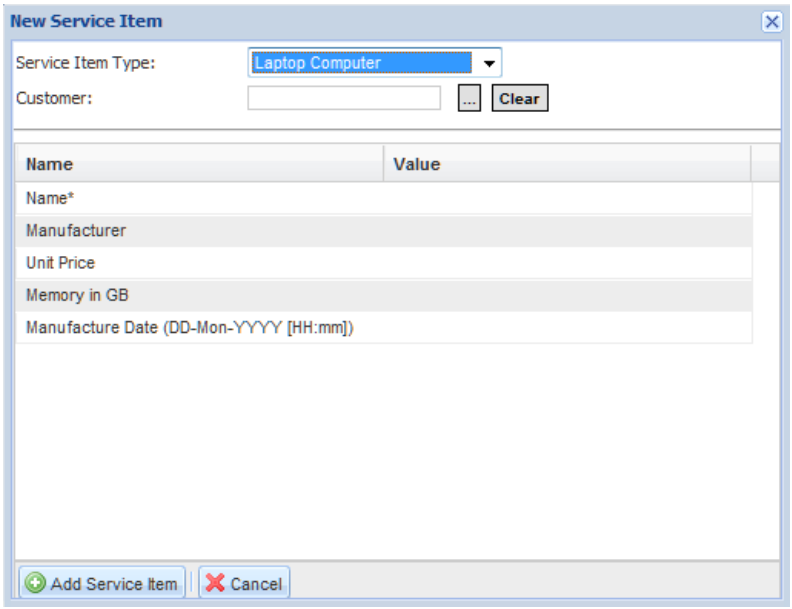
Existing service items are listed in alphabetical order by name.



Actions available are summarized in the table below. Additional information is provided in the following sections.

Action	Description
New	Add a service item of the specified type.
Delete	Delete the chosen service item.
Assign	Assign the service item to the chosen person (customer).
Un-Assign	Remove the customer to whom the service item is currently assigned.
Export to Excel	Export the list of service items to a text file in CSV (comma-separated values) format and display that file as an Excel spreadsheet.
Save View	Save as your default service item view the current view, with filters and field selection criteria in effect.
Filter and Search	Display the popup window to configure search parameters.

Adding a New Service Item



To add a service item:

Step 1 Click **New** to add a new service item. The New Service Item popup window appears.

Step 2 Choose the service item type.

Step 3 Optionally choose the customer.

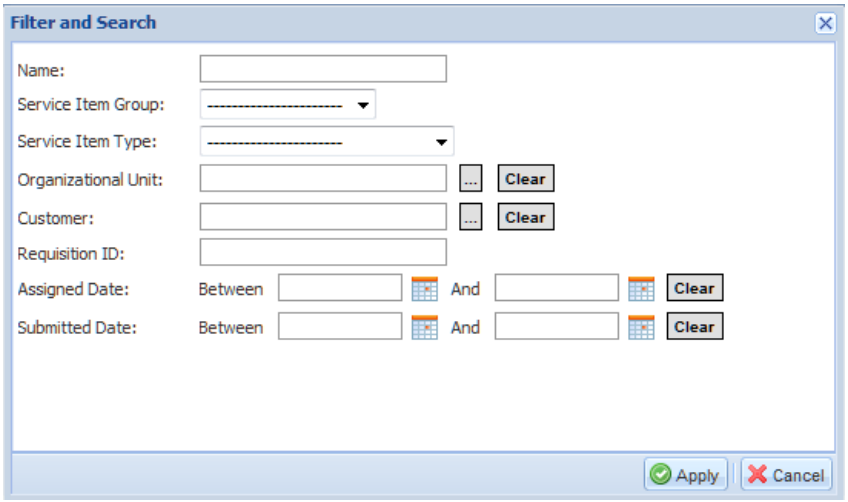
Step 4 Fill in the attributes—Name is mandatory and must be entered. Any values for DATETIME fields must be typed in the format "DD-Mon-YYYY" with an optional "HH:mm" for the hour and minute, for example, "01-Dec-2011 13:01". Once the item has been saved, the date and time are displayed in each user's preferred date and time format.

Step 5 Click **Add Service Item** when you are done.

Filter and Search

The Filter and Search option allows you to restrict the service items that are displayed. The filter criteria include all attributes that comprise the service item, as well as the customer and organizational unit to which the service item is assigned; the date the service item was assigned to that person; and the requisition ID through which the service item was created.

By default, filter criteria for alphanumeric fields use a "Contains" filter. For example, entering '386' (without the quotation marks) in the Name field below will find all service item instances whose name includes the string '386'.



Saving Views

To save the filter and search criteria currently in effect, click **Save View**. If you exit from Manage Service Items and return, the saved filter criteria will be in effect. To revise the saved view, click **Filter and Search** and edit the criteria as appropriate. Click **Save View** after you have applied those changes to replace the view with the edited or default view (if you remove all criteria).

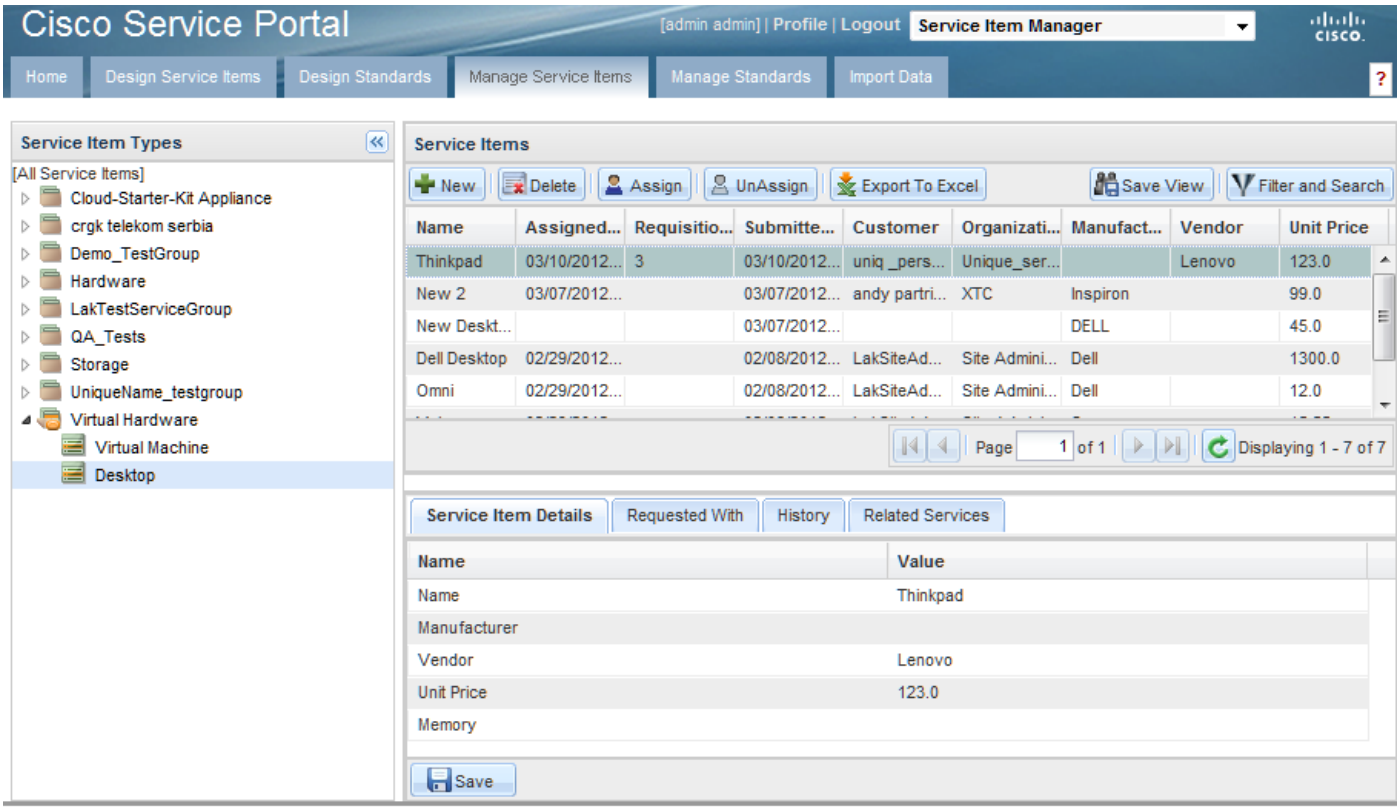
Export to Excel

To export service item data to spreadsheet format, click **Export to Excel**. A text file in comma-separated-value (CSV) format is produced. You may open the file in Excel or save it to your local disk for future use.

Associating Service Items with Services

The **Associated Services** subtab of the **Manage Service Items** tab allows service designers to associate a service item with one or more services.

Associating a service with a service type item allows My Services users to use shortcuts to order services applicable to service items previously assigned to them. Rather than having to search for the service in the My Services catalog, they can display the service item in their My Items portlet. When a service item is chosen, all associated services for which the user has ordering permission are displayed.



When the user clicks on an associated service, the service form appears. If that service has been configured correctly (as explained in the "Configuring Active Form Components" section on page 3-27), the service form is prefilled with information on the user's service item.

In principle, all services that affect the lifecycle of the service item should be associated with that service item type. In practice, that means that all services that include a Service Item Task to create, update, or delete a service item should be associated with that service item type.

Defining Standards

Service Item Manager provides the ability to specify "User-defined Standards". These standards, in the form of custom tables, can aid in the design of active form components—particularly in data retrieval rules that enable customers to "drill down" to specific answers or choices when ordering services. In fact, the main (if not only) point of a Standard is to provide reference data used to validate user entries into a service form or to provide default values for fields on that form.

Standards tables perform the same functions as relational database tables maintained in an external datasource—rows can be retrieved either for display (in a drop-down list) or validation (of user-supplied data). Both Standards tables and external tables can be used in these ways, regardless of whether the dictionary fields affected by the data retrieval rule are in a service item-based dictionary.

Standards tables and external tables differ in these ways:

- Standards tables can be maintained wholly within Lifecycle Center—no DBA intervention is required to create the table, modify its structure, or maintain its contents. In addition to providing a user interface for creating, deleting or modifying Standards records, Lifecycle Center also includes the capability to import data from an XML file into a Standards table and to create or modify the structure of that table.
- Standards table can be used directly in table-based data retrieval rules, by choosing the standard from the Standards datasource. They are also available for use in SQL-entry data retrieval rules or in the construction of SQL-based option lists. The database table name of a standard is the Name of the standard prefixed by "St".
- Lifecycle Center includes some pre-configured Standards tables for use with VMware requests. Data for these standards can either be entered manually or imported from an existing vCenter instance.

Once a Standard has been defined, the next step must be to populate the Standard table with data relevant to your Request Center installation. Once the data is present, the Standard can be used in a data retrieval rule or option list.

Using Standards Tables

Follow the steps below to use Standards tables within Request Center:

1. Specify the functional requirements for the Standard table. What data does each table need to contain? How will it be used within Request Center?
2. Use Service Item Manager to define the Standard. For Virtual Data Center standards, you may review the standard definition, but not change it.
3. Populate the Standard table with data.
4. Write data retrieval rules to access the Standard table.

Defining Standards Tables

Use the **Design Standards** tab to view the available standard groups and associated standards and to create or modify groups or standards.

You can create a new standards group by clicking the plus sign (+), and then choosing **New Standard Group** from the Create menu. A group is defined by specifying the group name and optionally supplying a description.

A standard definition consists of the standard's name, a display name, optional description, and a set of attributes that comprise the standard.

- The Display Name is the user-friendly version of the name; it may contain spaces.
- The standard Name is the name by which the system references the standard and its data. It corresponds to a table that is dynamically created and maintained in the Request Center transactional database. A Standard name can contain only alphanumeric characters and the underscore (_), with no embedded spaces. It must begin with an alphabetic character. Service Item Manager creates a database table with the same name as the standard name with a prefix of "St".

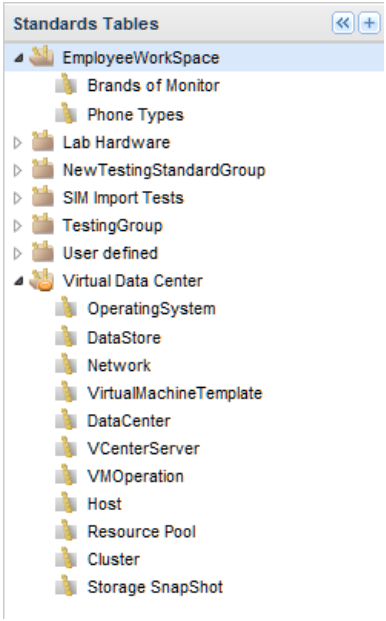
The attributes specify the fields (data) that are maintained about the standard. All standards must have a Name. Other attributes may be added as shown below and described in the following table.

Field	Description
Display Name	The name of the attribute that will appear as the attribute label on all Request Center pages, including the My Items portlet.
Name	The internal name for the attribute; keywords and reserved words in the underlying database, for example, INTEGER, ORDER, or VARCHAR, cannot be used. Name length is limited to 27 characters.
Attribute Type	The data type for storing attribute data.

All attributes added or updated since the last save are marked by a red triangle at the upper left of the attribute's display name.

Virtual Data Center Standards

Use the **Design Standards** tab to review the reference data (Standards) required to support the creation and maintenance of VM service items. These standards definitions are provided as part of the "Virtual Data Center" group.



The VMware services can use these standards to ensure that only valid VM configuration options can be requested. Data for most standards can be imported from a vCenter server by using the **Import Data** options described later in this chapter.

Standard	Description	Import Available?
OperatingSystem	Guest operating systems for a virtual machine	No
DataStore	Data stores available in the Virtual Machine data center	Yes
Network	Network connections (NICs) available for use in a VM	Yes
VirtualMachine Template	VM template information and relevant software included	Yes
DataCenter	Data center or centers hosting VMware hosts	Yes
vCenter Server	vCenter servers integrating with Request Center	No
VMOperation	Supported VMware operations	No
Host	Host names available with their respective data centers	Yes
Resource Pool	Resource pools along with the clusters or hosts in which they are located	Yes
Cluster	Clusters available within the data center	Yes

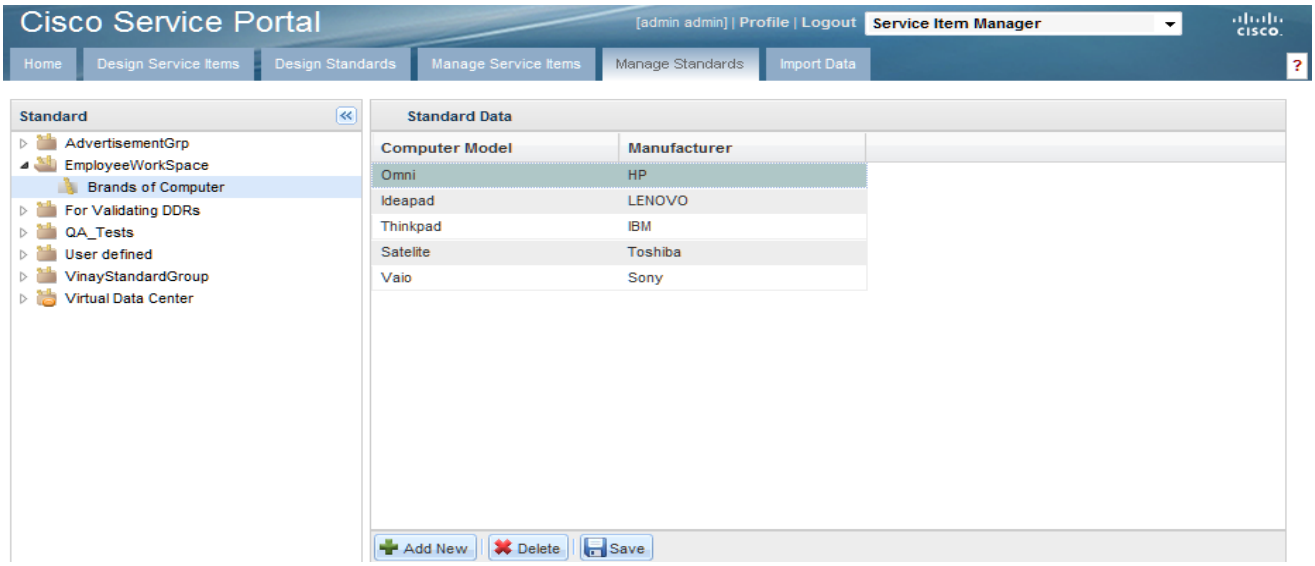
Managing Standards

Request Center supports the following ways to add data to standards tables:

- Use the **Manage Standards** tab in Service Item Manager to interactively edit standards data.

- Use the **Import Data** tab in Service Item Manager to import standards data or definitions from a file, and to import Virtual Data Center standards from a vCenter instance.

The **Manage Standards** tab presents a grid containing all attributes specified for the standard. You can add new standards; modify attribute values for existing standards; or delete one or more entries.



The Import Data option for importing standards is explained in the "Importing Service Items and Standards" section on page 3-47.

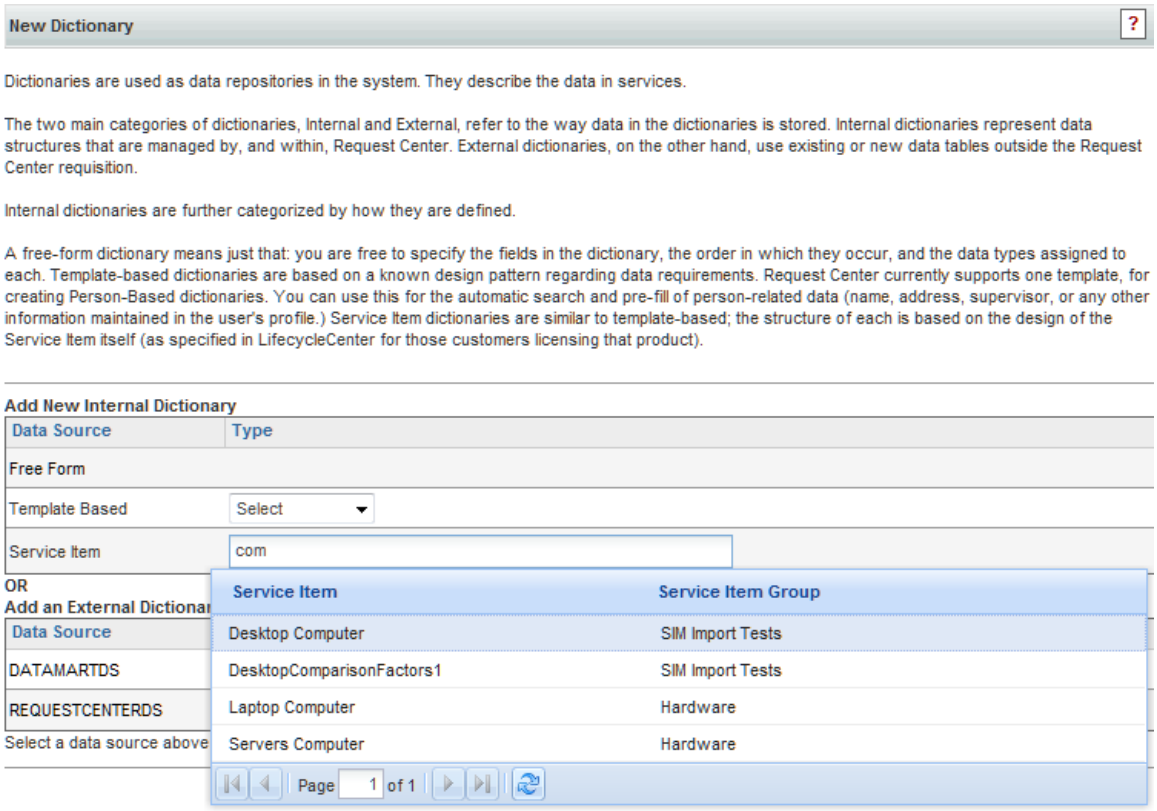
By default, Catalog Deployer deploys any standards entries when a service that references the specified standard via a data retrieval rule is deployed. You can override this behavior by changing the Administration setting to "Deploy standards entries". This would be desirable, for example, if administrators in the production environment were responsible for maintaining the standards, rather than having all standards defined in a production or test environment.

Configuring Service Item Dictionaries

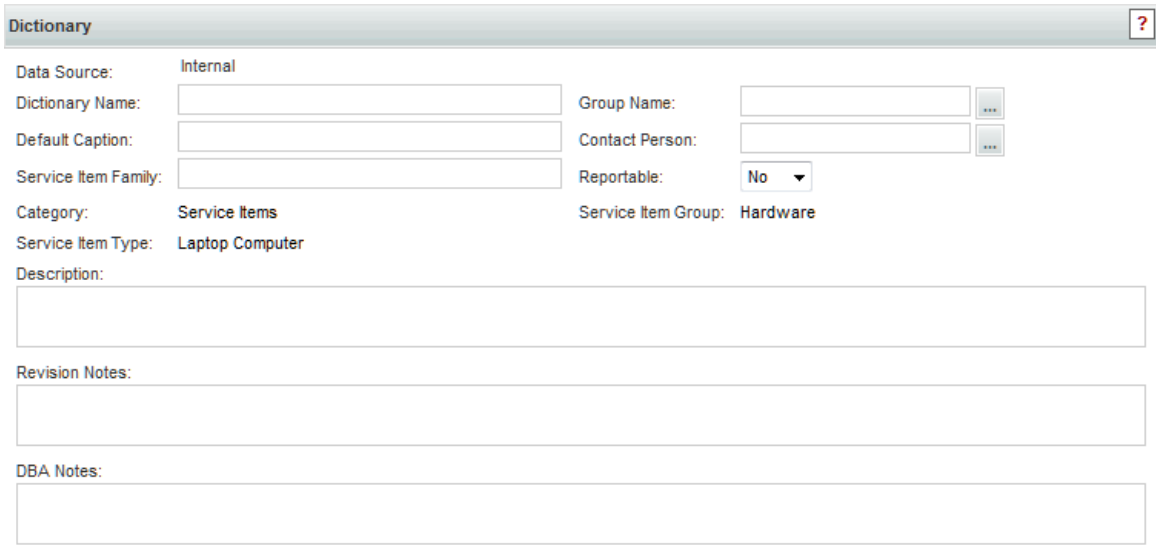
Once a Service Item has been designed, you need a dictionary that will hold the data about that service item as collected during a service request. Fields in that dictionary provide the data stored on the service item and can optionally serve to update the history of the service item and its subscriptions.

Defining Service Item-Based Dictionaries

Go to Service Designer and choose the Dictionaries option. Choose **New > Dictionary** to display the New Dictionary page. To base a dictionary on a previously defined Service Item, enter one or more characters of the name of the Service Item you want in the text box to the right of the "Service Item" label. Enter "*" to search for all Service Items. Click on the Service Item you want.



The Dictionary page appears. Fill in the top of the page normally.

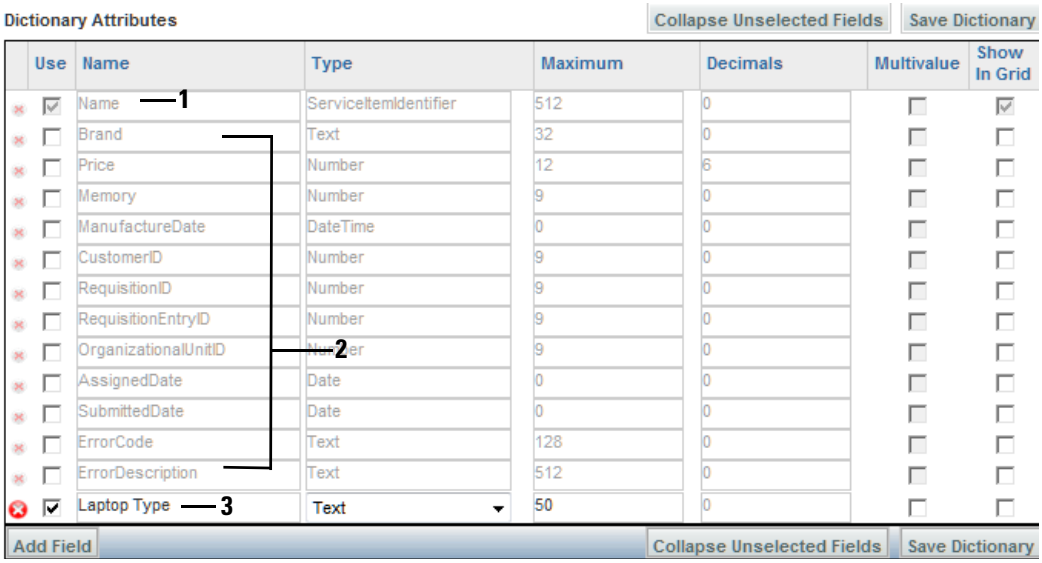


The Category, Service Item Group, and Service Item Type are not enterable. The values for these fields are set to “Service Items”, and the group and service item type on which you based this dictionary, respectively. As in previous releases, Service Item Family is not used by Request Center and any value may be supplied. This value is available for query and grouping in the Request Center data mart.

The following types of fields may be included in a service item-based dictionary:

- Fields that correspond to attributes defined in the service item itself.
- Fields about the service item subscription (history) and delivery history that are available for use in conjunction with the dictionary. These fields provide additional information regarding the service item's current usage and subscription history.
- For Virtual Machine service items only: Fields regarding the operation to be performed by the VMware adapter.
- User-defined fields.

The definition for a dictionary based on a user-defined service item might look like this:



1	Service Item fields	**3**	User-defined field
2	Service Item subscription fields		

Service Item Fields

Attributes specified in a user-defined service item are, by default, included in dictionaries based on that item. You may include the fields you want to include in the dictionary by checking the check box to the left of the field name. The Name field must be included in the dictionary; any other fields in the service item are optional. All data types are inherited from the service item's definition. The Name field has a reserved data type, "ServiceItemIdentifier" and "Show in Grid" is checked.

For Virtual Machine service items, only the Name field (corresponding to the VirtualMachineName) is automatically included in the dictionary (checked). This allows integration designers to customize the design of the dictionary to the VMware operations to be performed, and their required fields.

Service Item Subscription and History Fields

The next set of fields that appear on the Dictionary page correspond to data maintained for the service item subscription and history. For the preconfigured subscription fields, simply choose the fields you want to include in the dictionary by checking the check box to the left of the field name.

Service Item Subscription fields are summarized in the table below. These fields are automatically included in the service item's history and subscription data, with appropriate values filled in, even if you do not include them in the dictionary. The reason for including them in the dictionary is to supplement or override the default behavior that Request Center uses to provide values to these fields.

The requisition's subscription and history data is recorded when the Create or Update Service Item task for the service item is executed. If the CustomerID and OrganizationalUnitID are not included in the service item-based dictionary, Request Center uses its default logic to supply a value for each field. If a field is included in the dictionary, the value of the form field at the time the service item task is executed is used.

Field Name	Description
CustomerID	ID of the customer for the request/service item.
RequisitionID	The RequisitionID of the requisition (shopping cart) which includes the service request that created the service item subscription.
RequisitionEntryID	The RequisitionEntryID of the service request that created the service item subscription.
OrganizationalUnitID	The Home OU of the customer to whom the service item is assigned.
AssignedDate	The date and time the service item was assigned to the customer, that is, the date the "Service Item - Create" task was executed.
SubmittedDate	The date and time the request which included the "Create Service Item" task was submitted.
ErrorCode	Not used.
ErrorDescription	A textual description of the error code received if an attempt to create or assign the service item failed.

If the CustomerID or OrganizationalUnitID is included in the dictionary, the service designer is responsible for writing a rule or other mechanism to provide a value to the field.

Some sample scenarios for overriding the default behavior of Request Center include:

- Create the service item (SI) but with no owner, that is, no one has a subscription to it. For example, a new laptop has arrived, but it has not yet been assigned to anyone. This would be an alternative to manually creating the service item via the **Manage Service Items** tab or importing the service item from an external source. In this scenario, both the CustomerID and OrganizationalUnitID would be included on the service form (probably hidden by a rule in the Ordering moment), but no value supplied.
- Allow the request initiator to explicitly choose the customer for the service item (via a person-based dictionary). Copy the customer's information to CustomerID and OrganizationalUnitID fields included in the service item-based dictionary (SIBD), overriding the default customer (the initiator). This scenario is explained in more detail in the "Best Practices" section on page 3-66.
- Create the SI with no specified owner (a person corresponding to the CustomerID), but with an owning OU; for example, a project team needs a server, but the server is the responsibility of the team, not an individual person. This scenario is similar to the above, but only the CustomerID needs to be in the dictionary, its contents blanked out by a rule in the Ordering moment. The OrganizationalUnitID would be assigned—how? It could inherit the OU ID of the initiator or customer, or you could choose an OU/customer via a Person Search function.
- Unassign the service item from someone—An employee leaves the company or a project and their equipment is temporarily unassigned.

- Run a data retrieval rule to display information about when the SI was created (for example, which Requisition and Requisition Entry IDs were on the previous order).

The error description field provides feedback in case a service item task fails. This is a useful debugging tool for developers, cluing them in to possible conditional rules that might be required to validate data entry.

Possible reasons for failure to create a service item include leaving the item's Name blank, or attempting to create a service item of the same type with the name of an item that already exists. If such a task fails, the service item is not created, and any changes to the requisition are rolled back. Similarly, a task to update or delete a service item might fail if the referenced item does not exist.

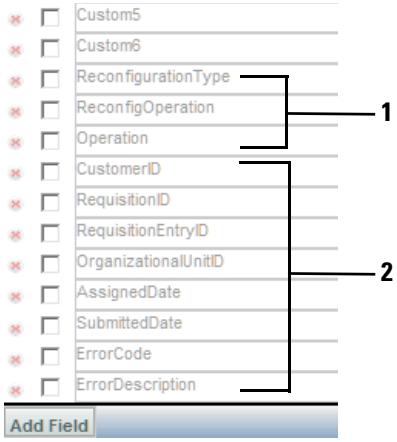
Virtual Machine Operation Fields

The Virtual Machine "Operation" field specifies the activity to be performed on the vCenter server. The Operation field must be included in all dictionaries used in the VMware integration.

Some VMware operations are used to reconfigure an existing virtual machine. For these operations, the dictionary fields ReconfigOperation and ReconfigurationType are also required.

For a detailed explanation of the use of these fields, see the "Configuring a VMware Operation" section on page 3-33.

The Operation-related fields that are available for use in a service item dictionary based on a Virtual Machine are shown below.



1	Virtual Machine operation fields	**2**	Service Item subscription fields

User-Defined Fields

Service item designers may add fields to any service item-based dictionary. Such fields might not be appropriate for the service item itself, but are a critical part of service requests relating to the service item. For example, you may want to include comments on why the customer is requesting the item, or initial or recurring monthly costs associated with item usage. Field names must not match the names of fields from the Service Item History/Subscription available for use, even if those fields are not included in this dictionary.

Such a field just becomes data that lives on the form and in the dictionary data for the form; it will never be recorded as part of the service item and is not viewable via the Service Item History and Subscription queries. However, like any dictionary field, it is viewable in the completed request and reportable via the Advanced Reporting module if the dictionary or a service that includes the dictionary is made reportable.

Maintaining Service Item-Based Dictionaries

If you create a SIBD and then add a new attribute to the corresponding service item Request Center does *not* automatically add a new field corresponding to that attribute to the dictionary. You must manually add the new field. As long as you use the same field name as the attribute name (the Data Name, not the Display Name), Request Center will correctly synchronize fields in the dictionary with the attributes in the service item.

Similarly, if you create a SIBD and then *delete* an attribute from the corresponding service item, Request Center does *not* automatically remove the dictionary field corresponding to the deleted attribute. You have to do that manually. If you don't, then you'll end up with a field on the form that has no corresponding data for the service item.

Specifying Fields in Virtual Machine Dictionaries

The Virtual Machine SIBD provides the attributes to be used with the Service Link VMware adapter for operations in vCenter Server. The required and optional fields vary depending on the type of operation involved, as explained in the "Configuring a VMware Operation" section on page 3-33. In general, the Virtual Machine and Datacenter name fields are required for all operations.

Request Center does *not* automatically show/hide fields on the form based on the operation specified or provide any default value for the fields. Hence for a dictionary that is being used for different VMware operations, you'll need to design the service form such that all the required fields get populated.

Upon successful completion of clone, create or reconfigure operations, the Virtual Machine service item is refreshed with the latest values of all the supported attributes from vCenter. The inbound update does not depend on which fields are included in the dictionary.

The owner of the Virtual Machine service item is automatically set to the customer of the requisition that invokes the VMware operation for creating or updating the virtual machine. You may override the default ownership assignment by:

- Including the CustomerID or OrganizationalUnitID in the service item-based dictionary
- Using a conditional rule to explicitly set the value of these fields

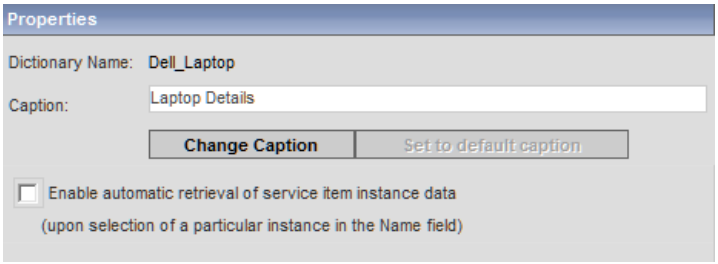
The service item attributes are updated when the next service item task for the item is executed.

Configuring Active Form Components

Once an SIBD has been defined, it can be included in an active form component. The procedure for doing so is the same as for including any dictionary in a form component—on the Form Content tab, click **Add Dictionaries** and choose the dictionary from the popup search window. If desired, you may change the display order of dictionaries or fields in the form component.

Service Item-Based Dictionary Display Properties

An SIBD has one unique property—the ability to automatically retrieve and prefill data about an existing service item instance. For a chosen form, this option is found on the Display Properties tab for the service-item-based dictionary, as shown below.



"**Enable automatic retrieval of service item instance data**" should typically be checked if the service is to be used to update or delete an existing service item. To take advantage of this prefill capability, you may need two form components based on the same service item dictionary—one to be included in services where the item is created and the second to be included in services that update or delete the item.

Using Service Items in Data Retrieval Rules

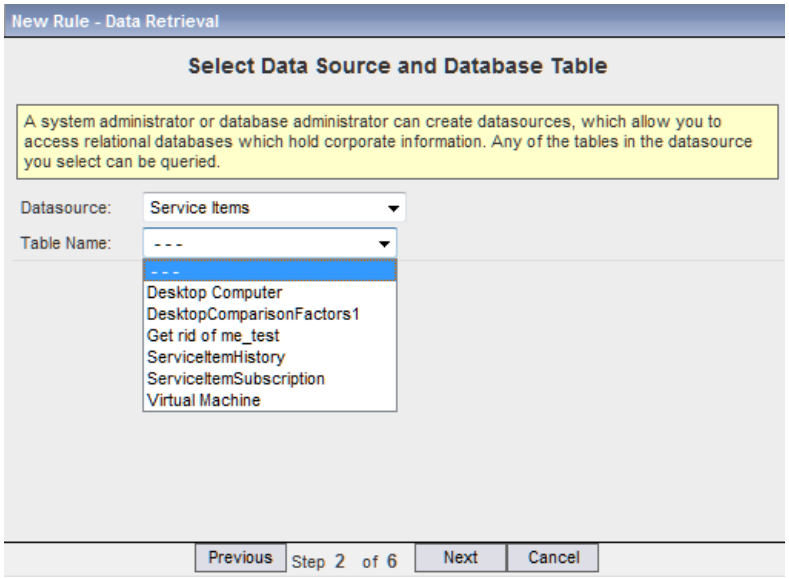
Service items are available for use in table-based data retrieval rules.

1. Create the rule and specify its name, description, and triggering event.
2. Choose "Database Table Lookup" as the Query Type.
3. The second page of the Rule Wizard appears. You can then choose "Service Items" as the Datasource.

The drop-down list for Table Name is populated to include:

- Any service items defined in Service Item Manager, including both user-defined service items and the Virtual Machine service item supplied with all Service Portal installations
- ServiceItemHistory, a table automatically maintained to track the history of a service items
- ServiceItemSubscription, a table automatically maintained to track subscriptions (the current status) of service items

You can then proceed to complete the rule definition as you would for any table-based data retrieval rule. For details on defining rules, see Chapter 2, "Active Form Components".



You can use a service item in a SQL-entry data retrieval rule by referencing the database table name which the system assigns to the service item. The table name is the prefix “Si” followed by the item name, with spaces removed. An easy way to find out the database table name (without returning to the Manage Service Items page to look up the item’s name) is to define and save a table-based rule using the service item. The generated SQL on the Summary page includes the table name.

Using Standards in Data Retrieval Rules

Like service items, standards can be freely used in both table-based and SQL-entry data retrieval rules.

Configuring the Delivery Plan

The life cycle of a service item may include many events:

- The service item is created and assigned to the person who requested it (or the person designated on the service form as its intended user)
- The service item may be updated to reflect a change in its status or configuration
- The service item is taken out of service, so must be deleted from the repository

For all service item types except Virtual Machines, each of these events may be implemented as a Service Item Task (SIT). Such a task can be integrated in a delivery plan of a service that includes a Service Item-Based Dictionary for the service item. When a service item task is executed, the service item is provisioned, updated, or deleted, as indicated by the task, and a history of this transaction is recorded.

For Virtual Machines, a Service Link agent configured to use the VMware adapter is chosen as the task’s workflow type. The agent’s properties determine the VMware API operation to be executed.

For lifecycle events on service items that are managed in external systems, sometimes the details of these service items are absent from the service requests because they are subordinate to the parent service items being requested. A typical example is a virtual application request that results in the provisioning of multiple virtual machines. The requester may be presented with only the virtual application description and the Service-Item Based dictionaries for the virtual machines are not part of the service

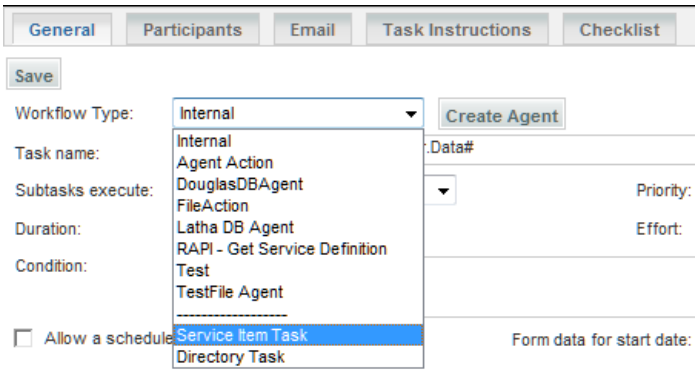
form. This precludes the use of Service Item Tasks to capture the details of the virtual machine service items. In such cases, an agent with an inbound Service Item Listener Adapter can be used in the delivery workflow to process incoming requests for create, update, and delete service item operations from external systems.

Configuring an Internal Service Item Task

To integrate a service item task into a delivery plan:

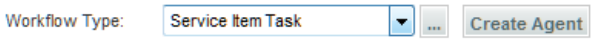
Step 1 Edit the service in **Service Designer > Services**. Click the **Plan** tab.

Step 2 On the General subtab of the Plan tab, choose **Service Item Task** from the Workflow drop-down list. This will always be the last workflow type listed, following any external tasks.

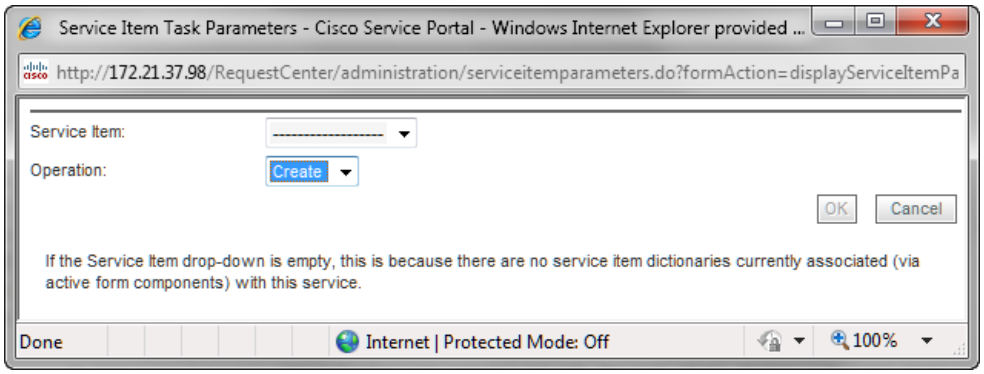


Step 3 Complete the remaining items on the General tab. Since the task will complete almost immediately, the Task Duration needs only a nominal value (or zero).

Step 4 Click **Save** to save the task. Once a service item task has been saved, an ellipsis (…) appears to the right of the Workflow Type.



Step 5 Click the ellipsis. A popup for supplying details about the Service Item Task appears.



Step 6 Choose the service item to which the task applies from the Service Item drop-down list.

Step 7 Choose the operation to be applied (Create, Update, or Delete) from the Operation drop-down list. See the next section for more information on service item task operations.

Step 8 Click **OK** to save the task definition and dismiss the popup window.

Service Item Task Operations

Service item task operations instruct Request Center to create, update, or delete a service item. The service item task is executed as specified by its sequence in the service's workflow.

Creating a service item

When a service item task with an operation type of "Create" is executed, Request Center:

- Creates an entry for the service item in the Service Item Knowledge Base. The entry includes all attributes of the service item for which data has been supplied via the service form.
- Creates an entry for the service item in the Service Item Subscription table. This table records all service items and their current status.
- Records the customer, current request, the date and time the request was submitted, and the date and time the service item was created.
- Creates an entry for the service item in the Service Item History table. This table records the requisition that created the service item.

Updating a service item

When a service item task with an operation type of "Update" is executed, Request Center:

- Updates the existing entry for the service item in the Service Item Knowledge Base.
- Updates the entry for the service item in the Service Item Subscription table to reflect the changed status of the service item.
- Creates an entry for the service item in the Service Item History table. This table records all operations (and the requisition in which the service item task occurred) that affected the status of a service item.

Deleting a service item

Deleting a service item removes all traces of the service item from the system, including its history. Do you really want to do that? Deleting an item would erase all references to the item, including its history. In some scenarios, it might be better to include an attribute in the service item to mark it as "Inactive" or "Defunct", and to write conditional rules that prohibit provisioning such items.

Service Item Subscription Processing Rules

The customer and organizational unit information within the Service Item Subscription table can be set independently from that of the requisition:

- If no subscription information is provided in the create operation, the item is assigned to the customer of the requisition and that person's Home Organizational Unit.
- If only the Customer ID is specified at the time a service item instance is created, the Organizational Unit ID of the item is set to the home organizational unit of the customer.

- If a value is provided for either the Customer ID or Organizational Unit ID, that value is used to update the service item subscription. If that value is either null or zero, the corresponding subscription field is set to null. The absence of a property in the service item based-dictionary means no change/override on the attribute value.
- Requisition ID and Requisition Entry ID provided in the dictionary are ignored and will not be used to update the subscription record.

The possible combinations for customer and organizational unit assignment and the outcome are summarized as follows:

When service item is created

Service Item Based Dictionary Field		Resulting Subscription	
Login Name	**OU Name**	**Customer**	**OU**
None	None	Customer of Requisition	Home OU of Customer
None	Blank, Invalid Value, or Zero	Customer of Requisition	NULL
None	Valid OU	Customer of Requisition	OU provided
Blank or Invalid Value	None	NULL	Home OU of Customer
Blank, Invalid Value, or Zero	Blank, Invalid Value, or Zero	NULL	NULL
Blank, Invalid Value, or Zero	Valid OU	NULL	OU provided
Valid Customer	None	Customer provided	Home OU of Customer
Valid Customer	No Value	Customer provided	NULL
Valid Customer	Valid OU	Customer provided	OU provided

When service item is updated

Service Item Based Dictionary Field		Resulting Subscription	
Login Name	**OU Name**	**Customer**	**OU**
None	None	No change	No change
None	Blank, Invalid Value, or Zero	No change	NULL
None	Valid OU Name	No change	OU provided
Blank, Invalid Value, or Zero	None	NULL	No change
Blank, Invalid Value, or Zero	Blank, Invalid Value, or Zero	NULL	NULL
Blank, Invalid Value, or Zero	Valid OU Name	NULL	OU provided
Valid Customer	None	Customer provided	Home OU of Customer
Valid Customer	Blank, Invalid Value, or Zero	Customer provided	NULL
Valid Customer	Valid OU Name	Customer provided	OU provided

Configuring an External Service Item Task

To integrate a Service Item Listener Adapter-based external task into a delivery plan:

1. Define a new agent in Service Link, specifying a "Service Task" as the Context Type.
2. Specify the "Dummy" adapter as the outbound adapter, and the "Service Item Listener" adapter as the inbound adapter. No transformations are required.
3. Skip the pages on configuring the outbound and inbound adapter properties and outbound parameters.
4. Define any inbound agent parameters to update dictionary fields, if necessary.
5. Edit the service's Plan tab to include an external task, and choose the agent created above in the Workflow drop-down list.
6. Define any required agent parameter mapping as how you would for external tasks. (See the "External Tasks in the Workflow" section on page 1-36.)

External Service Item Task Operations

Once an external service item task becomes active (that is, has a status of Ongoing) in the workflow, Service Link listens for the incoming requests to create, update, or delete service items. The changes made to the service item repository are similar to those of internal service item tasks. The channel ID of the inbound requests allow Service Link to identify the service request to be used for tracking the service item history.

You can perform other task updates that are typical of external tasks—for example, adding comments and sending parameter updates—through the incoming requests as well. When all service item operations are processed successfully, you must be sure to send a "done" action to complete the task so that the next task in the delivery plan can be triggered.

For more information about the Service Item Listener Adapter and the specification of inbound service item messages, see the Service Link chapter in the *Cisco Service Portal Integration Guide*.

Configuring a VMware Operation

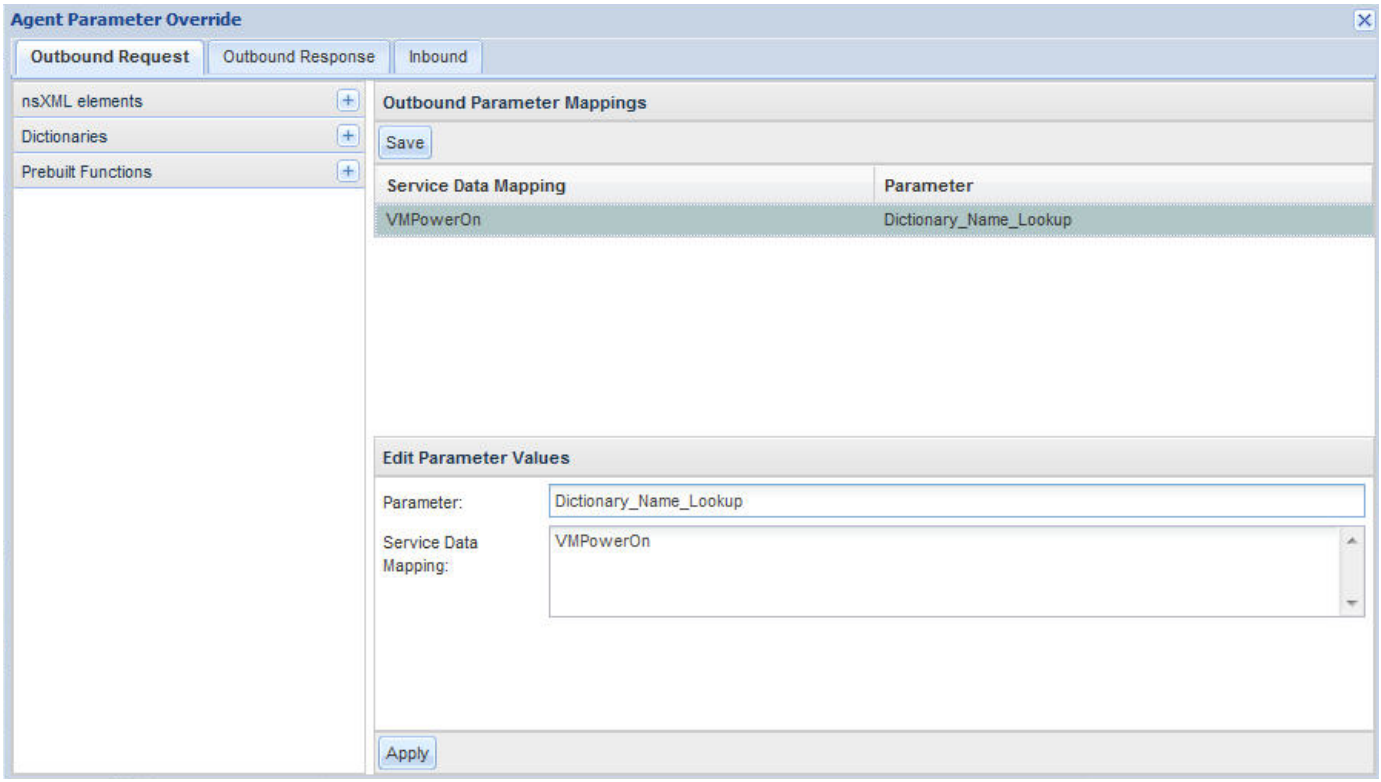
All VMware operations are configured through the use of dictionary based on the Virtual Machine service item. Fields in that dictionary are supplied values (either "automatically" through active form rules or via user data entry). The values in the designated dictionary are then passed to vCenter Server via the VMware agent invoked as part of the service's delivery plan.

To integrate a VMware adapter task into a delivery plan:

Step 1 Edit the service in **Service Designer > Services**. Click the **Plan** tab.

Step 2 Choose the appropriate VMware agent from the Workflow drop-down list, complete the remaining items on the General tab and then click **Save**.

Step 3 Click the ellipsis (**…**) that should now appear to the right of the Workflow Type to bring up the Service Link Agent Parameter Override dialog box:



Step 4 For the Dictionary_Name_Lookup parameter in the **Outbound Parameter Mappings** section, enter the VM dictionary name that contains the VMware operation details into the Service Data Mapping field, and then click **Apply**.

Step 5 Click **Save** to save the agent parameter mapping.

Step 6 Click **X** to close the popup window.

If the delivery plan calls for multiple VMware operations, separate tasks should be created to match the individual operations, referencing the appropriate VM dictionaries.

Fields in the Virtual Machine-based dictionary are configured as follows:

- The VMware operation (action) is specified as the value of the “Operation” field in the Virtual Machine dictionary. These operations correspond to entries listed in the Standards table Vituperation, and are case sensitive. Filling in this value is typically done by the service designer, and the field is not visible to the service requestor.
- VMware operations which reconfigure an existing VM require the use of the ReconfigOperation field.
- Some reconfigure operations also require the use of the ReconfigurationType field.
- All operations typically require that values be provided for the DatacenterName and VirtualMachineName fields.
- Specific operations may require the use of other dictionary fields, as detailed in the sections that follow.

Supported VMware operations are described in detail in the following sections. These sections list the dictionary fields that may be used for each operation, to specify values that are passed to vSphere vCenter. Optional fields are enclosed in square brackets ([]).

The VMware adapter supports the following of operations:

- Instance management (creating and deleting virtual machines)
- Power cycle
- Snapshot management
- Configuration changes

Instance Management

Instance management includes the following operations:

- Create a virtual machine
- Clone a virtual machine, based on a template
- Delete a virtual machine

Create a Virtual Machine

The “create” operation creates a virtual machine.

VM Operation (case-sensitive)	Parameters	Remarks
create	Datacenter Name Virtual Machine Name [Description] Host Name CPU Count Memory size (in MB) Guest OS Name Disk Size 1 (in KB) Datastore 1 Name Network Adapter Name 1 Network Adapter Type 1 [Network Adapter Name 2] [Network Adapter Type 2] [Network Adapter Name 3] [Network Adapter Type 3] [Network Adapter Name4] [Network Adapter Type 4] [VendorResourcePoolID] [CPU Reservation] [CPU Limit] [CPU Shares Level] [CPU Shares Count] [Memory Reservation] [Memory Limit] [Memory Shares Level] [Memory Shares Count]	The new VM stays powered off after it is provisioned. By default, disk 1 created is using “*thick*” provisioning. The VMware adapter does *not* validate the network adapter name against the virtual networks available in vCenter. It does *not* validate the network adapter type against the operating system on the VM either. For distributed network switches, the format for the network adapter name is <*PortGroupName*> (<*SwitchName*>), for example, “dvGroup1 (Switch2)”. The network adapter type “VMXNET_3” is not supported. If Description is omitted, the default value of “<*VirtualMachineName*> was created using Request Center” is automatically entered in the Annotation attribute of the VM.

Clone a Virtual Machine

Two operations are available to create a virtual machine based on a previously defined VM or template. The "coldClone" operation is applicable only when the clone source is a VM in powered on status. The source VM is powered off before cloning takes place, and then powered on afterwards.

VM Operation (case-sensitive)	Parameters	Remarks
clone coldClone	Datacenter Name Virtual Machine Name [Description] Host Name Template Name [CPU Count] [Memory size] (in MB) [Disk Size 1-8] (in MB) [Datastore Name 1-8] [Network Adapter Name 1-4] [Network Adapter Type 1-4] [VendorResourcePoolID] [CPU Reservation] [CPU Limit] [CPU Shares Level] [CPU Shares Count] [Memory Reservation] [Memory Limit] [Memory Shares Level] [Memory Shares Count]	The new VM is powered on automatically after it is provisioned. When cloning from a VM instance, specify the VM name in the field VirtualMachineTemplateName. If a value is present in any of the optional parameters, a reconfigure operation may be triggered automatically after the new VM is created to modify the VM settings to the parameter values specified. If the reconfigure operation fails for any reason, the clone operation is rolled back and the VM is deleted from vCenter. The following changes are *not* supported during the clone operation: • Add or remove disks • Add or remove network adapters • Override thin/thick provisioning setting of disks in the VM template • Set CPU limit or reservation to zero • Set Memory limit or reservation to zero If Description is omitted, the default value of "<Virtual Machine Name> was created using Request Center" is entered in the Annotation attribute of the VM.

Delete a Virtual Machine

The "delete" operation deletes a virtual machine from vSphere vCenter and removes all service item history and subscription data about the VM from the Request Center Service Item repository.

VM Operation (case-sensitive)	Parameters	Remarks
Delete	Datacenter Name Virtual Machine Name	The VM is automatically powered off before it is deleted.

Power Operations

For all power cycle operations, the Universally Unique Identifier (UUID) of a virtual machine (when available) can be substituted for the Virtual Machine Name.

Operation	VM Operation (case-sensitive)	Parameters
Power on VM	poweron	All power operations require the same field values to be supplied: • Datacenter Name • Virtual Machine Name
Power Off VM	poweroff	
Reset VM	reset	
Suspend VM	suspend	
Reboot VM	reboot	
Shutdown VM	shutdown	
Standby VM	standby	

Snapshot Management

The snapshot taken does not include the VM's memory. (The option "Snapshot the virtual machine's memory" is unchecked in the native client.)

Operation	VM Operation (case-sensitive)	Parameters	Remarks
Create a snapshot	snapshot coldSnapshot	Datacenter Name Virtual Machine Name Snapshot Name Snapshot Description	For the coldSnapshot operation, the VM is powered off before the snapshot is taken and powered on afterward.
Go back to a snapshot	revertSnapshot	Datacenter Name Virtual Machine Name Snapshot Name	VM power state may be changed based on the power state captured in the snapshot.

Operation	VM Operation (case-sensitive)	Parameters	Remarks
Remove a snapshot	removeSnapshot	Datacenter Name Virtual Machine Name Snapshot Name	Any sequential series of snapshots under the "tree" of the snapshot to be removed will also be deleted from the VM. For example, if two snapshots have been taken for a VM sequentially, when removing the first snapshot, the second snapshot is removed as well.
Remove all snapshots	removeAllSnapshots	Datacenter Name Virtual Machine Name	

Virtual Machine Reconfiguration

Many operations are available to reconfigure a virtual machine. For all of these, the value of the VMOperation is "**reconfigure**". What differentiate the requests are the different values for the ReconfigurationType and ReconfigOperation parameters.

Operation	Parameters	Remarks
Change memory size	Datacenter Name Virtual Machine Name ReconfigurationType = "**memory**" Memory (in MB)	
Change CPU	Datacenter Name Virtual Machine Name ReconfigurationType = "**cpu**" CPU Count	Hot plug may or may not be supported depending on the machine/operating system settings.
Add Disk	Datacenter Name Virtual Machine Name ReconfigurationType = "**disk**" ReconfigOperation = "**add**" Disk Size (in MB) Datastore Name	Use the pair of fields that correspond to the device label number. For example, to add the second disk, enter values into DiskSize2 and Datastore2_Name. (Note that the addition of hard disk 4 and above is not supported.) Multiple disks can be added in a single add disk operation. The disk added uses "*thick*" provisioning.

Operation	Parameters	Remarks
Remove Disk	Datacenter Name Virtual Machine Name ReconfigurationType = "**disk**" ReconfigOperation = "**remove**" Device Label	Examples of Device Label for disk reconfiguration are: • Hard disk 1 • Hard disk 2 • Hard disk 3
Increase Disk size	Datacenter Name Virtual Machine Name ReconfigurationType = "**disk**" ReconfigOperation = "**modify**" Disk Size (in MB) Device Label	Disk size cannot be decreased.
Add NIC	Datacenter Name Virtual Machine Name ReconfigurationType = "**nic**" ReconfigOperation = "**add**" Network Adapter Name(s) Network Adapter Type(s)	Use the pair of fields that correspond to the device label number. For example, to add the second NIC, enter values into NetworkAdapter2 and NetworkAdapter2Type. (Note that the addition of network adapter 5 and above is not supported.) Multiple NICs can be added in a single add NIC operation. Add NIC operation cannot be performed while VM is powered on.
Remove NIC	Datacenter Name Virtual Machine Name ReconfigurationType = "**nic**" ReconfigOperation = "**remove**" Device Label	Examples of Device Label for NIC reconfiguration are: • Network adapter 1 • Network adapter 2 • Network adapter 3 • Network adapter 4 Remove NIC operation cannot be performed while VM is powered on.

Operation	Parameters	Remarks
Modify NIC	Datacenter Name Virtual Machine Name ReconfigurationType = "**nic**" ReconfigOperation = "**modify**" Network Adapter Name Network Adapter Type Device Label	Specify *both* the network name and adapter type for any change. Modify NIC operation cannot be performed while VM is powered on.
Add Optical Drive	Datacenter Name Virtual Machine Name ReconfigurationType = "**optical**" ReconfigOperation = "**add**" CD/DVD Drive(s) Floppy Drive	Add only one optical drive at a time. Specify a value in one of the following parameters: CDDVDDrive1, CDDVDDrive2 or FloppyDrive.
Remove Optical Drive	Datacenter Name Virtual Machine Name ReconfigurationType = "**optical**" ReconfigOperation = "**remove**" Device Label	Examples of Device Label for optical drive are: • CD/DVD Drive 1 • CD/DVD Drive 2 • Floppy drive 1
Modify Resource Allocation	Datacenter Name Virtual Machine Name ReconfigurationType = "**resource**" [CPU Reservation] [CPU Limit] [CPU Shares Level] [CPU Shares Count] [Memory Reservation] [Memory Limit] [Memory Shares Level] [Memory Shares Count]	Reducing CPU or Memory limit/reservation to zero is not supported.

Inbound Responses from vCenter Server

Error Messages

When vCenter Server cannot successfully complete an operation, an error message is returned from vCenter Server and delivered by the VMware agent to the service request that invoked the operation. You see this message in the ErrorDescription field in the Virtual Machine dictionary.

From a Service Link perspective, receiving an error message is still considered a successful communication and therefore the status as shown in the **Messages** list on the Service Link View Transactions **page** is "Completed". The service request should be designed so that the requester or performer can make use of the error message to control the workflow and generate notifications as necessary. The ErrorDescription field should be included in any Virtual Machine dictionary you create from the Virtual Machine service item. Consider hiding the field with form rules when the user is ordering the service.

See the "VMware Adapter Error Messages" section on page 3-73 for the list of error messages and recommended actions.

Virtual Machine Details

Upon successful completion of the Create, Clone, and Reconfigure VM operations, VM instances are created/updated in the Service Portal Service Item repository. All attributes that are supported for inbound integration with VMware are refreshed for the service item instance, regardless of whether the attributes are included in the Virtual Machine-based dictionary.

By default, the owner of the VM is set to the customer of the service request. (If the request was ordered on behalf of another user, the VM owner is the request's customer, not the requestor.) This behavior can be overridden by configuring the Active Form Component that includes the VM-based dictionary, as explained in Chapter 2, "Active Form Components".

The IP Address and Domain name of the VM, though available in vCenter Server, are not always captured in time in the inbound response sent to the VMware agent, so these request details may remain blank.

The VM instances can be accessed and maintained in the Service Item Manager module. Custom attributes for individual VM instances can also be specified there to facilitate the use of filtering criteria in data retrieval rules.

The Delete VM operation updates the Service Item repository to remove the VM instance from being accessible by the end users.

Miscellaneous Considerations

Service Items Datasource

In Service Designer, you can query the Service Items data source for a person's VMs, using the PersonID as the CustomerID. Then, assuming that the person selects a VM from a drop-down list populated by a data retrieval rule, you use the VM name as the key to the VirtualMachine table. Multiple rows may be returned from that second query—and indeed the query may return a VM that does not belong to the user. So in this case, you have to anticipate multiple results coming back, and probably ask the user to inspect each in turn to see which one matches the desired machine (likely using the DataCenter and Host as additional identifying attributes).

Reconfiguring a Virtual Machine

To reconfigure a virtual machine, you must first power off the virtual machine. The reconfigure API call does not automatically power off the virtual machine upon a request to reconfigure the virtual machine. If the virtual machine is already down, the reconfigure request may proceed. If the virtual machine is up upon receipt of the reconfigure request, the reconfigure task may still be completed successfully, but it is not recommended because the reconfigure action may disrupt processes that are currently running on the machine. Certain actions are, in fact, prohibited if they were to be performed through the VMware Infrastructure Client. Therefore, service designers need to add a "power off" task BEFORE the "reconfigure" task. This ensures that the subsequent request to reconfigure the virtual machine may proceed. If the virtual machine is already down upon receipt of the power off task, the power off task fails, but this failure is harmless and does not impede the progress of the reconfigure task.

Using the VM Dictionary and the VirtualMachine Table

If you instantiate a dictionary from the Virtual Machine service item type with the intention of distributing results from a data retrieval rule that queries the VirtualMachine table to the fields in that dictionary, you will encounter an error when distributing the HardDisk1 and Memory columns. The error will not be caught until the user attempts to submit the form. The workaround is to either distribute these columns to Text fields in the dictionary (like one of the Custom*x* fields), or create another dictionary to which to distribute the results retrieved by the rule.

Furthermore, if you are planning to use the HardDisk1 or Memory values in any conditional rules that check for a certain numeric limit as to what the user can request, you will need to strip the unit of measure string away from the numeric characters.

The HardDisk1 and Memory fields in the dictionary template are defined as Number fields because the VMware adapter requires them to be numeric (even though they are written to the VirtualMachine table as strings). The dictionary template is optimized toward use by the agent, and not toward using data retrieval rules to retrieve current values.

Using the VM Dictionary with Service Item Tasks

Since Virtual Machine-based dictionaries can be used like any other service item dictionary with a service item task, you may want to have a VMware adapter task and a service item task both reference the same Virtual Machine dictionary in the delivery plan. However, it is possible that there are optional fields that are left blank for the VMware operations when they would actually carry values in the service item instance. For example, HardDisk1 is left blank in a "clone" operation to mean that the VM template disk 1 can be cloned as is. If such a dictionary is used in a follow-up service item task to reassign the VM from one person to another, the update action will also reset HardDisk1 to zero.

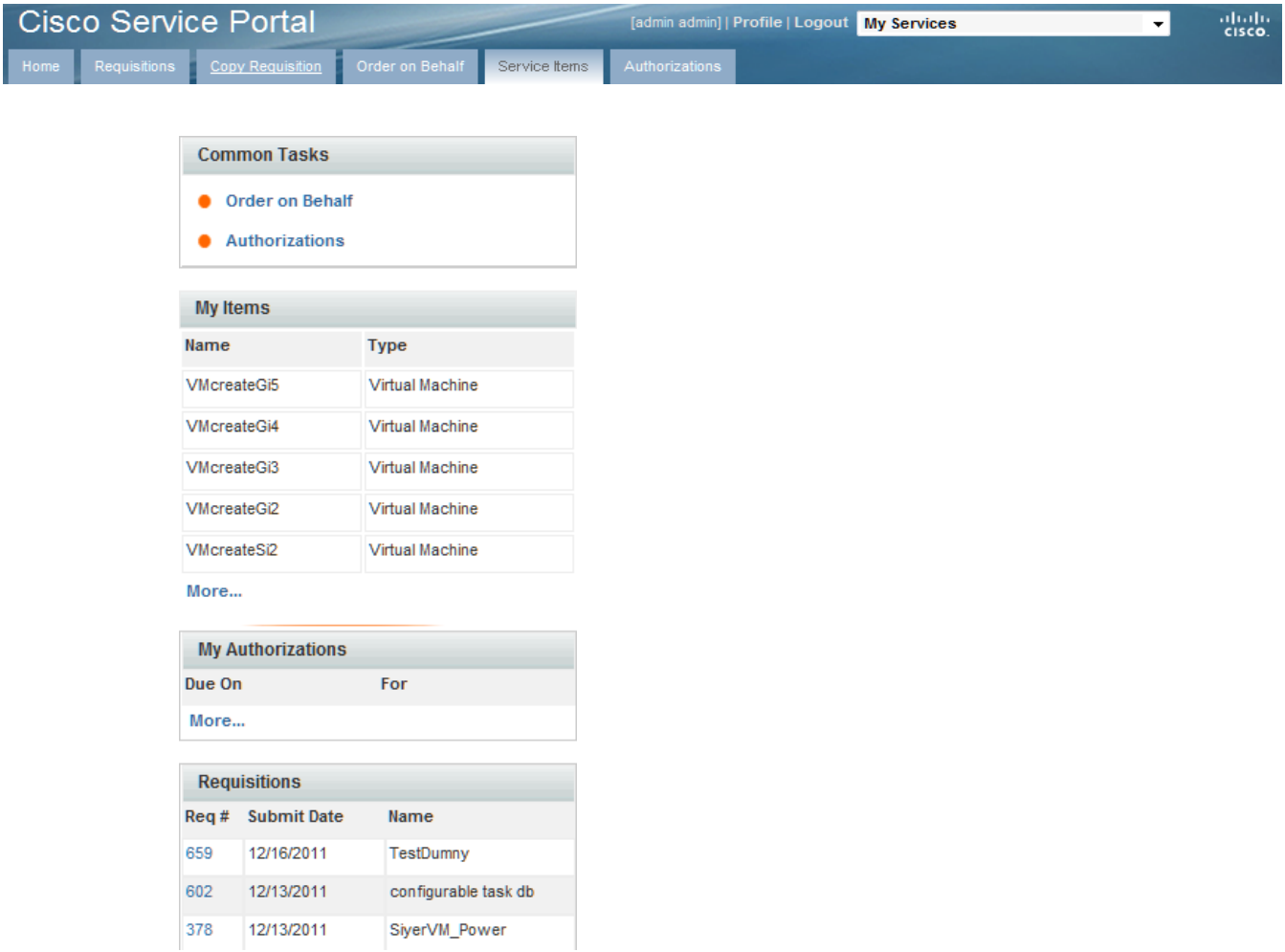
To avoid inadvertent updates to the VM attributes, it may be necessary to have separate VM dictionaries to be defined for different purposes. Form rules may be used to copy values across dictionaries to reduce the need for duplicate data entry.

An End User's View of Service Items

Viewing My Service Items

My Services offers a Service Items tab and My Items portlet that allow users to access and order services associated with service items. Service items are assigned to a person when they are provisioned via service requests, or when the items are assigned directly to the person using the Service Item Manager module.

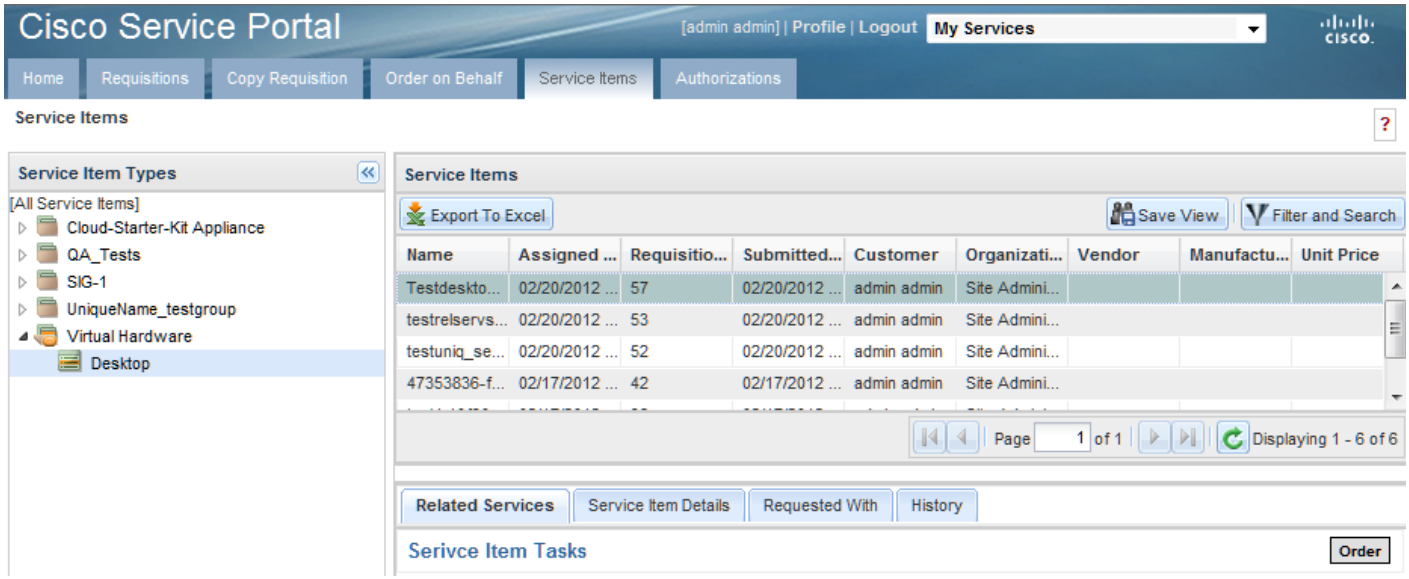
The My Items portlet displays the five service items most recently provisioned to the user. Click on the **More** link in the My Items portlet or the Service Items tab to see a complete list of service items. To see the My Items portlet, the My Services "View Service Items Portlet" setting must be turned "On" in the Administration module (see the "Administration Setting for My Items Portlet" section on page 3-70). In addition, the service items that each user can access depend on the capability granted to the user (as explained in the "Administration" section on page 3-68). Users can either "View my service items", "View service items in my business units" or "View service items in my business units and their sub-units".



Working with My Service Items

By default, the Service Items page displays all service items assigned to the current user (or members of the current user's business unit). You can adjust the display by:

- Expanding the service item groups displayed in the list panel on the left, and choosing a particular service item type to display.
- Using **Filter and Search** to specify search criteria.



The Service Items page offers the following subtabs:

- Related Services (open by default) allows users to request any service that can modify the configuration of the chosen service item.
- Service Item Details shows all attribute detail for the chosen service item.
- Requested With shows other service items, if any, that were provisioned via the same service request that provisioned the chosen service item.
- History shows all transactions that have affected the status of the chosen service item.

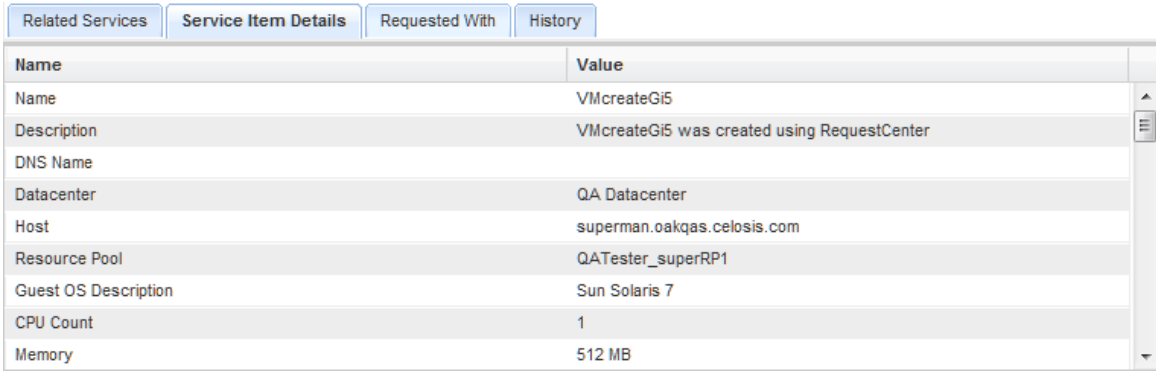
Related Services

The services displayed in the **Related Services** subtab act just like services displayed via a search on the My Services home page: you can click on the service name to display the overview of the service before ordering, or proceed directly to ordering by clicking **Order**.

Unlike the My Services home page, however, these services are right there, without having to search!

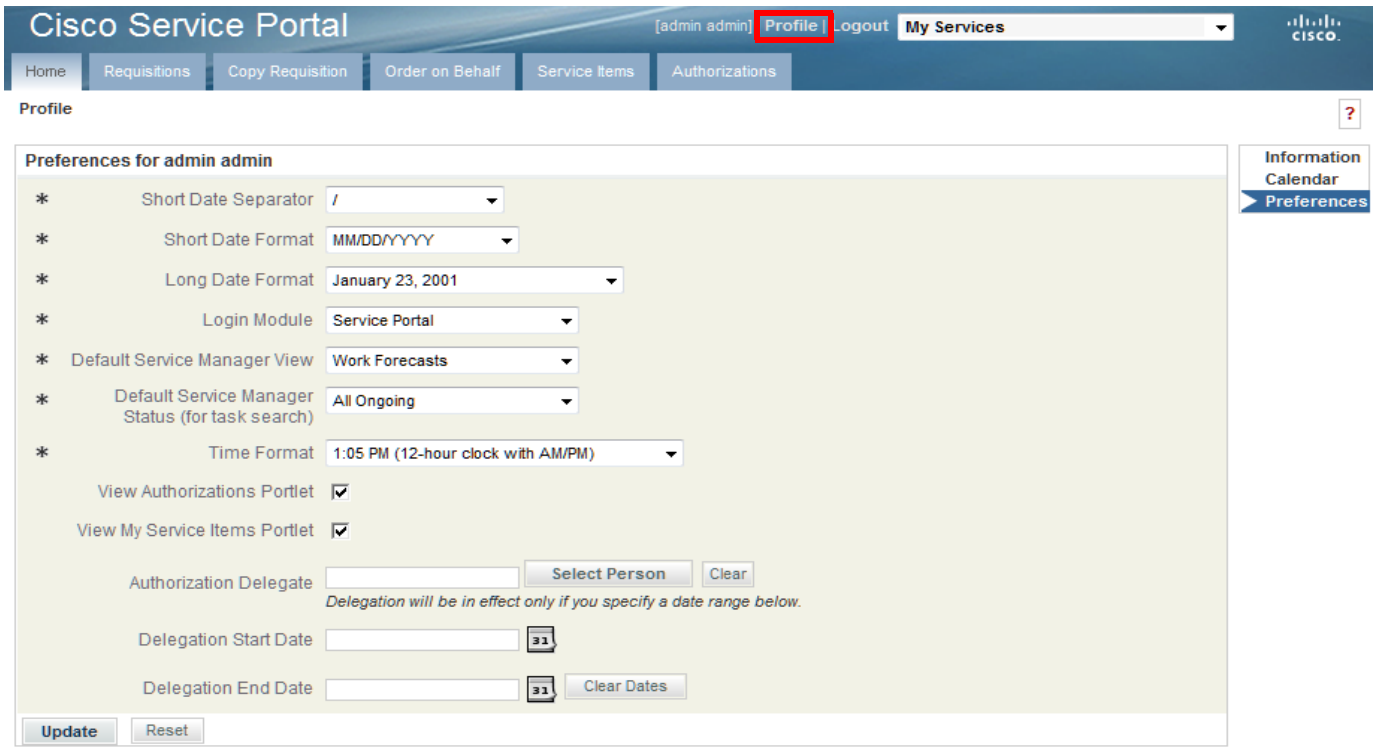
Service Item Details

The **Service Item Details** subtab displays details on the currently chosen service item.



User Preference for My Items Portlet

End users can choose to show or hide the "My Items" portlet from their My Services home page by unchecking the option to "View My Service Items Portlet" in the Preferences page of their personal profile. The profile is accessed by clicking the **Profile** link on the top of the My Services home page.



Importing Service Items and Standards

Overview

Once a service item has been defined, it is ready for use. However, before you deploy services referring to service items, you need to think about the lifecycle of those items. Have service items previously been tracked in an automated or semiautomated system external to Request Center? If so, you may want to import this legacy information into Request Center, so that Request Center can be used to automatically track service item provisioning and other updates.

Importing service item data is optional. It is typically done in the scenario outlined above, to initialize the Request Center knowledge base with service items previously assigned. Service item data can be imported at any time, to synchronize Request Center data with other records that may have been maintained externally.

In addition to importing service item data, the service item definition itself can be imported, either with or without corresponding data. This automates the definition of the service item, so that it does not have to be done manually.

Standards can also be imported using options similar to those available for service items—either the standard definition, the standard entries, or both can be imported.

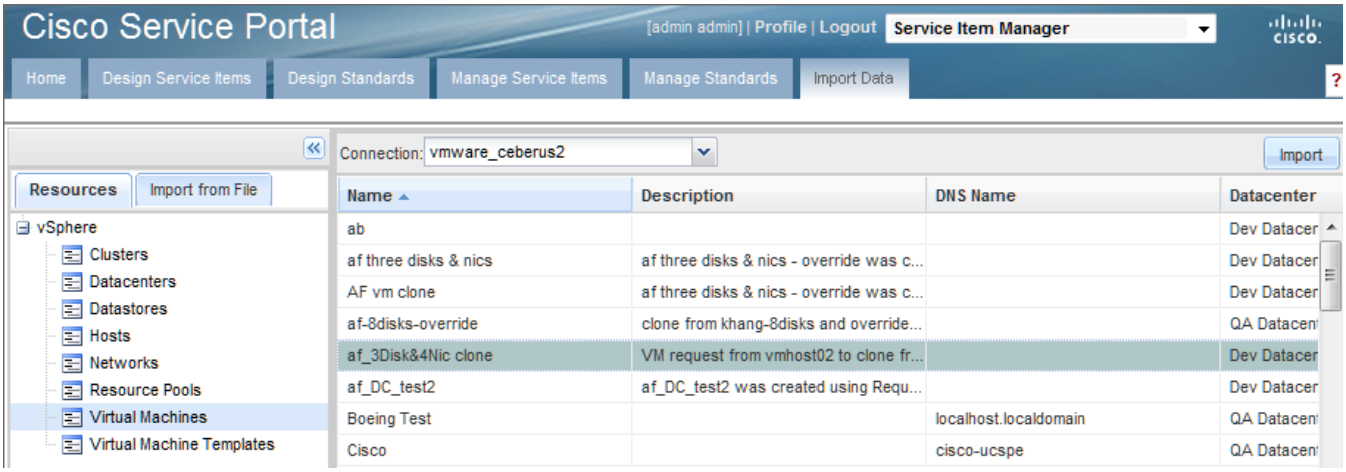
Lifecycle Center includes the following methods for importing service items and standards:

- The Import Data option in Service Item Manager allows administrators to import service items or standards on demand from file.
- A request can include a Service Link task to import service items or standards.
- Virtual Machine service items and standards can be imported from a vCenter instance.

Importing Virtual Machines and Standards from vCenter

You may be installing Request Center in an enterprise where vCenter has been in use for some time. In that case, you have VMs that have already been provisioned. In order to use Service Portal to maintain these VMs through the rest of their life cycle, you need to import them.

In Service Item Manager, use the vSphere connection utility on the **Import Data** tab to import data about these instances of the Virtual Machine service item.



- On the Import Data tab of Service Item Manager, choose the name of your vCenter instance from the Connection drop-down list. This drop-down list is populated with VMware adapter agents defined in the Service Link module.
- Choose the virtual machines to be imported by locating the Virtual Machine node on the left pane and highlighting one or more rows on the right-hand grid. At the top, right-click **Import** to import the machines chosen.

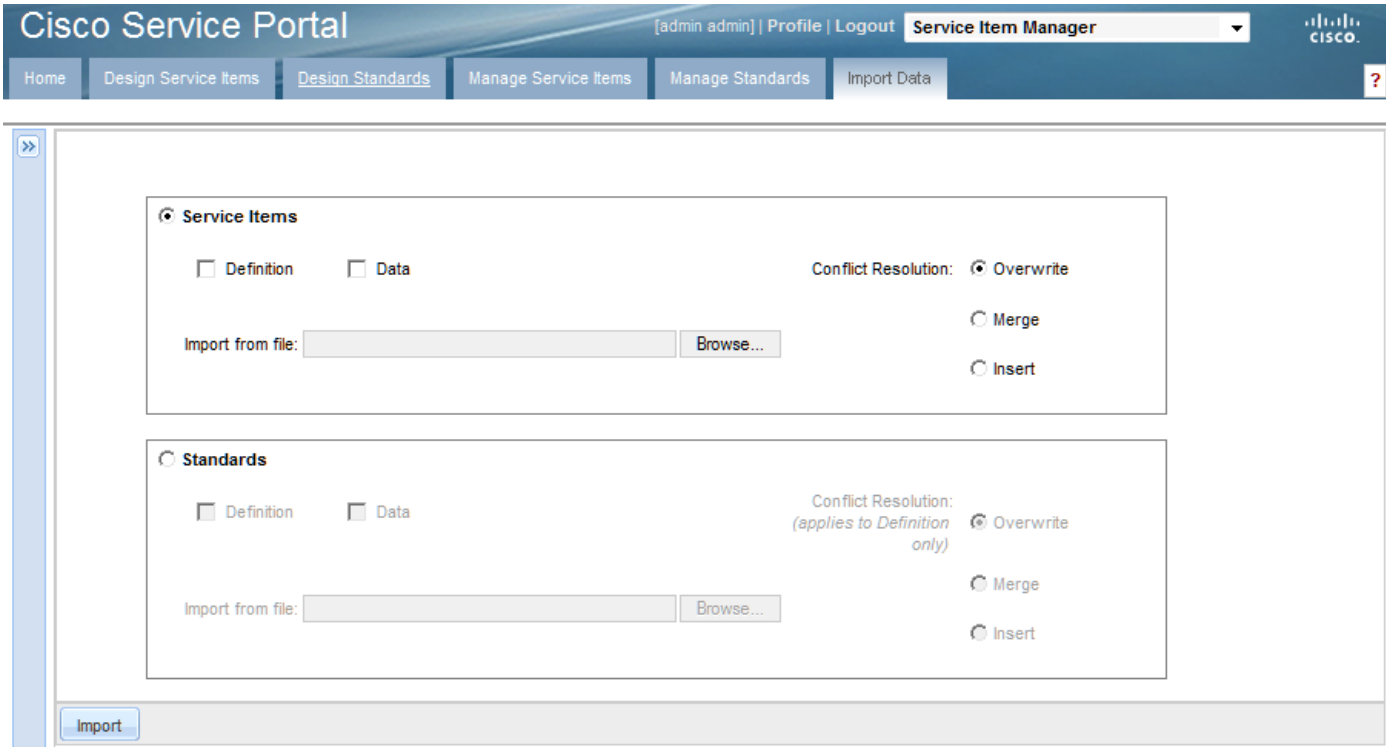
The chosen machines are added to the Virtual Machine service item table. Virtual machine instances imported from vCenter show the import date as the Assigned/Submitted date; the Owner is blank. Once you have entered your organization's personnel (either manually, or, more typically, via Directory Integration), you can highlight a virtual machine, click **Assign Owner**, and choose the machine's owner. This allows the machine's owner (or an administrator acting on behalf of the owner) to enter service requests that affect the status of this machine.

If a user (or administrator, acting on the user's behalf) orders a service to create a virtual machine, and that service request is fulfilled, the user name and the provision date will automatically be recorded and displayed on the Manage Service Item Instances page.

To import other vSphere entities such as datastores and hosts, choose the respective node on the left-hand pane and follow the same steps as mentioned above for importing virtual machines. The chosen instances are added to the corresponding Virtual Data Center standards.

Importing Service Items and Standards from File

In Service Item Manager, use the **Import from File** subtab to import data from an external source for service items and standards. The definition of the service item or standard can also be imported. The file to be imported must use ANSI encoding—either ASCII or UTF-8 will work. Unicode encoding is not supported.



The file can contain any combination of definition and data sections—both or just one or the other. The Import option imports only those portions of the field (definition or data) as specified.

If you choose the “Definition” option, but the XML file contains only the data section, nothing is imported. Similarly, if you choose the “Data” option, but the XML file contains only the definition section, nothing is imported.

If you choose to import both “Definition” and “Data”, and the XML file contains only a definition section, the system only imports the definition. If the XML file contains both definition and data sections, both are imported.

When service item data is imported, an entry is added to the Service Item History table and the Service Item Subscription is created or updated, as appropriate.

Importing Service Items

The table below summarizes the behavior of the Import Utility when importing service item definitions or data from file. This behavior also applies to a Service Link Service Item task, described later in this chapter.

Definition/Data	Conflict Resolution	Description
Definition	Overwrite	The import replaces the existing Service Item definition with the definition contained in the import. For an existing service item, existing data is preserved when column names between the old and new versions match.
	Merge	The import can be used to merge new columns into the existing Service Item definition. A new Service Item is created if the item does not already exist.

Definition/Data	Conflict Resolution	Description
	Insert	The import fails if the service item already exists. If the service item does not exist, the service item and underlying database table are created.
Data	Overwrite	Update (overwrite) any existing service items with a matching item found in the import; insert any new items in the import file. If the import file does not contain a value for an attribute of the service item, then the value of the attribute is set to null for that service item, even if the service item previously existed and the attribute had a value.
	Merge	Update any existing service items with a matching item found in the import; insert any new items found in the import file. There are two conditions in which the attributes of an existing service item will NOT be updated: (1) the import file does not contain the property tag for this attribute; (2) the service item attribute already had a non-null/zero value. In either of these cases, the existing attribute value will not be overwritten.
	Insert	Insert any new Service Items found in the import; existing Service Items are unchanged.

Importing Standards

When importing standards data, all existing data is always overwritten by imported data. The option to import standards data supplements or replaces Catalog Deployer's actions in deploying a service that includes a data retrieval rule that references a standard. By default, Catalog Deployer will deploy both the standard definition and any data previously defined in the source environment to the target environment. This behavior is desirable if standards data does not vary from environment to environment. If this is not the case, you may alter this default behavior by turning off the Administration setting to "Deploy Entries (data) in Standards Tables".

When importing a standards definition, the same conflict resolution options are available as for importing the definition of a service item, as explained above. The conflict resolutions applied to standards data are summarized below.

Definition/Data	Conflict Resolution	Description
Data	Overwrite	If a standard record exists whose attribute values match all attribute values of the standard being imported then this standard is updated so that standard has values ONLY for the attributes that are specified in the record being imported. This implies that if the standard record being imported has specified values for only some attributes BUT the existing standard in the database has values specified for additional attributes, then those values would be set to NULL.
	Merge	Same as Insert below.
	Insert	If a standard exists whose attribute values match all attribute values of the standard being imported then this standard is not created.

Import File Format

Each import file is an XML file in an industry-standard CIM (Common Information Model) compatible format, version 2.3.1. CIM is based on an object-oriented model and uses technology adapted from Unified Modeling Language (UML). A Document Type Definition (DTD) describing the file format used by Service Portal can be found at RequestCenter.war\DTD on the application server.

Service Portal recognizes only a subset of entities that are available under CIM. Those entities that it does not recognize it ignores. In Request Center's CIM implementation:

- Each service item or standard is a **class**.
- The details about the service item—its description, display name, and assigned group (classification)—are **qualifiers** for the service item.
- The attributes specified for the service item are **properties** of the service item class.
- Each attribute (property), in turn, has **qualifiers**. These qualifiers constitute the detailed configuration for each attribute—its description, caption, whether it is visible in My Services, and the sequence in which the attribute occurs in the service item definition.
- Each service item class may have many **instances**. Each of these corresponds to one service item instance.
- Each instance of a service item has one **property** for each attribute of the service item. You may also include, optionally, three attributes that comprise the subscription information for the service item during import—Requisition Entry ID, Customer Login Name, and Organizational Unit Name. The value in the XML file updates the value of the corresponding attribute or subscription field of the service item instance.

The previous discussion applies to both service items and standards in general. Note that standards are not "owned" by any individual user or organizational unit *per se* and therefore you cannot set subscription attributes for them.

Syntax for a Service Item Import File

The following table summarizing the syntax for specifying a service item or standards import file. The same syntax applies to both service items and standards.

Each XML file consists of one SI Import Specification.

The Import Specification (SIImportSpec), in turn, consists of a SIClassDefinition, followed by a list of one more SIInstances for the specified class (Service Item or standard). One or both of the class definition and instances may be included in the file. File content should match the import options specified.

Each SIClass specification consists of the service item/standard definition, embedded within appropriate XML tags, followed by the definition of each of the attributes.

Each attribute definition specifies the name, caption, and other configuration options specified for the attribute.

Each service item instance (SIInstance) specifies the name of the service item/standard to which the instance applies and a list of the values for each of the attributes of the item. If an attribute value is blank, appropriate tags must still be included in the import file, with no value specified for the attribute.

SIImportSpec =	S<?xml version="1.0" encoding="utf-8"?> <CIM CIMVERSION="" DTDVERSION=""> <DECLARATION> <DECLGROUP> SIClassDefinition SIInstanceList </DECLGROUP> </DECLARATION> </CIM>
SIClassDefinition =	<<VALUE.OBJECT> SIClass </VALUE.OBJECT>
SIClass =	<CLASS NAME="*Service Item Name*"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE></VALUE> </QUALIFIER> <QUALIFIER NAME="Classification" TYPE="string"> <VALUE>*Service Item Group*</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayName" TYPE="string"> <VALUE>*SI Display Name*</VALUE> </QUALIFIER> SIAttributeList </CLASS>
SIAttributeList =	SIAttributeDefinition [SIAttributeDefinition]

SIAttributeDefinition =	<PROPERTY NAME="Name" TYPE="string"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE>Description of the attribute</VALUE> </QUALIFIER> <QUALIFIER NAME="Caption" TYPE="string"> <VALUE>Caption for the attribute</VALUE> </QUALIFIER> <QUALIFIER NAME="IsVisible" TYPE="sint32"> <VALUE>1 if visible, 0 if not visible</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayOrder" TYPE="sint32"> <VALUE>Sequence of the attribute</VALUE> </QUALIFIER> </PROPERTY>
SIInstanceList =	SIInstance [SIInstance]
SIInstance =	<VALUE.OBJECT> <INSTANCE CLASSNAME="ClassName"> SIPropertyList </INSTANCE> </VALUE.OBJECT>
SIInstanceData List=	SIInstanceData [SIInstanceData]
SIInstanceData=	<PROPERTY NAME="AttributeName" TYPE=SIDataType> <VALUE>Attribute value</VALUE> </PROPERTY> Optional subscription properties: <PROPERTY NAME="#RequisitionEntryID#" TYPE="sint32"> <VALUE>Requisition entry ID</VALUE> </PROPERTY> <PROPERTY NAME="#LoginName#" TYPE="string"> <VALUE>Customer login name</VALUE> </PROPERTY> <PROPERTY NAME="#OrganizationalUnitName#" TYPE="string"> <VALUE>Organizational unit name</VALUE> </PROPERTY>
SIDataType =	{“char16” | “string” | “datetime” | “real64” | “sint32” | “sint64”}

The datatypes do not exactly match the datatypes specified when you define a Service Item or Standard using Service Item Manager. Map the Service Item Manager datatypes to those used in the import file using the following table.

Service Item Manager Datatype	Import File Datatype
STRING(32)	char16
STRING(128)	<not supported>
STRING(512)	string
INTEGER	sint32
LONGINTEGER	sint64
DOUBLEFLOAT	real64
MONEY	Use real64
DATETIME	Datetime (must use the following format for date value = yyyy-mm-dd hh:mm:ss, for example 2009-04-15 12:00:00)

Service Item Subscription Processing Rules

Unlike create and update operations performed by internal and external service item tasks, the create and update operations handled through the file import mechanism happen outside the context of a service request delivery plan. Hence the import program does not validate the relationship among the Requisition Entry ID, Customer Login Name, and Organizational Unit Name provided in the import file. As long as they are valid IDs/names, the input is accepted.

Here are the high-level processing rules for subscription:

- If no subscription information is provided when a service item instance is created, the item stays unassigned.
- If only Customer Login Name property is specified at the time a service item instance is created, the Organizational Unit owner of the item is set to the home organizational unit of the customer.
- If Customer Login Name or Organizational Unit name properties are specified, the values are used to update the service item subscription. When the value is blank, the corresponding subscription field is set to null. The absence of a property in the import file means no change/override on the property value.
- When a Requisition Entry ID is provided in the properties, the corresponding requisition ID is associated with the service item and displayed in My Services and Service Item Manager.
- Once a Requisition Entry ID is set for a service item, it cannot be modified or reset to null.

The possible combinations for customer and organizational unit assignment and the outcome are summarized in the table below.

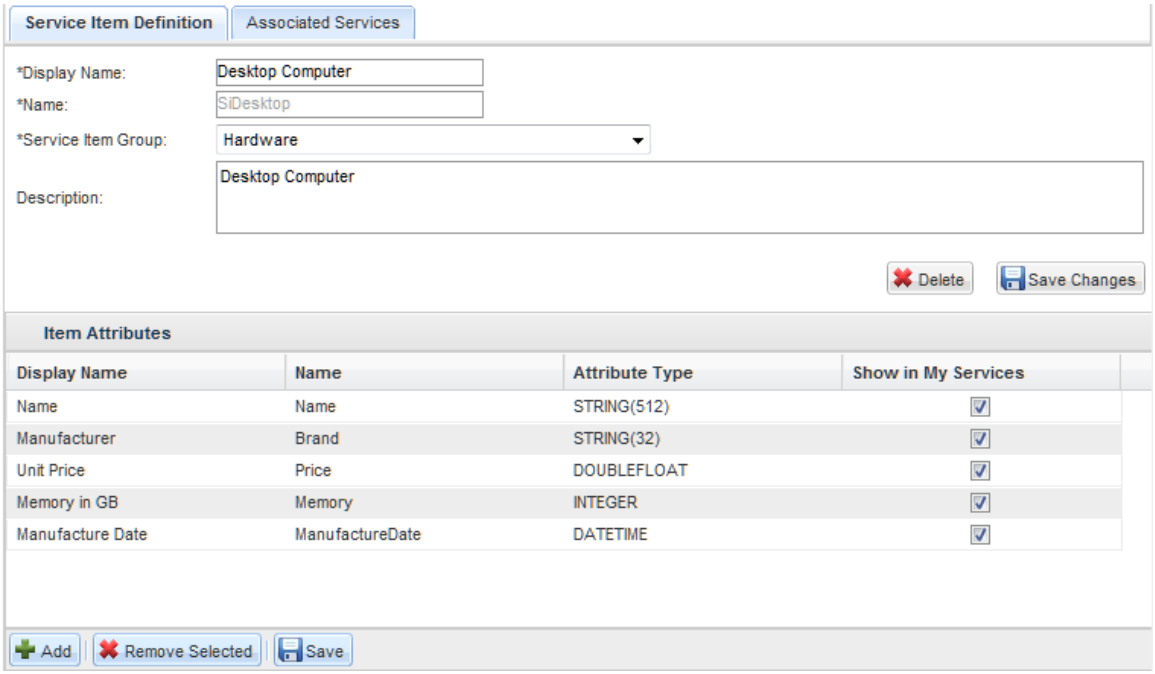
:

When service item is created			
Property In XML		**Resulting Subscription**	
Login Name	**OU Name**	**Customer**	**OU**
None	None	NULL	NULL
None	Blank or Invalid Value	NULL	NULL
None	Valid OU Name	NULL	OU provided
Blank or Invalid Value	None	NULL	NULL
Blank or Invalid Value	Blank or Invalid Value	NULL	NULL
Blank or Invalid Value	Valid OU Name	NULL	OU provided
Valid Customer	None	Customer provided	Home OU of Customer
Valid Customer	No Value	Customer provided	NULL
Valid Customer	Valid OU Name	Customer provided	OU provided

When service item is updated			
Property In XML		**Resulting Subscription**	
Login Name	**OU Name**	**Customer**	**OU**
None	None	No change	No change
None	Blank or Invalid Value	No change	NULL
None	Valid OU Name	No change	OU provided
Blank or Invalid Value	None	NULL	No change
Blank or Invalid Value	Blank or Invalid Value	NULL	NULL
Blank or Invalid Value	Valid OU Name	NULL	OU provided
Valid Customer	None	Customer provided	No change
Valid Customer	Blank or Invalid Value	Customer provided	NULL
Valid Customer	Valid OU Name	Customer provided	OU provided

Service Item Import File Example

Assume that a “Desktop” service item has been created, with the following definition:



The import file for this service item might look like the sample below:

Sample XML	Description/Usage
<?xml version="1.0" encoding="utf-8"?> <CIM CIMVERSION="" DTDVERSION=""> <DECLARATION> <DECLGROUP>	Beginning of SI Import Specification.
<VALUE.OBJECT> <CLASS NAME="Desktop"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE>Desktop Computer.</VALUE> </QUALIFIER> <QUALIFIER NAME="Classification" TYPE="string"> <VALUE>Hardware</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayName" TYPE="string"> <VALUE>Desktop Computer</VALUE> </QUALIFIER>	Properties of an SI—its name, description, display name, and group (classification).

Sample XML	Description/Usage
<PROPERTY NAME="Name" TYPE="string"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE>The name of the computer.</VALUE> </QUALIFIER> <QUALIFIER NAME="Caption" TYPE="string"> <VALUE>Name</VALUE> </QUALIFIER> <QUALIFIER NAME="IsVisible" TYPE="sint32"> <VALUE>1</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayOrder" TYPE="sint32"> <VALUE>1</VALUE> </QUALIFIER> </PROPERTY>	Definition of the Name attribute of the SI. The Name attribute is required in all SIs; its caption must also be "Name".
<PROPERTY NAME="Brand" TYPE="char16"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE>Brand name of the computer.</VALUE> </QUALIFIER> <QUALIFIER NAME="Caption" TYPE="string"> <VALUE>Manufacturer</VALUE> </QUALIFIER> <QUALIFIER NAME="IsVisible" TYPE="sint32"> <VALUE>1</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayOrder" TYPE="sint32"> <VALUE>2</VALUE> </QUALIFIER> </PROPERTY>	Definition of the Brand attribute of the SI.

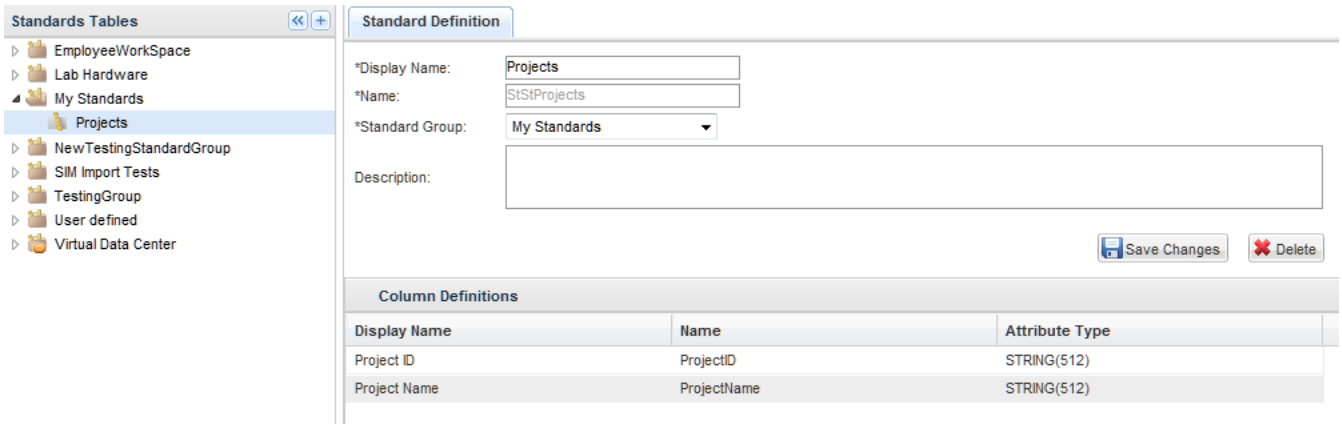
Sample XML	Description/Usage
<PROPERTY NAME="Price" TYPE="real64"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE>MSRP</VALUE> </QUALIFIER> <QUALIFIER NAME="Caption" TYPE="string"> <VALUE>Unit Price</VALUE> </QUALIFIER> <QUALIFIER NAME="IsVisible" TYPE="sint32"> <VALUE>1</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayOrder" TYPE="sint32"> <VALUE>3</VALUE> </QUALIFIER> </PROPERTY>	Definition of the Price attribute of the SI.
<PROPERTY NAME="Memory" TYPE="sint32"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE>Amount of RAM in GB</VALUE> </QUALIFIER> <QUALIFIER NAME="Caption" TYPE="string"> <VALUE>Memory in GB</VALUE> </QUALIFIER> <QUALIFIER NAME="IsVisible" TYPE="sint32"> <VALUE>1</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayOrder" TYPE="sint32"> <VALUE>4</VALUE> </QUALIFIER> </PROPERTY>	Definition of the Memory attribute of the SI.

Sample XML	Description/Usage
<PROPERTY NAME=“ManufactureDate” TYPE=“datetime”> <QUALIFIER NAME=“Description” TYPE=“string”> <VALUE>Date of manufacture</VALUE> </QUALIFIER> <QUALIFIER NAME=“Caption” TYPE=“string”> <VALUE>Manufacture Date</VALUE> </QUALIFIER> <QUALIFIER NAME=“IsVisible” TYPE=“sint32”> <VALUE>1</VALUE> </QUALIFIER> <QUALIFIER NAME=“DisplayOrder” TYPE=“sint32”> <VALUE>5</VALUE> </QUALIFIER> </PROPERTY> </CLASS> </VALUE.OBJECT>	Definition of the ManufactureDate attribute of the SI.

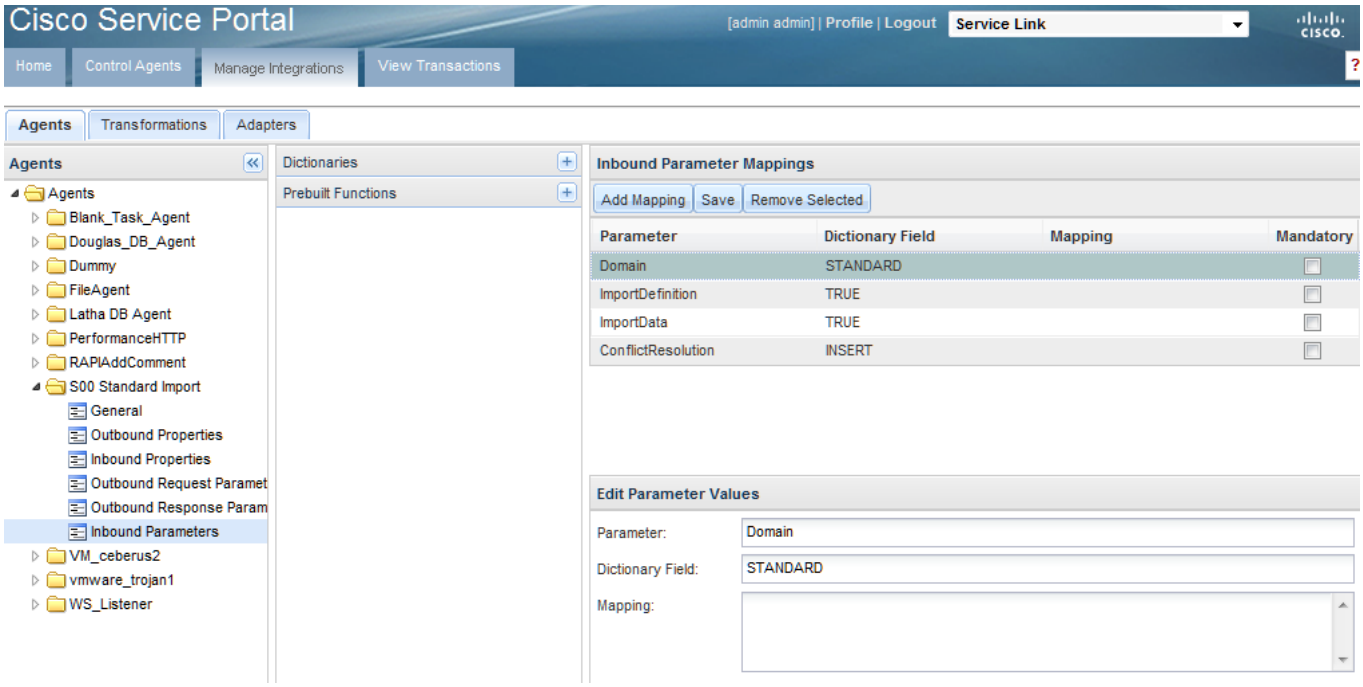
Sample XML	Description/Usage
<VALUE.OBJECT> <INSTANCE CLASSNAME=“Desktop”> <PROPERTY NAME=“Name” TYPE=“string”> <VALUE>Thinkpad T60</VALUE> </PROPERTY> <PROPERTY NAME=“Brand” TYPE=“char16”> <VALUE>LENOVO</VALUE> </PROPERTY> <PROPERTY NAME=“Price” TYPE=“real64”> <VALUE>899.99</VALUE> </PROPERTY> <PROPERTY NAME=“Memory” TYPE=“sint32”> <VALUE>3</VALUE> </PROPERTY> <PROPERTY NAME=“ManufactureDate” TYPE=“datetime”> <VALUE>2009-04-15 12:00:00</VALUE> </PROPERTY> <PROPERTY NAME="#RequisitionEntryID#" TYPE="sint32"> <VALUE>10</VALUE> </PROPERTY> <PROPERTY NAME="#LoginName#" TYPE="string"> <VALUE>jsmith</VALUE> </PROPERTY> <PROPERTY NAME="#OrganizationalUnitName#" TYPE="string"> <VALUE>Finance</VALUE> </PROPERTY> </INSTANCE> </VALUE.OBJECT>	One instance of a Desktop SI, for a “Thinkpad T60”.
</DECLGROUP> </DECLARATION> </CIM>	End of the SI Import Specification.

Standard Import File Example

Assume that a “Projects” standard has been created, with the following definition:



The Service Link Service Item Task would be defined with the following agent parameters:



The import file for this standard might look like the sample below:

Sample XML	Description/Usage
<?xml version="1.0" encoding="utf-8"?> <CIM CIMVERSION="" DTDVERSION=""> <DECLARATION> <DECLGROUP>	Beginning of Standard Import Specification.
<VALUE.OBJECT> <CLASS NAME="Projects"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE>Describe it here</VALUE> </QUALIFIER> <QUALIFIER NAME="Classification" TYPE="string"> <VALUE>My Standards</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayName" TYPE="string"> <VALUE>Projects</VALUE> </QUALIFIER>	Properties of the Standard—its name, description, display name, and group (classification).
<PROPERTY NAME="ProjectID" TYPE="string"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE>ID of the Project</VALUE> </QUALIFIER> <QUALIFIER NAME="Caption" TYPE="string"> <VALUE>Project ID</VALUE> </QUALIFIER> <QUALIFIER NAME="IsVisible" TYPE="sint32"> <VALUE>1</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayOrder" TYPE="sint32"> <VALUE>1</VALUE> </QUALIFIER> </PROPERTY>	Definition of the ProjectID attribute of the Standard.

Sample XML	Description/Usage
<PROPERTY NAME="ProjectName" TYPE="string"> <QUALIFIER NAME="Description" TYPE="string"> <VALUE>Name of the project</VALUE> </QUALIFIER> <QUALIFIER NAME="Caption" TYPE="string"> <VALUE>Project Name</VALUE> </QUALIFIER> <QUALIFIER NAME="IsVisible" TYPE="sint32"> <VALUE>1</VALUE> </QUALIFIER> <QUALIFIER NAME="DisplayOrder" TYPE="sint32"> <VALUE>2</VALUE> </QUALIFIER> </PROPERTY>	Definition of the ProjectName attribute of the Standard.
<VALUE.OBJECT> <INSTANCE CLASSNAME="Projects"> <PROPERTY NAME="ProjectID" TYPE="string"> <VALUE>001</VALUE> </PROPERTY> <PROPERTY NAME="ProjectName" TYPE="string"> <VALUE>Reporting System Upgrade</VALUE> </PROPERTY> </INSTANCE> </VALUE.OBJECT>	First instance of a Project Standard.

Sample XML	Description/Usage
<VALUE.OBJECT> <INSTANCE CLASSNAME="Projects"> <PROPERTY NAME="ProjectID" TYPE="string"> <VALUE>002</VALUE> </PROPERTY> <PROPERTY NAME="ProjectName" TYPE="string"> <VALUE>Decision Support System Installation</VALUE> </PROPERTY> </INSTANCE> </VALUE.OBJECT>	Second instance of a Project Standard.
</DECLGROUP> </DECLARATION> </CIM>	End of the Standards Import Specification.

Importing Service Items and Standards using Service Link

You can use a Service Link agent to import either service items or standards. This allows you to include the import step as a task in a standard service request. The agent will import a file formatted as described above, and supports the same import options as are available via the Import from File utility. The message is logged in Service Link as a "SIM Import" message type.

For detailed instructions on using Service Link, see the *Cisco Service Portal Integration Guide*. To configure a Service Link agent to import service items or standards:

1. Define a new agent in Service Link, specifying a "Service Item" task as the Context Type. At a minimum, you will need separate agents for standards and service items.
2. Specify the "Dummy" adapter as the outbound adapter, and the "File" adapter as the inbound adapter. No transformations are required.
3. Skip the pages on configuring the outbound adapter properties and its parameters.
4. For the inbound adapter properties, specify the names of the directories to be used for processing the import file. The file should be written to the designated "Input" directory.
5. For the inbound parameters, create four parameters and enter the appropriate value in the Dictionary Field as shown in the table below. (Parameter values are case sensitive.)

Parameter Name	Valid Values	Description
Domain	SERVICE ITEM STANDARD	An import file can contain data or definition of either a standard or service item.
ImportDefinition	TRUE FALSE	True if the definition portion (if any) of the import file should be processed; false otherwise.

Parameter Name	Valid Values	Description
ImportData	TRUE FALSE	True if the data portion (if any) of the import file should be processed; false otherwise.
ConflictResolution	INSERT OVERWRITE MERGE	Behavior of the import process in reconciling new data or definitions, in the input file, with data or definitions currently in Request Center.

Service Item Import DTD

The Document Type Definition (DTD) that describes the format required for a Service Item or Standards import file is available on the Request Center application server, under RequestCenter.war\DTD. The DTD is given below.

```
<!ENTITY % CIMName "NAME           CDATA         #REQUIRED">
<!ENTITY % CIMType "TYPE (char16|string|sint32|sint64|datetime|real64)">
<!ENTITY % ClassName "CLASSNAME      CDATA         #REQUIRED">

<!ELEMENTCIM (DECLARATION)>
<!ATTLISTCIM
    CIMVERSION CDATA #REQUIRED
    DTDVERSION CDATA #REQUIRED
>

<!ELEMENTDECLARATION (DECLGROUP)+>

<!ELEMENTDECLGROUP (VALUE.OBJECT*)>

<!ELEMENTVALUE.OBJECT (CLASS | INSTANCE)>

<!ELEMENTCLASS (QUALIFIER*, PROPERTY*, METHOD*)>
<!ATTLISTCLASS
    %CIMName;
>

<!ELEMENTQUALIFIER (VALUE?)>
<!ATTLISTQUALIFIER %CIMName;
         %CIMType;              #REQUIRED
>

<!ELEMENTPROPERTY (QUALIFIER*, VALUE?)>
<!ATTLISTPROPERTY %CIMName;
         %CIMType;              #REQUIRED
>
<!ELEMENTVALUE (#PCDATA)>

<!ELEMENTINSTANCE (QUALIFIER*, PROPERTY*)>
<!ATTLISTINSTANCE
    %ClassName;
>

<!ELEMENTMETHOD (QUALIFIER*)>
<!ATTLISTMETHOD %CIMName;
>
```

Best Practices

What is a Service Item?

As is always the case when designing a dictionary, you need to think of all the requests in which it might be used. The dictionary should incorporate all fields that may be required in these requests—field appearance or usage can always be customized in the active form components.

This principle (of designing a dictionary and form component for reuse) is particularly important for service item-based dictionaries, since services may be available for the entire lifecycle of the service item, not just the initial request to provide a service (or item) to the customer. You may want to design the dictionary to support the service item tasks that can be included in a service request: create the service item; modify the service item; or remove the service item from its current owner.

Data Integration Options with Lifecycle Management

Some service items you create and track in Service Portal may already be tracked to some degree in an external Asset Management System (AMS), or in a target platform such as a hypervisor. Laptops and workstations, as well as virtual machines and virtual applications, are good examples of such service items. For service items such as these, Service Portal offers a number of alternatives for data integration. The particular alternative you choose depends on a number of factors, including the maturity of the external system, the nature and quality of data in that system, the frequency by which the system changes and whether there are events in the system that you wish to have trigger changes in Service Portal.

There is also the matter of which attributes (that is, data elements) of service items you want the end users of Service Portal to see. For example, things like the Manufacturer, Make, and Model of a laptop; or the GUID of a virtual machine, are not going to change and therefore don't necessarily need to be stored and updated in Service Portal. However, having that data readily visible to the end-user can help him make decisions about follow-on services to order.

Using the Laptop example, and assuming you have an external Asset Management System, here are the possible strategies you may wish to adopt:

1. **Store the Asset ID only in Service Portal**. The only attribute for the Laptop SI is the Name (into which you write the Asset ID from the external system). The end-user in My Services sees only *that* (that is, the fact that he has a laptop, and that laptop's Asset ID). When this laptop is referred to in service requests, you use one of two methods to pull the current data (Make, Model, Manufacturer, Disk Space, and Memory) from the external system onto the form:
   - A data retrieval rule queries the external database, based upon the Asset ID
   - You use a Service Link database or web services adapter to get data from the external system in an inbound message
2. **Store the Asset ID, plus attributes that won't change, in Service Portal**. Assume the attributes for the Laptop SI are the Name (Asset ID), Make, Model, and Manufacturer. An initial Asset ID lookup brings those attributes onto the form from the external system, and stores them in the Service Portal database. When this laptop is referred to in service requests, you use one of the two methods mentioned above to pull current data for Disk Space and Memory from the external system onto the service form. (In other words, Disk Space and Memory are never stored in Service Portal.)

3. **Same as #1 or #2, but enhance the service item data with business-contextual data**. This is really just a variation on the two approaches listed above. The business data gathered during the processes of requesting services and fulfilling those requests is captured and written as service item attributes in Service Portal. This data is never stored in the external system.
4. **Store all attributes in Service Portal, the single "system of record"**. An initial load (via Import) is done to populate Service Portal with all laptops. From then on, everything affecting laptops is handled via Service Portal service requests. *Now* the question becomes updating the external system. You could do this either manually or via a Service Link agent.
5. **Store all attributes in Service Portal; changes may occur both in Service Portal and in the external system**. As in #4 above, but there may be events occurring in the external system that need to be reflected in Service Portal. For example, you may not want to consult the external system to see the list of laptops currently in inventory (that is, not assigned to anyone); maybe you want all those stored in Service Portal for easier data retrieval. In this case, you could use the Import API or Service Item Listener Adapter to handle updates from the external system to Service Portal.

All of the above approaches are available in Service Portal. Data retrieval rules provide a form-data-level integration that is easily configured by service designers. Service Link provides a "process API" that is easily built into the workflows for services. Finally, the web services API, although not mentioned above, is yet another method for initiating processes in Service Portal from an external system; it could be used to submit requests whose sole purpose is to update the Service Portal data.

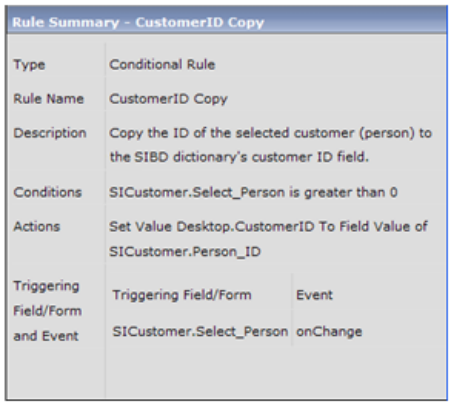
Using SIBDs Rather than Order-on-Behalf

The Service Portal order-on-behalf process allows a service request initiator to order a service on behalf of another person, the intended customer. A drawback of this approach is that the intended customer must have ordering permissions for the service. This contradicts some corporate policies where, for example, IT or administrative personnel should be able to order services for other people, but people not in the IT or administrative roles should not be able to order these services for themselves.

Without service items and service item-based dictionaries, a workaround for this situation is possible but hardly ideal. An initiator could choose the person for whom the service was intended (using a person-based dictionary), but there was no way for that person to be able to track "his" requisitions. This situation is remedied by the use service items and service item based dictionaries. An initiator can order a service (and provision a service item) and designate a chosen person as the customer. The customer would then be able to monitor the request in the form of a service item that would appear in his Service Items page.

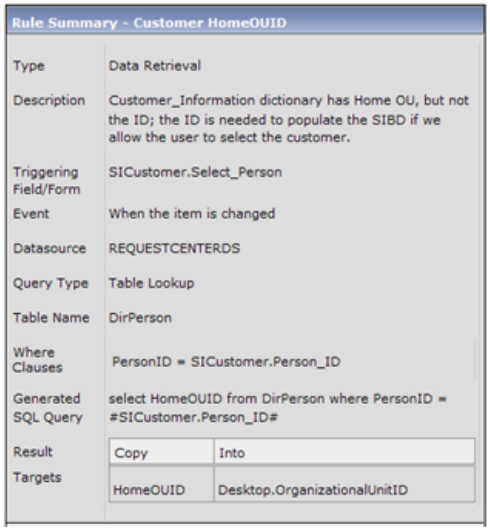
The procedure for implementing this functionality is fairly straightforward:

- Design the service item.
- Create the service item-based dictionary, being sure to include in the dictionary the CustomerID and OrganizationalUnitID. Optionally include a user-defined Status field, which can be updated as the delivery plan progresses.
- Include the SIBD in an active form component.
- Create a person-based dictionary to allow the initiator to choose the customer for the service item. Include this dictionary either in the same active form component as the SIBD or another form component—as long as both dictionaries are included in the service.
- In the active form component containing the person-based dictionary write two rules as shown below, to copy information on the chosen person to the service item-based dictionary.



Rule Summary - CustomerID Copy

Type	Conditional Rule
Rule Name	CustomerID Copy
Description	Copy the ID of the selected customer (person) to the SIBD dictionary's customer ID field.
Conditions	SICustomer.Select_Person is greater than 0
Actions	Set Value Desktop.CustomerID To Field Value of SICustomer.Person_ID
Triggering Field/Form and Event	Triggering Field/Form: SICustomer.Select_Person; Event: onChange



Rule Summary - Customer HomeOUID

Type	Data Retrieval
Description	Customer_Information dictionary has Home OU, but not the ID; the ID is needed to populate the SIBD if we allow the user to select the customer.
Triggering Field/Form	SICustomer.Select_Person
Event	When the item is changed
Datasource	REQUESTCENTERDS
Query Type	Table Lookup
Table Name	DirPerson
Where Clauses	PersonID = SICustomer.Person_ID
Generated SQL Query	select HomeOUID from DirPerson where PersonID = #SICustomer.Person_ID#
Result Targets	Copy: HomeOUID; Into: Desktop.OrganizationalUnitID

An administrator is now able to order the service for various customers, and the customers can see the service item in the Service Items page or My Items portlet (provided they have been granted the My Services 360-Degree Consumer or Professional role.) If the service item has associated services for which the owners of the SIs have ordering permission, they are able to order those services directly from the Service Items page.

A drawback of this approach is that the requisition would not be searchable by “Customer”, within Service Portal’s online modules, including Reporting and Advanced Reporting. In Advanced Reporting, the Service Item’s Customer replacement dictionary (in the example above, SICustomer) would need to be reportable.

Using Service Items for Capacity Management

Sometimes, the provisioning of service items of must be closely monitored. For example, the license for a particular software application might limit usage to a maximum number of named users. Or, the data center may be restricting the number of servers or server capacity available to a particular business unit or in a particular physical location.

Nonorderable service items could be used in situations like these. Consider tracking the assignment of software licenses. A service item “Software License Capacity” could be defined.

Administration

Overview

Site administrators can use the Organization Designer module to grant user’s access to Service Item Manager and the Administration module to configure how end users can access service items in My Services.

Roles for the Service Item Manager Module

Roles and capabilities, managed with the Organization Designer module, allow site administrators to control access to functionality provided by the Service Item Manager module. The Service Item Manager role will allow administrators to view the service items assigned to all Service Portal users.

Standard roles relating to Service Item Manager and the capabilities included in each are summarized in the table below.

		Role			
Capability	**Description**	**Service Item Manager**	**Service Item Designer**	**Service Item Admin.**	**Service Standards Manager**
Manage Standards Definitions	Allows the user to define and manage standards, including adding and deleting entries		X	X	
Manage Standards Data	Allows the user to add and delete entries from standards tables		X	X	X
Manage Service Item Definitions	Allows the user to define new service items and their attributes		X	X	
Manage Service Item Instances	Allows the user to create, update, and delete instances of service items (including the ability to import items, where applicable)	X		X	
Import Service Item and Standards Data	Use the import options to import service item and standards data and definitions			X	

The Service Item Manager roles should be assigned sparingly. Granting a user the capability to manage standards data or define standards means that user can edit all standards definitions and all standards data. A similar caveat applies to the capabilities controlling service item definitions and data.

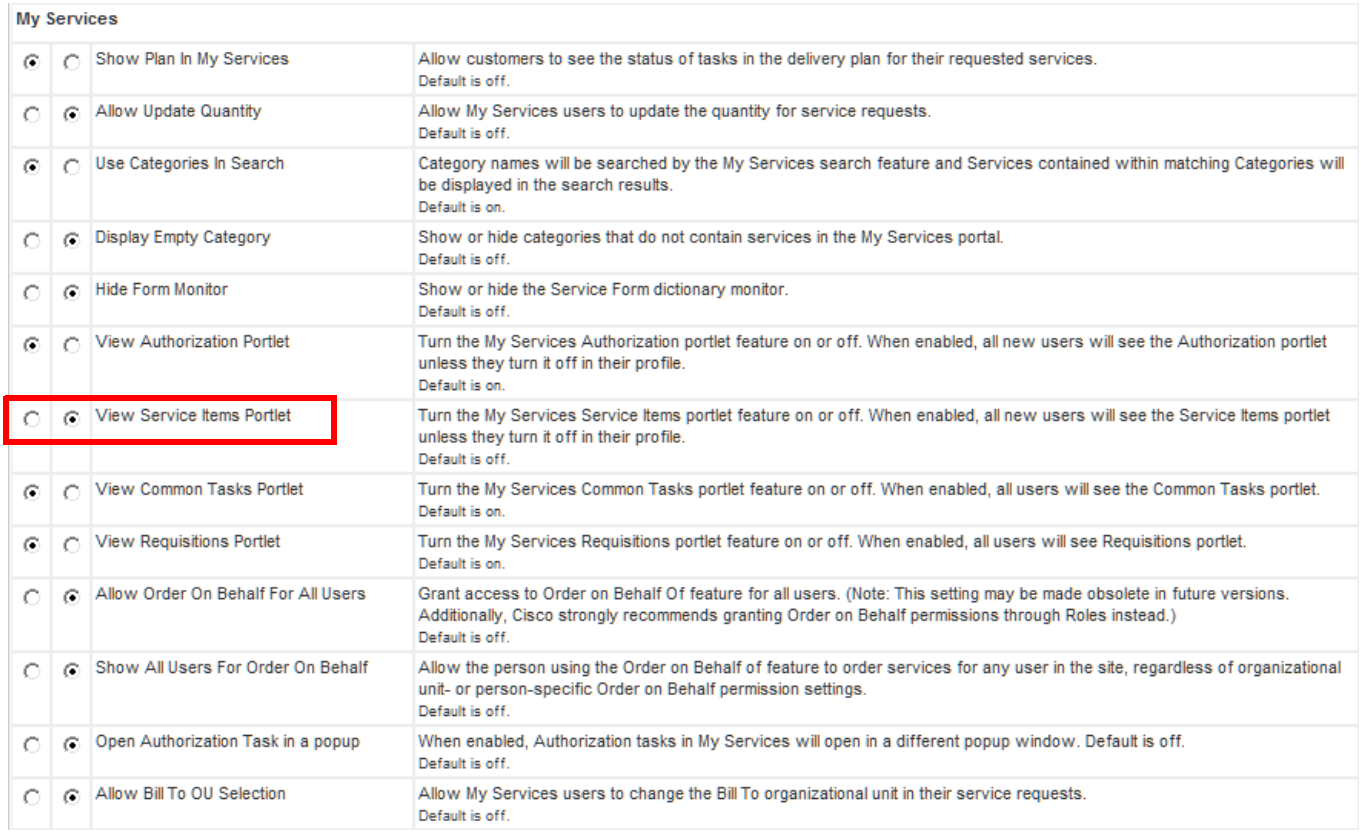
Configuring Access to My Service Items

Access to the Service Items tab and Service Items portlet in My Services can be controlled in three ways:

- Global access for all users must be turned on via an Administration setting.
- Once the ability to view Service Items has been activated, users with appropriate roles are able to view the My Items portlet and the Service Items tab.
- Individual users can choose to hide the portlet via a Profile setting.

Administration Setting for My Items Portlet

For any users to view the My Services module's My Items portlet, the Administration module setting "View Service Items Portlet" must be turned on. To review or modify this setting, choose **Administration > Settings > Customizations**. Look in the My Services section for the "View Service Items Portlet" setting, as shown below.



My Services			
◉	○	Show Plan In My Services	Allow customers to see the status of tasks in the delivery plan for their requested services. Default is off.
○	◉	Allow Update Quantity	Allow My Services users to update the quantity for service requests. Default is off.
◉	○	Use Categories In Search	Category names will be searched by the My Services search feature and Services contained within matching Categories will be displayed in the search results. Default is on.
○	◉	Display Empty Category	Show or hide categories that do not contain services in the My Services portal. Default is off.
○	◉	Hide Form Monitor	Show or hide the Service Form dictionary monitor. Default is off.
◉	○	View Authorization Portlet	Turn the My Services Authorization portlet feature on or off. When enabled, all new users will see the Authorization portlet unless they turn it off in their profile. Default is on.
○	◉	View Service Items Portlet	Turn the My Services Service Items portlet feature on or off. When enabled, all new users will see the Service Items portlet unless they turn it off in their profile. Default is off.
◉	○	View Common Tasks Portlet	Turn the My Services Common Tasks portlet feature on or off. When enabled, all users will see the Common Tasks portlet. Default is on.
◉	○	View Requisitions Portlet	Turn the My Services Requisitions portlet feature on or off. When enabled, all users will see Requisitions portlet. Default is on.
○	◉	Allow Order On Behalf For All Users	Grant access to Order on Behalf Of feature for all users. (Note: This setting may be made obsolete in future versions. Additionally, Cisco strongly recommends granting Order on Behalf permissions through Roles instead.) Default is off.
○	◉	Show All Users For Order On Behalf	Allow the person using the Order on Behalf of feature to order services for any user in the site, regardless of organizational unit- or person-specific Order on Behalf permission settings. Default is off.
○	◉	Open Authorization Task in a popup	When enabled, Authorization tasks in My Services will open in a different popup window. Default is off. Default is off.
○	◉	Allow Bill To OU Selection	Allow My Services users to change the Bill To organizational unit in their service requests. Default is off.

Turn this setting on to allow users to see the My Items portlet on the My Services home page, or off to omit the portlet from the page.

Roles for the Service Items Tab and My Items Portlet

RBAC capabilities control the access to the service item information for an individual. Roles are also available out of the box for users who need to have such access:

Role	Description
My Services 360-Degree Consumer	Enables end-users to access those portions of the Service Catalog to which they are entitled, to view service component details or initiate requests for services, and to view and manage the service items delivered to themselves as part of catalog services.
My Services 360-Degree Professional	Enables end-users to access those portions of the Service Catalog to which they are entitled, to view service component details or initiate requests for services, and to view and manage the service items delivered to anyone in their business units as part of catalog services.

The My Items portlet and Service Items tab only appear if the end user has been granted the “My Services 360-Degree Consumer” or “My Services 360-Degree Professional” role or any custom role that contains the “View My Service Items”, “View Service Items for My Business Units” or “View Service Items for My Business Units and their sub-units” capability.

Database Administration

All data and configuration details relating to service items are maintained in the Metadata Repository (MDR). The MDR is implemented as a set of database tables within the Service Portal transactional database.

Service Items

Each service item type has a corresponding table in the transactional database that stores service item instances. The name of the table is the service item name as specified when the service item is defined plus the prefix “Si” for “Service Item”. Once a service item has been saved, its name cannot be changed. The display name can be modified. However, any data retrieval rules based on that service item would be invalidated.

Each attribute of the service item corresponds to a column in the database table.

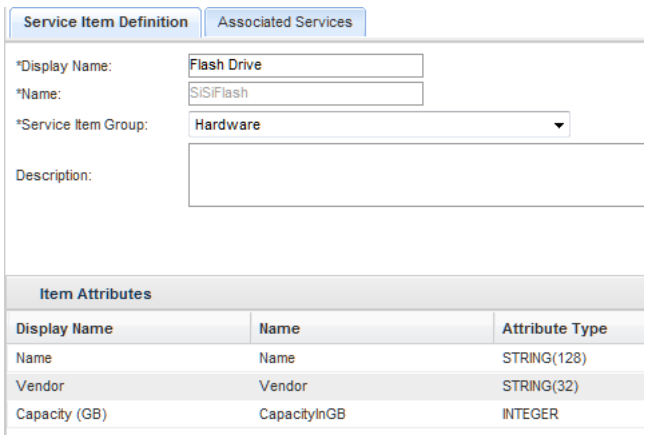


Table Name:	SiFlashDrive
Column Names:	PrimaryID OwnerDataTypeID OwnerID Name Vendor CapacityInGB

In addition to columns corresponding to the attributes specified for the service item, Service Portal adds three columns used for processing and retrieving service items:

Column	Description
PrimaryID	Identity column
OwnerDataTypeID	Not used
OwnerID	Not used

The Name column/attribute, which is required in all service items, must be unique within the service item. An index enforcing uniqueness (and facilitating retrieval by name) is automatically created.

The Service Portal datatypes specified for each column are implemented in database-specific datatypes, as shown in the table below.

Service Portal Datatype	Oracle Datatype	SQLServer Datatype
SHORT_STRING	VARCHAR2(32)	NVARCHAR(32)
MEDIUM_STRING	VARCHAR2(128)	NVARCHAR(128)
LONG_STRING	VARCHAR2(512)	NVARCHAR(512)
INTEGER	INTEGER	INT
MONEY	NUMBER(38,6)	MONEY
LONG	NUMBER(38)	NUMERIC (38)
DOUBLE	NUMBER(38,5)	NUMERIC(38,5)
DATETIME	DATE	DATETIME

Standards

Like service items, each standard has a corresponding table in the transactional database that stores standards data. The name of the table is the standard name as specified when the standard is defined plus the prefix “St” for “Standard”. Once a standard has been saved, its name cannot be changed. The display name can freely be modified.

Service Items, History, and Subscriptions

You should generally not have to write a SQL query using the service items, history and subscriptions table. Most of this information is provided to users via the My Items portlet and page. However, it is possible to write a data retrieval rule retrieving information from these tables. For example, perhaps you don’t want a user to request a service which provisions a service item if he already has one.

In order to write SQL-based data retrieval rules, you need to know a little bit about the table structure.

- As specified above, the name of the table corresponding to a service item is the item’s name prefixed by “Si”. So, an SI with the display name **State of the Art Laptop**, with a table name of **StateOfTheArtLaptop** has a corresponding table in the database named **SiStateOfTheArtLaptop**.
- How is that table related to ServiceItemSubscription? The easiest way to join it is on ServiceItemTypeName. So, if I want to get the subscription information for a State of the Art Laptop named **bfine-nov13-2009**, this query works:

  ```
  select * from SiServiceItemSubscription
   where ServiceItemTypeName = 'State of the Art Laptop'
     and DisplayName = 'bfine-nov13-2009'
  ```

  Or, I could get a list of all State of the Art Laptops by executing this query:

  ```
  select*from siserviceitemsubscription sis
  join siStateoftheArtLaptop al
  on al.PrimaryID = sis.ServiceItemID
  where sis.serviceitemtypename ='State of the Art Laptop'
  ```

- And then how do I get from there to the ServiceItemHistory records? Well, the key is the **PrimaryID** in the Service Item Subscription equals the **OwnerID** in the Service Item History. So:

  ```
  select sih.*, sis.*from SiServiceItemHistory sih
  join siServiceItemSubscription sis
  on sih.OwnerID = sis.PrimaryID
  where sis.ServiceItemTypeName ='State of the Art Laptop'
  and sis.DisplayName ='bfine-nov13-2009'
  ```

VMware Adapter Error Messages

The following table lists the error messages that may be returned by the VMware adapter in the ErrorDescription field of a virtual machine dictionary. The table explains each error and provides a recommended action.

You may wish to treat some of these errors as "benign" in your service design, as they will have no consequence for the end-user. For example, the "Invalid power state specified" error for a "poweron" operation on a virtual machine that is already powered on is one that can be ignored in the delivery of the service.

Operation	Sub-Operation	vSphere Error Description	Explanation	Recommended Action
All		Could not initialize vmware connection : RemoteException	The VM agent cannot establish a connection to vSphere server, due to one of the following problems: vSphere server is not responding, the URL specified is incorrect, trusted SSL handshake is unsuccessful.	Check one of the followings: vSphere server is running, the URL specified in the Service Link agent is spelled correctly, signer certificate for vSphere exist in the trusted CA keystore.
All		Could not initialize vmware connection : InvalidLogin	VM agent cannot establish a connection to vSphere because the login name or password is incorrect.	Check that the vSphere login information specified in the Service Link agent is spelled correctly.
All		Invalid operation specified	Invalid operation code is specified.	Check that the Operation code specified in the request is spelled correctly, and is one of the supported operations. Operation codes are listed in the "VMOperation" Standard.
All		Could not initialize vmware connection : NoPermission	The login used for vSphere connection does not have the necessary permission to perform the operation specified.	Change the login in the VM Agent configuration to one that has the necessary permission to perform the operation specified, or grant the permission to the user in VIM Client. See the VM Client user guide for the user roles and permissions.
All		Missing mandatory values : Datacenter. Cannot complete specified operation	Datacenter name is not specified.	Check that the Datacenter name is not left blank in the request.
All		Could not find datacenter : <*Datacenter_Name*>	Datacenter name cannot be found in the vSphere server.	Check that the Datacenter name specified in the request is spelled correctly, and exists in the target vSphere server.

Operation	Sub-Operation	vSphere Error Description	Explanation	Recommended Action
All		Virtual Machine Name cannot be empty	Virtual machine name is required for all operations.	Check that the Virtual Machine name is not left blank in the request.
All operations, except for create, clone, coldClone		Could not find virtual machine : <*VirtualMachine_Name*>	Virtual machine name cannot be found in the vSphere server.	Check that the Virtual Machine name specified in the request is spelled correctly, and exists in the Datacenter specified.
All operations, except for create		Invalid power state specified	The virtual machine or template specified in the request is in a power state that does not support the requested operation. For example, if machine is already powered off, it cannot be rebooted.	Legitimate processing error for end user's follow up actions (fix request or fix VM environment).
All operations, except for create, clone, coldClone		NotSupported	Certain operations can only be performed on a virtual machine not on a template.	Legitimate processing error for end user's follow-up actions (fix request).
create, clone, coldClone		error: The name '<*VirtualMachine_Name*> ' already exists.	The name of the virtual machine to be created already exists and cannot be used for a new machine.	Legitimate processing error for end user's follow up actions (fix request or fix VM environment).
create, clone, coldClone		Could not find host : <*HostName*>	The host cannot be found in the vSphere server under the Datacenter specified.	Check that the host name specified in the request is spelled correctly, and exists in the Datacenter specified along with it.
create, clone, coldClone		Missing mandatory values : Host Name. Cannot complete specified operation	Host name is required for create/clone virtual machine operations.	Check that the host name is not left blank in the request.
clone, coldClone		Missing mandatory values : Template Name. Cannot complete specified operation	Template name is required for clone virtual machine operation.	Check that the Template name is not left blank in the request.
clone, coldClone		Could not find virtual machine : <*Template_Name*>	Template name cannot be found in the vSphere server.	Check that the Template name specified in the request is spelled correctly, and exists in the Datacenter specified.
create, clone, coldClone		Invalid Resource Pool ID specified	Resource Pool ID cannot be found in the vSphere server.	Check that the Resource Pool ID specified in the request is spelled correctly, and exists in the Datacenter specified.

Operation	Sub-Operation	vSphere Error Description	Explanation	Recommended Action
create, clone, coldClone		Invalid datastore specified for create/clone	The datastore cannot be found in the vSphere server under the Datacenter specified.	Check that the datastore name specified in the request is spelled correctly, and exists in the Datacenter specified along with it.
create, clone, coldClone		The only allowed valid adapter types are Flexible, E1000 and VMXNET_2.Please Specify one amongst these three	The value for Network Adapter Type is incorrect.	Check that the value for one of the Network Adapter Type is spelled correctly. Valid values are: Flexible, E1000, VMXNET_2.
create		Incomplete request. Both network name and the network adapter type are required in this case.	Network Adapter name is missing.	Check that the Network Adapter name is not blank in the request.
create		Missing mandatory values : null. Cannot complete specified operation	Datastore name is required for create virtual machine operations.	Check that datastore name is not left blank in the request.
create		error : A specified parameter was not correct. configSpec.numCPUs	The value specified for the number of CPUs of the new virtual machine is invalid.	Check that the number of CPUs specified in the request is within reasonable range for the guest operating system.
create		error : A specified parameter was not correct. configSpec.memoryMB	The value specified for the memory size of the new virtual machine is invalid.	Check that the memory size (in MB) specified in the request is within reasonable range for the guest operating system.
create, clone, coldClone		error : File is larger than the maximum size supported by datastore '*<datastore_name>*'	Hard disk capacity specified for the new virtual machine exceeds what is available in the datastore.	Check that the hard disk space specified in the request is within reasonable range.
create, clone, coldClone		Some of the required parameters either DiskSize or DataStore Name is not specified	Either Hard disk capacity or Datastore name is not specified for the new virtual machine.	Check that either the hard disk capacity or the Datastore name is not left blank on the request.
reboot, shutdown, standby		ToolsUnavailable	Windows OS operations cannot be performed if VMware tools have not been installed / brought up in the virtual machine.	Legitimate processing error for end user's follow up actions (fix request or fix VM environment).
snapshot, coldSnapshot, revertSnapshot, removeSnapshot		Missing mandatory values : Snapshot Name. Cannot complete specified operation	Snapshot name is a required attribute for all create/revert/remove snapshot requests.	Check that the snapshot name specified in the request is not left blank.

Operation	Sub-Operation	vSphere Error Description	Explanation	Recommended Action
snapshot, coldSnapshot		Missing mandatory values : Snapshot Description. Cannot complete specified operation	Snapshot description is a required attribute for all create snapshot requests.	Check that the snapshot description specified in the request is not left blank.
revertSnapshot, removeSnapshot		Could not find snapshot	Snapshot name provided cannot found in vSphere for the virtual machine specified.	Check that the snapshot name specified in the request is spelled correctly, and exists for the virtual machine specified along with it.
reconfigure		Invalid reconfigure type specified	Reconfigure Type is a required attribute for all reconfigure actions.	Check that the reconfigure type specified in the request is spelled correctly. Valid options are: memory, cpu, nic, disk, optical.
reconfigure		Missing mandatory values : null. Cannot complete specified operation	ReconfigOperation is a required attribute for all reconfigure actions.	Check that the ReconfigOperation value is not left blank on the request. Valid options are: add, modify, remove.
reconfigure	add disk, add optical, add nic	TooManyDevices	Number of virtual devices exceeds the maximum for a given controller.	Check that the number of devices to be added is within range. (Up to 8 hard disks, 2 optical drives and 4 network adapters can be added through the VM agent, but only one device at a time.)
reconfigure	add disk	FileAlreadyExists	File already exists on datastore for the hard disk to be added.	Legitimate processing error for end user's follow up actions (fix request).
reconfigure	modify disk, modify optical, modify nic, remove disk, remove optical, remove nic	The device being updated or removed does not exist in the virtual machine.	Device label is incorrect.	Check that the value in the Device label is set to an existing device for the virtual machine.
reconfigure	modify disk	error : Invalid operation for device '0'.	Disk size cannot be decreased.	Check that the disk size specified in the request is not smaller than the original size.
reconfigure	cpu	error : A specified parameter was not correct. configSpec.numCPUs	CPU value is not correct.	Check that the number of CPUs in the request is a reasonable number for the guest operating system.
reconfigure	memory	error : A specified parameter was not correct. configSpec.memoryMB	Memory size is not correct.	Check that the memory size in the request is a reasonable number for the guest operating system.

CHAPTER 4

Portal Manager

- Overview, page 4-1
- Portal Designer, page 4-3
- Content Portlets, page 4-6
- Reserved Portlets, page 4-12
- Custom Content, page 4-15
- HTML and JavaScript Portlets, page 4-17
- JSR Portlets, page 4-21
- Portal Pages, page 4-25
- Portal Settings, page 4-31
- Reference Data, page 4-36
- An End User's View of the Portal, page 4-46
- Importing and Exporting Portal Content, page 4-52
- Portal Access Control, page 4-56

Overview

The Portal Manager solution provides users with a familiar content management experience that can be tailored to their needs.

Portal Manager addresses many user interface customization requirements by providing administrators and designers with finer control over the appearance of their Service Catalog implementation. At the same time, the portal platform allows multiple external data sources to coexist within Service Catalog screens, providing a holistic view into Data Center or general IT services and resources.

The portal front-end provides a way to interact with services, service items, standards, offerings, and other core entities in the application, by integrating portlets exposing this content into the portal. Capabilities to customize the portal's appearance or manage portlet content can be assigned to selected users, roles or groups through the use of Role-Based Access Control (RBAC). These capabilities include:

- Drag and drop user interface, fashioned after MyYahoo and iGoogle portals
- User-selected skins
- User-selected content
- Ability for users to create their own portal pages

Portal Designer provides an interface to build a variety of portlets using application data, JavaScript/HTML, ad-hoc lists, or third-party JSR-compliant portlets. Portal Designer allows interface designers to:

- Create portlets from external or third-party sources, such as monitoring data or virtual machine data from VMware
- Create portlets to highlight common services or
- Create portlets to show users what they already own, with links to services related to those items
- Show announcements, video, or other types of media
- Leverage RBAC to create a flexible user interface that is at once simple for casual users, and advanced for power users

Portal Manager Roles and Capabilities

Access to the capabilities provided by Portal Designer is controlled via standard Role-Based Access Control (RBAC). Design personnel can be granted access to all or selected portions of the Portal Designer functionality. Access to the functionality of the portal front-end, including the ability to customize the portal by adding portlets or changing the look-and-feel of the portal pages, may also be controlled via RBAC. Details on the portal-related capabilities and how to assign these to project personnel are given in the *Cisco Service Portal Configuration Guide*, and in the *Organization Designer Online Help* regarding roles.

Similarly, end users' ability to access the portal front-end can be controlled via RBAC. Only users with a role that includes the "Access Service Portal" capability are able to see the "Service Portal" module menu and navigate to the portal pages and portlets.

Prerequisites

Portal Designer is a separately licensed module of Service Portal. You must be licensed to run Portal Designer in order to use the content management functionality.

Portal Manager modules are supported on Internet Explorer 7 and 8. The portal front-end is also supported on Firefox 3.6.

Portal Designer

Portal Designer is the Service Portal module that allows designers and administrators to design and manage pages and portal content; and to specify which users or groups or users are able to access particular content.

Portlet Taxonomy

A portal or "web portal" provides a user-configurable user interface for accessing application functionality.

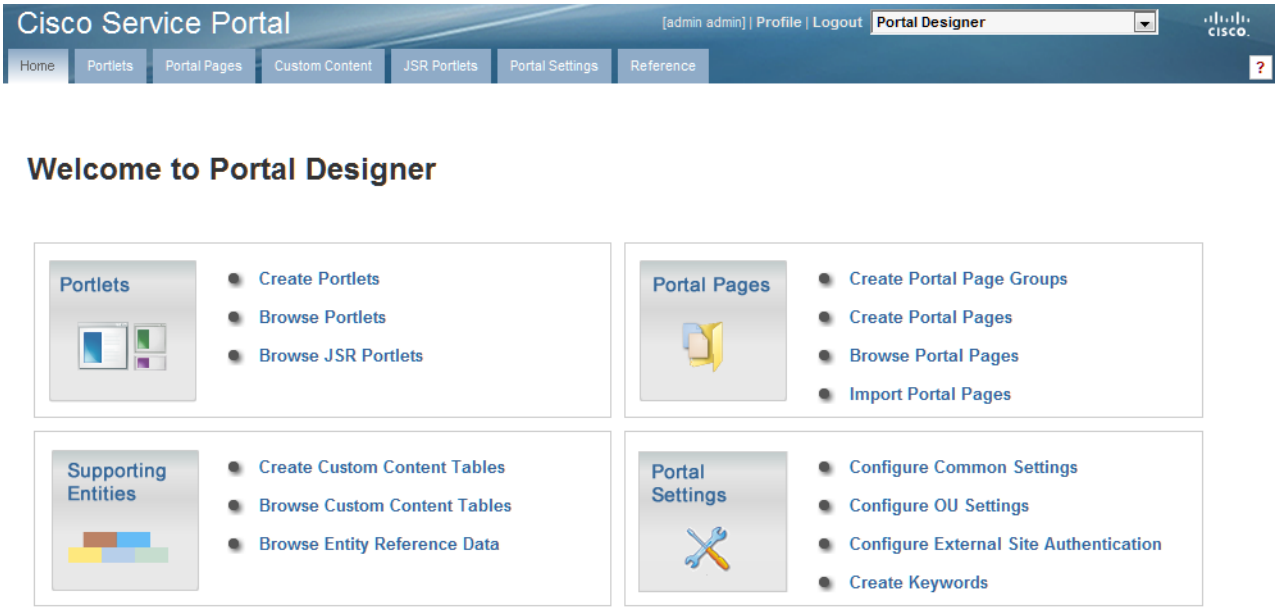
A portal page can include one or more "portlets", software modules that can be plugged into the portal page and arranged as nonoverlapping portions of the page. There are two types of portlets available for the users in the application:

- Reserved portlets – These are preconfigured portlets that are installed with every application instance and are available in the My Workspace module only. The Reserved portlets are Search, Order Status, and Approvals. See the "Reserved Portlets" section on page 4-12.
- User-defined portlets: These include JSR portlets or portlets developed using the Portal Designer. Portal Designer allows the designer to define the content and presentation of the portlets with predefined filters and lookup, HTML, and JavaScripts. A JSR portlet may be developed in any Java development environment that is compliant with JSR 168 or 286, and optionally using the Service Portal Java Client to leverage the application public APIs. A third-party JSR portlet can also be easily integrated into the Portal Manager solution.

Note The system-defined portlets available in the My Services module are not true "portlets" in that they not available in Portal Designer and cannot be added to a portal page. Some examples of these are "My Authorizations" and "My Requisitions".

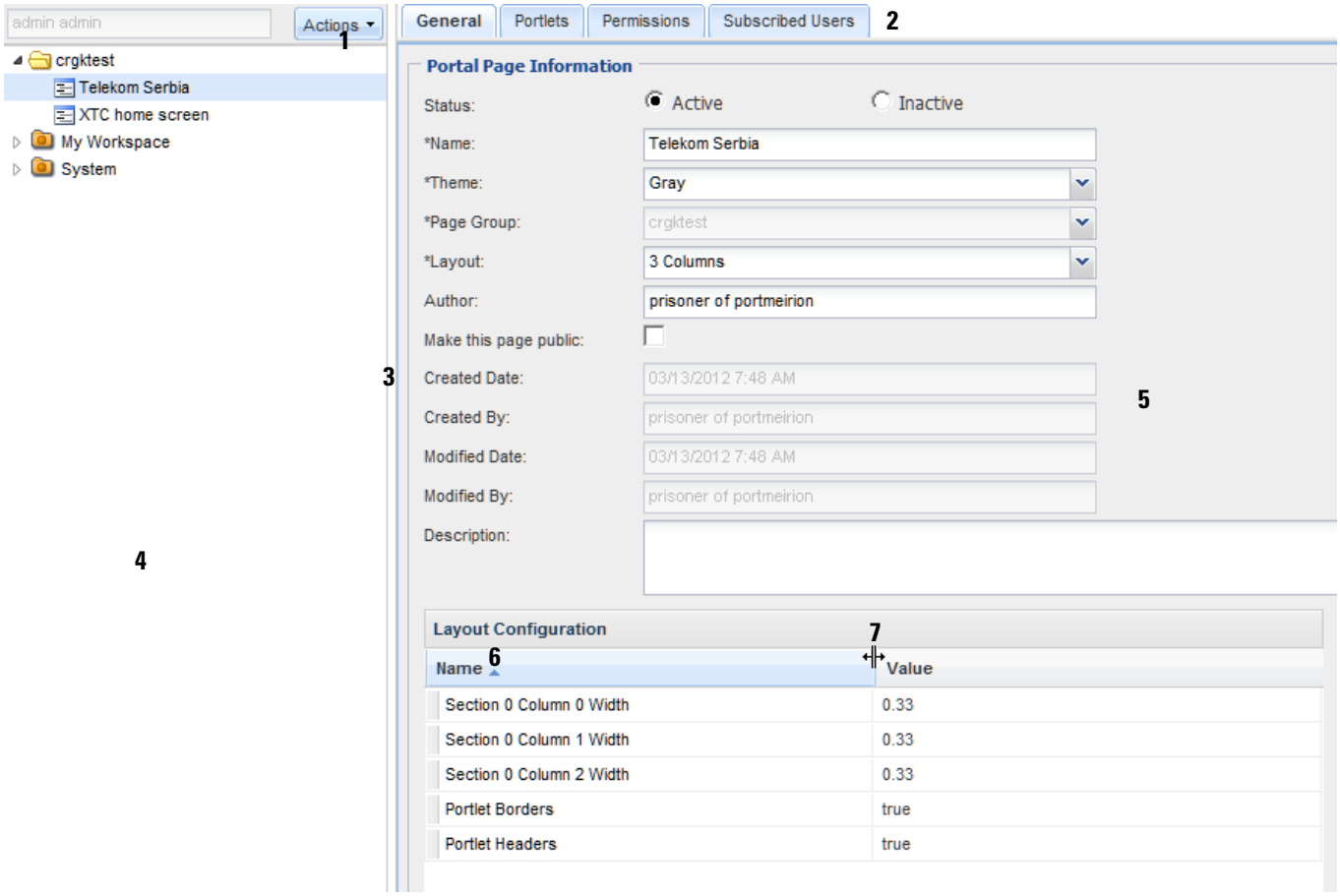
Managing the Portal Designer Screens

When you choose the Portal Designer module, the Portal Designer home page appears.



You can access the Portal Designer screens by clicking on the tab corresponding to the option you want, or by clicking on a bulleted function in the Portlets, Portal Pages, Supporting Entities, or Portal Settings sections.

Each option presents a page similar in appearance to the example below, consisting of a list pane on the left and a content pane on the right displaying the objects of interest, such as portlets or portal pages.



1	Drop-down Action menu	**5**	Content Pane
2	Subtabs	**6**	Sort button
3	Divider	**7**	Width cursor
4	List pane		

You can also control the appearance of the screen:

- The width of the list pane can be adjusted by moving the divider.
- The width of the individual columns in the Layout Configuration (or other grids that appear on different subtabs of Portal Designer) can be adjusted by moving the width cursor.
- The order of the individual columns in the Layout Configuration (or other grids that appear on different subtabs of Portal Designer) can be changed by clicking on an column title, holding the mouse button down, dragging the mouse across the title line until the insert cursors appear (arrows pointing to the current position) and then releasing the mouse.
- On selected pages, the Sort button toggles the order in which rows are displayed.

Content Portlets

Content Portlets leverage the Service Portal REST API which support the RBAC-enabled access to the application data. The API framework, along with functionality for defining the appearance and behavior of portlets, allow portal designers to easily include predefined content in a portlet and to configure that portlet for inclusion in a portal page. The portlet content may consist of many types of data available within Service Portal, including definitional data (agents and service definitions); directory data (people and organizations); transactional data such as requisitions; and Lifecycle Center objects, including service items and standards. In addition, designers can define their own Custom Content, maintainable through Portal Designer, for inclusion in portlets.

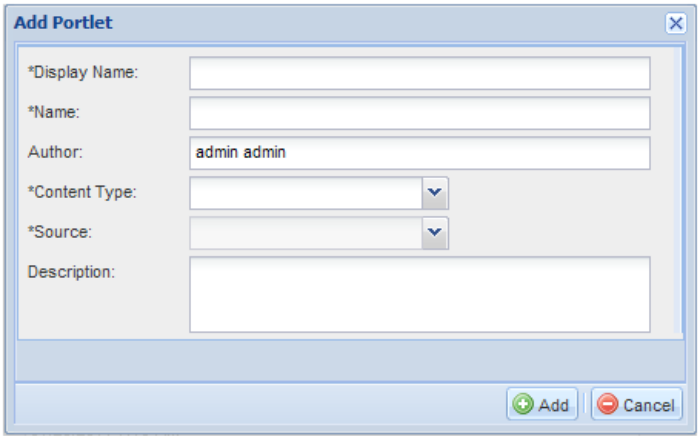
Defining a Content Portlet

A content portlet definition has four components, corresponding to the four subtabs displayed under the Portlets tab:

- General: A name for the portlet, as well as its contents and other general information.
- View: The display characteristics of the portlet. All portlets based on predefined content are displayed as grids. Designers can specify the grid size as well as details of its layout, both in normal and maximized view.
- Filter: Optional filters to restrict which rows from the content source are displayed in the portlet.
- Permissions: The Service Portal entities (people, organizations, roles, or groups) which have access to the portlet, and the type of access (read or read/write).

Portlet Content

Choose **Actions > New Portlet** to define a portlet. The Add Portlet window appears, as shown below:



Field	Description
Display Name	The name to be displayed as the default portlet title; free-format.
Name	The internal name for the portal; can contain only letters, numbers, and the underscore character (no spaces).

Field	Description
Author	A text string which defaults to the first and last name of the current user; may be edited.
Content Type	The type of content of the portal. Options are: **Core Entities** Entities used by Request Center and Demand Center; all such objects are listed under the Reference tab of Portal Designer. **Service Items** Any system- or user-defined service item, defined via Service Item Manager. **Standards** Any system- or user-defined standard, defined via Service Item Manager. **HTML/JavaScript** The portal designer will design the portlet according to rules specified in the "HTML and JavaScript Portlets" section on page 4-17. **Custom Content** User-defined tables, defined and maintained using the Custom Content tab of Portal Designer.
Source	Once the Content Type is chosen, a drop-down list of data sources available for that type appears. One is chosen as the basis of portlet content.
Description	Optional documentation on the portlet.

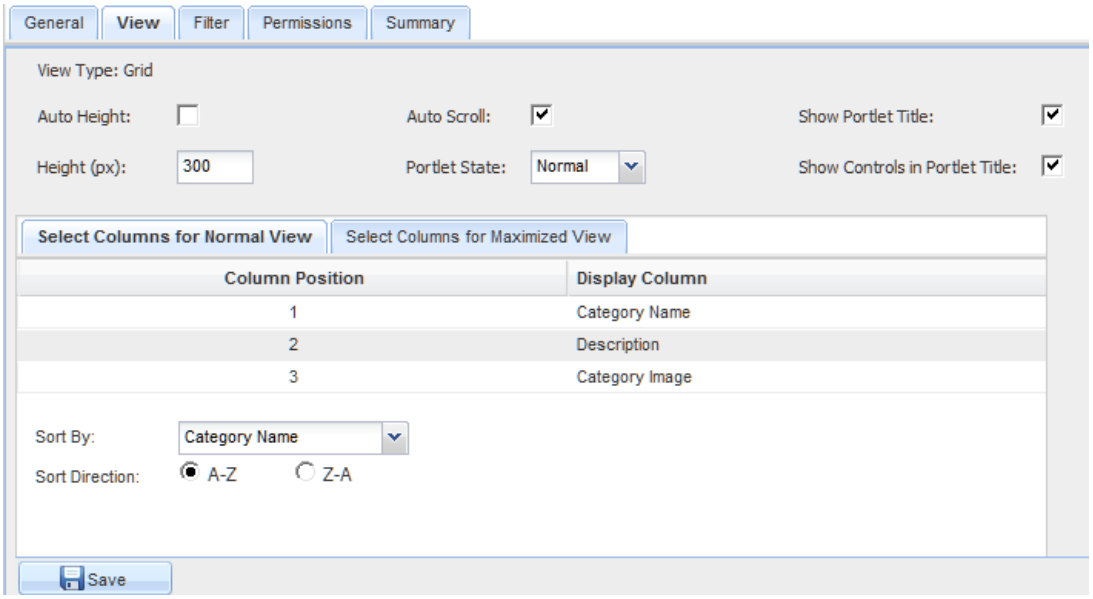
Once the portlet has been saved (added), the rest of its definition can be provided using the tabs available in the content pane. The portlet is assigned to a folder corresponding to its content data source, and selectable from the list pane on the left of the Portal Designer window.

For General information:

- All portlets are created in an "Active" status. Only "Active" portlets can be included on a portal page.
- To disable portlets from use in portal pages, set the status of the portlets to "Inactive" and remove them from the pages in which they are included. Inactive portlets that are still present in portal pages are hidden from the users.
- Keywords can optionally be associated with a portlet to allow users to search for portlets when adding content to portal pages. Such keywords are defined using the Portal Settings tab.

Portlet View

Content portlets are implemented with a "Grid" view. For such portlets, which reference a Service Portal entity, the View tab allows designers to specify which columns from the chosen data source to display (via the "Select Columns …" grids), and (optionally) which rows to display (via the Filter subtab). Sorting, in ascending or descending order, can be applied to any of the columns chosen in a view. The specified data is displayed in a user-configurable grid which may be sized to fit on a portal page.

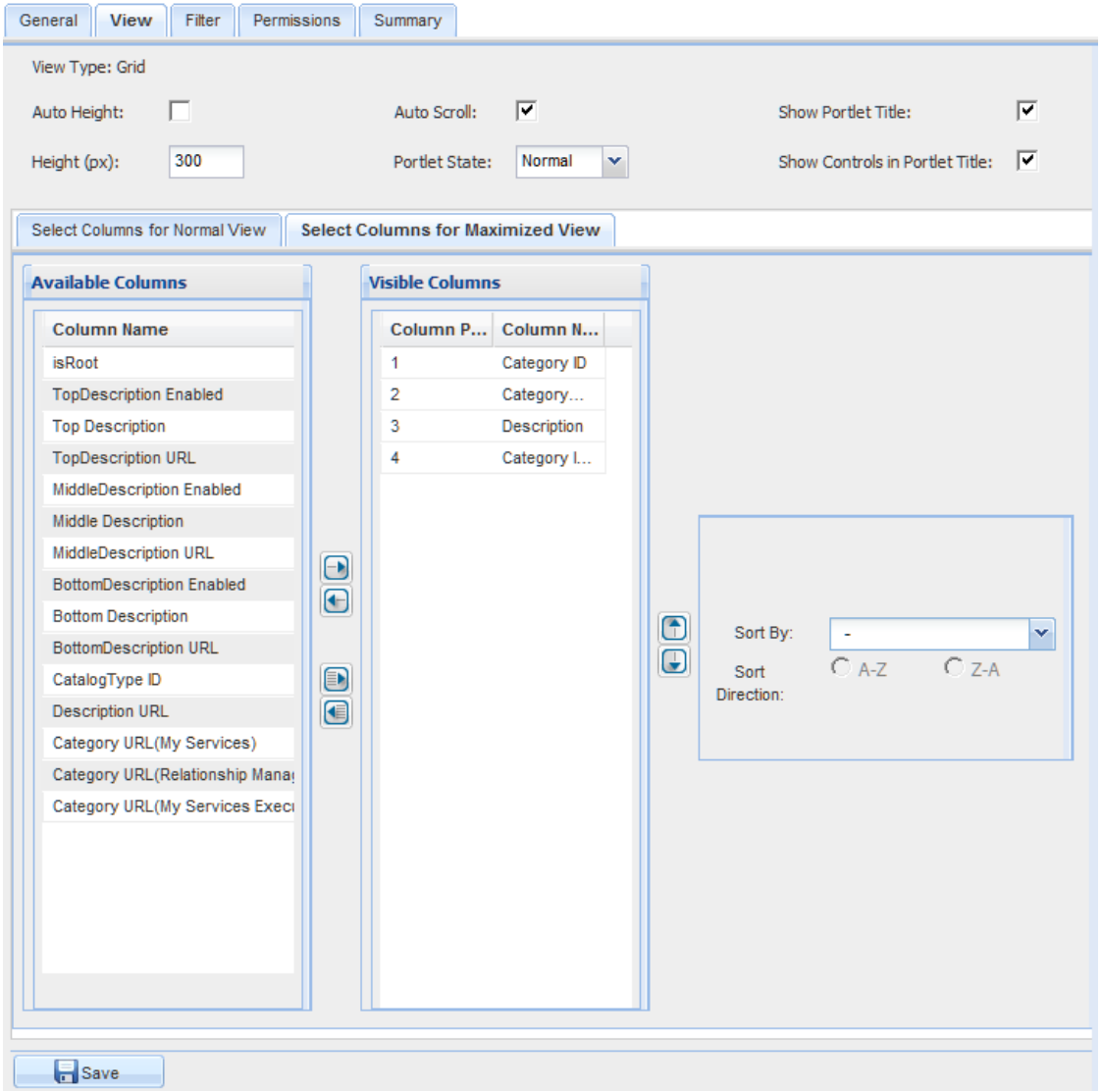


The view properties specify the default appearance of the portlet when it is included in a portal page. These can be overridden by individual users and on individual pages.

Field	Description
Height (px)	The initial height (in pixels) of the portlet; not applicable when the Auto Height setting is enabled.
Auto Height	Check box that indicates whether the height of portlet should be sized to the content automatically; not applicable to HTML portlets.
Auto Scroll	Check box that indicates whether a scroll bar should be displayed in the portlet; not applicable to HTML portlets.
Portlet State	Whether the portlet should initially be displayed in its Normal or Minimized view.
Show Portlet Title	Check box that indicates whether the portlet title (Display Name) should be displayed at the top of the portlet.
Show Controls in Portlet Title	Check box that indicates whether the controls on the title bar such as Minimize and Maximize buttons should be displayed.

Two views of the portlet data are available:

- The Normal View provides a summary/overview of portlet content. Up to three columns can be included and data sorted by the contents of one of those columns.
- The Maximized View provides more detailed portlet content and can include any number of columns from the portlet's data source.



To move one or more columns between the Available Columns and Visible Columns panes, click the columns (**Ctrl-Click** to choose multiple columns) and click the upper left- and right-Arrows buttons ( ). Click the lower left- and right-Arrow buttons to move *all* columns ( ). Click the up- and down-Arrows ( ) to rearrange the order in which the Visible Columns appear.

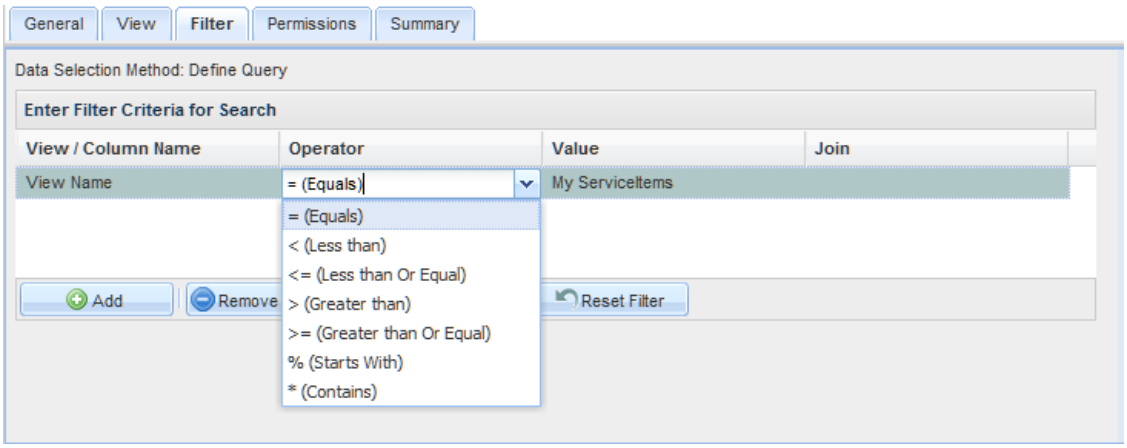
The columns that comprise Service Portal core entities and those supported for sorting are described in the "Reference Data" section on page 4-36. Other data sources for content portlets (Custom Content, Service Items, Standards) are site-specific—see your service catalog designers for detailed information.

For HTML or JavaScript portlets, the View subtab presents an editor for the designer to enter the source code that defines the appearance and content of the portlets. These two portlet types are explained in more detail in the "HTML and JavaScript Portlets" section on page 4-17.

Portlet Filter

The Filter subtab is available only for portlets that are defined by directly referencing a Service Portal entity. It allows designers to filter the data retrieved from the designated portlet source.

Query criteria are formulated by specifying one or more filters. Each filter consists of a relational statement comparing the value of a column in the entity on which the portlet is based to a specified. Multiple filters can be combined by ANDing or ORing individual filter specifications; rules of precedence apply.



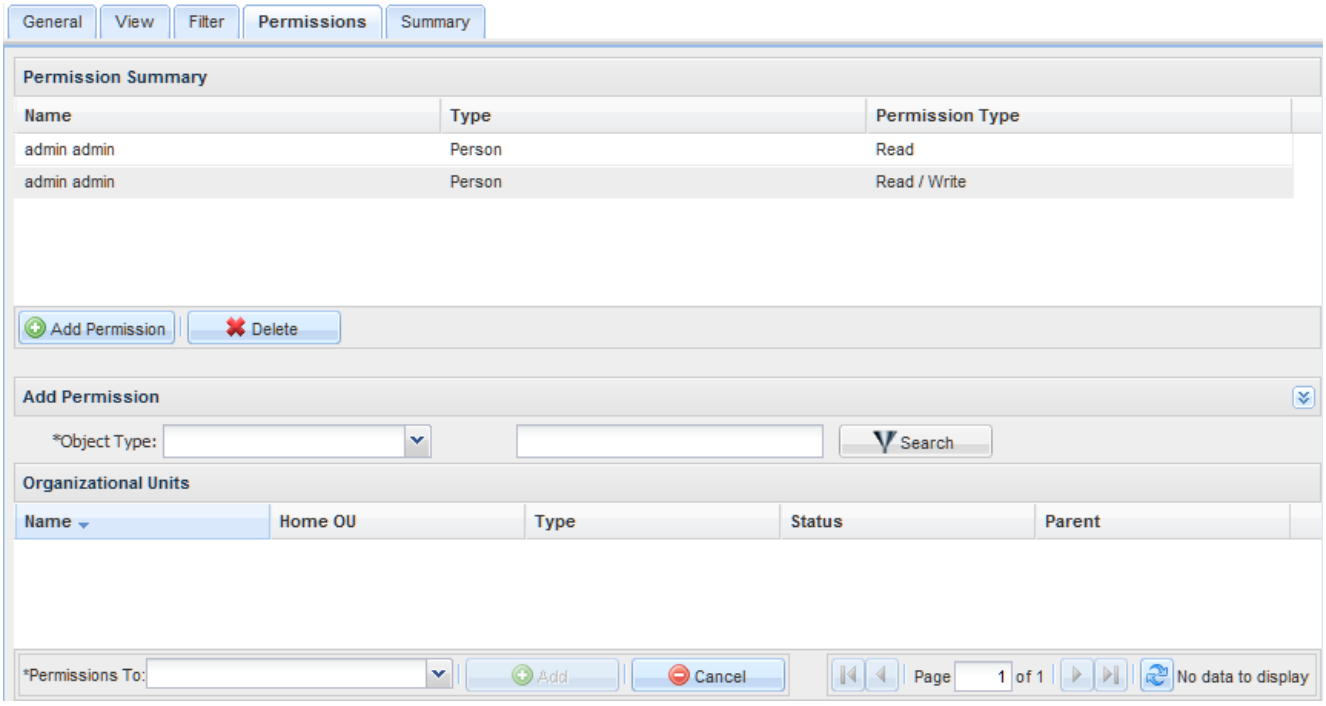
The filtering criteria available for selection are dependent on the entity referenced in the portlet. The drop-down lists for column and operation are context-sensitive and show only the supported combinations in the underlying REST API of the entity.

Filtering characteristics for different entity types include:

- Categories, services, service offerings, and agreements: Permissions to Order the Service/Initiate an Offering are applied on top of the filtering criteria. Portal end users are able to view only data to which they normally have access through the My Services and My Services Executive modules.
- Directory entities and agents: RBAC read permissions are enforced along with the filtering criteria. Portal end users are able to view only data which they would see in Organization Designer and Service Link.
- Service items, standards and custom content: All attributes are available as filters. In addition, subscription-based filters are available to display service items based on the user's access permissions:
  - My Service Items: Instances of the chosen service item owned by the portal user, or instances owned by the portal user's business units; the latter is applicable to users who have the My Services capability "View Service Items for My Business Units".
  - All Service Items: All instances of the chosen service item if the user has the Service Item Manager "Manage Service Items" capability.
- Requisitions, authorizations and tasks: Context-specific filters are available to restrict what the portal user can view in the portlets. The filters mimic the predefined views that are available in My Services and Service Manager (for example, Ordered for Myself, My Assigned and Unassigned Authorizations, Available Work).

Portlet Permissions

The Permissions subtab allows designers to designate which users should be able to access the portlet and the type of access to grant. Any users so designated should also be granted the Portal module capability to “Access Service Portal”.



Follow the procedure below to add permission to allow individual users or groups of users to place the current portlet on their portal pages:

Step 1 Click **Add Permission**. The Add Permission panel in the lower part of the page appears.

Step 2 For **Object Type**, choose Organizational Units, Group, Person, or Role. This determines the granularity with which you are assigning the permission to access the portlet.

Step 3 Fill in all or part of the name of the object in the search box and then click **Search**. All objects meeting the search criteria appear. To display all objects of the specified type, leave the search box blank. To display objects whose name matches a particular pattern, you may include the wildcard character (*) in the search string.

Step 4 Click the rows to which the permission is to be granted—use **Shift-Click** and **Ctrl-Click** to choose multiple rows.

Step 5 Click the **Permission To** drop-down menu to grant Read or Read / Write permission to the portlet.

Step 6 Click **Add** to add the specified users with the specified permission.

Portlet permissions also control which portlets users can access in Portal Designer if they have the “Manage Portlets” capability. The user who creates the portlet is automatically granted all access permissions to it.

Unlike the Service Designer module, RBAC filtering is applied to the objects available for assigning permissions. In other words, the portal designer needs to have read or read/write permissions to organizational units, people, groups and roles in order to search for them in the Add Permission pane and to view them in the permission summary grid once they have been added.

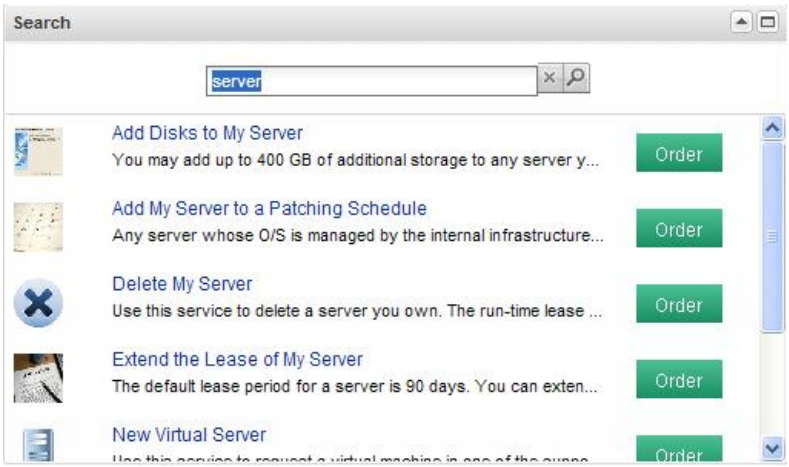
To enable all users to view the portlet, portal designers can assign Read permission to an "umbrella" organizational unit which is the parent of all business units. Alternatively, they can work with the organization designer to grant the portlet permission to the "Anyone" role in the Organization Designer module.

Reserved Portlets

Service Portal includes three Reserved Portlets—the Search Portlet, Order Status Portlet, and Approvals Portlet. In View Mode and Edit Mode of a portal page (see the "View Mode of a Portal Page" section on page 4-47 and the "Edit Mode of a Portal Page" section on page 4-49), clicking the Search, Orders, or Approvals button is a shortcut to add the Search, Order Status, and Approvals portlet, respectively, to the page, rather than using the Add Content button (see the "Add Content" section on page 4-49).

Search Portlet

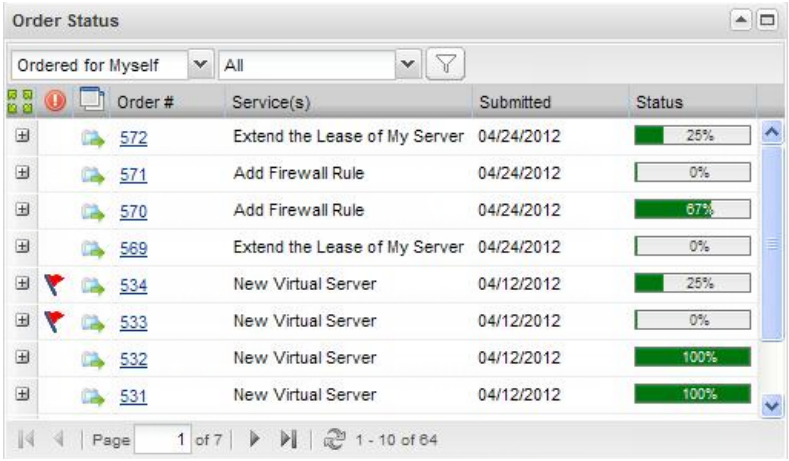
The Search portlet functions the same as the "Search for Services" function ("Search for services containing:" field) in My Services. See the *My Services Online Help* for more information.



To perform a Search, enter search criteria in the text box and click on the Search icon . Clicking on the icon clears the search box. Wildcards (*) are supported and perform as a case-insensitive search. A list of services matching the search criteria appears below where you can click on a service name to pop up the Service Overview/Summary page, or click **Order** to pop up the Service Order page.

Order Status Portlet

The Order Status portlet, used to track and view orders, is similar to the Requisitions tab in My Services. See the *My Services Online Help* for more information.



The Order Status portlet displays a list of requisitions filtered by requisition type and requisition status.

To change the filtering:

Step 1 From the left drop-down menu, choose the type of requisition to view—**Ordered for Myself**, **Ordered for Others**, or **Ordered for my unit**.

Step 2 In the right drop-down menu, choose a requisition status—**Preparation**, **Ordered**, **Ongoing**, **Cancelled**, **Closed**, **Rejected**, or **All**.

Step 3 Click the button to filter the requisition list based on your selections in the Steps above.

Note Filter selections are remembered for the Order Status portlet for each user on every page the portlet is added, even when filter selections are changed in the Requisitions tab of My Services.

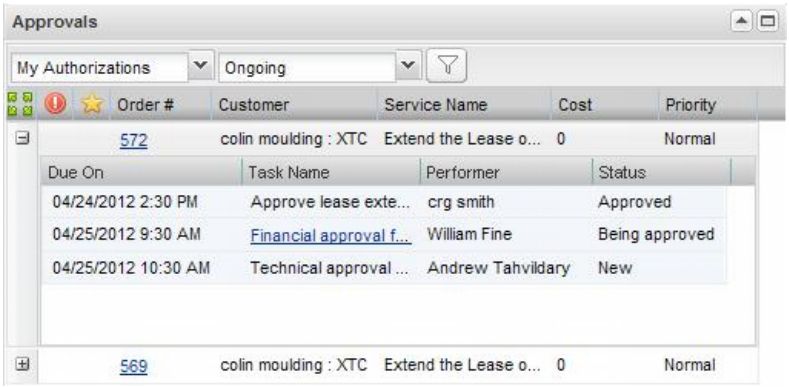
Note The "Ordered" requisition status only appears in the requisition list if the "Submit, Approve and Review Asynchronously" setting is turned on in the Common section of Administration > Settings > Customizations. See the Site Administration chapter of the *Cisco Service Portal Configuration Guide* for more information.

The Order Status portlet has the following features:

- Click the view button to view the delivery process in a popup window.
- Click the maximize button to view the Order Status portlet in the full page, showing all columns. Click the button to resize the portlet to its original size.
- Click on an order number from the Order # column to pop up the Requisition Summary page for that order.
- Click the icon to display service and delivery information for that order below the order row. Click the view button to view the delivery plan in a popup window. Click the service name to view service details in a popup window. Click the icon to close the display.
- The Status column displays a progress bar showing the completion percentage of the order based on the durations of all tasks. The progress bar only updates when a task is completely done.

Approvals Portlet

The Approvals portlet, used to track and view authorizations, is similar to the Authorizations tab in My Services. See the *My Services Online Help* for more information.



The Approvals portlet displays a list of authorizations filtered by authorization type and authorization status.

To change the filtering:

Step 1 From the left drop-down menu, choose the type of authorizations to view—**My Authorizations**, **My Assigned and Unassigned**, or **Authorizations for Others**.

Step 2 In the right drop-down menu, choose an authorization status—**Ongoing**, **Approved**, **Rejected**, **Reviewed**, **Cancelled**, or **All**.

Step 3 Click the button to filter the authorization list based on your selections in the Steps above.

Note Filter selections are remembered for the Approvals portlet for each user on every page the portlet is added, even when filter selections are changed in the Authorizations tab of My Services.

Note The "Ongoing" status appears as "Being Approved" or "Under Review" in the authorizations list.

The Approval portlet has the following features:

- Click the maximize button to view the Approval portlet in the full page, showing all columns. Click the button to resize the portlet to its original size.
- Click on a requisition number from the Req ID column to pop up the Task Details for that requisition.
- Click the icon to display all the tasks for that requisition below the requisition row. Click the icon to close the display.

Custom Content

Custom content comprises user-defined tables that serve as a source of content for the portal. Such tables can be referenced as the data source for portlets just like Lifecycle Center Standards. Such tables are defined and maintained in the Custom Content tab in Portal Designer and are organized into content groups.

Content groups allow custom content tables to be grouped in a logical manner for easier navigation and control of access permissions in Portal Designer. Read/write permissions can be granted at the group level for managing all content tables within the group or at a more granular level for individual tables.

Both custom content groups and content definitions can be created by clicking on the Actions drop-down arrow in Custom Content and choosing the appropriate option.

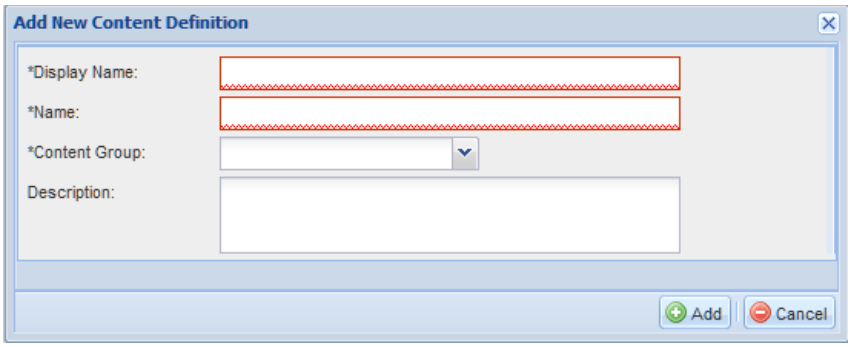
The "System" content group is available by default. It contains two commonly used custom content definitions—Announcements and Links—to provide convenience to portal designers.

Creating Custom Content Table

A custom content table has three components, corresponding to the three tabs displayed in the user interface:

- Content Definition: The columns of the table
- Content Data: The rows of data in the table
- Permissions: Service Portal entities (people, organizations, roles, or groups) which have access to the portlet, and the type of access (read or read/write)

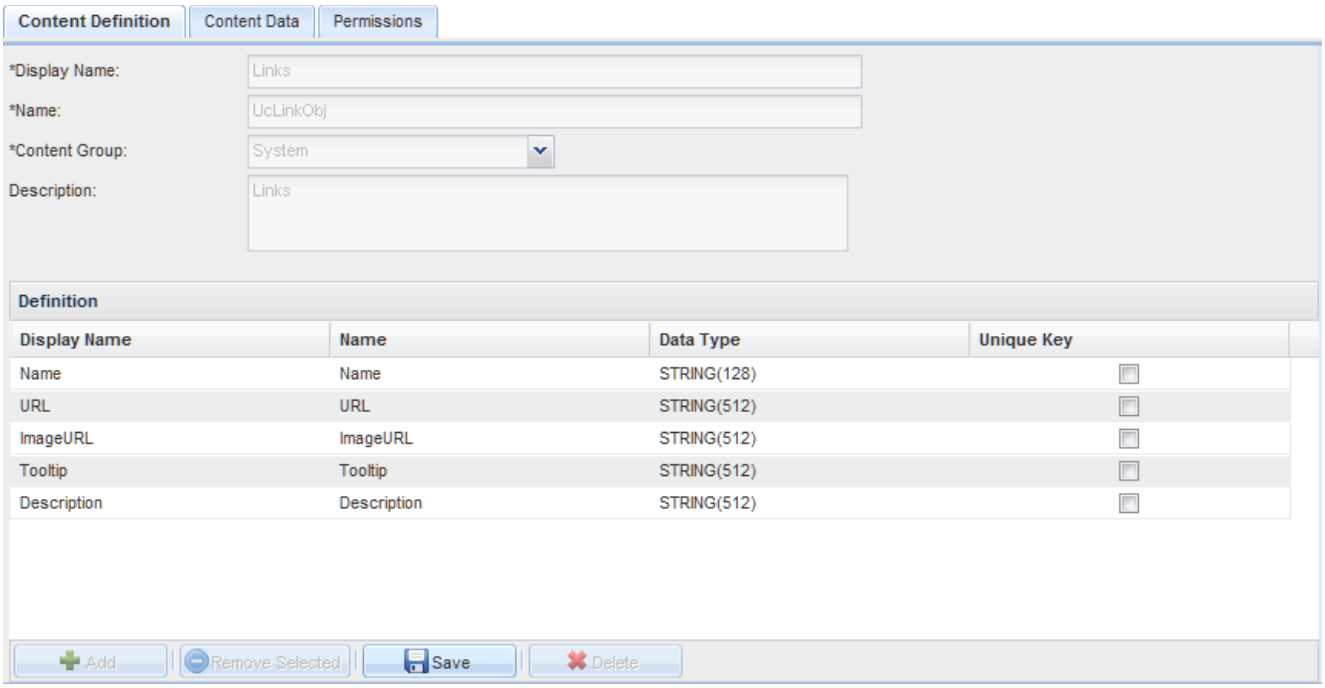
To create a custom content table, choose **Actions > New Content Definition** to define a new custom content table. The Add New Content Definition window appears, as shown below:



Field	Description
Display Name	The name to be displayed as the name of the table; free-format
Name	The internal name for the table; can contain only letters, numbers, and the underscore character (no spaces)
Content Group	The group in which the custom content table is located
Description	Optional documentation on the custom content table

Once the table has been saved (added), the rest of its definition can be provided using the tabs available in the content pane.

Content Definition



Content Definition comprises of the name, description and table columns that are characterized by the following four attributes:

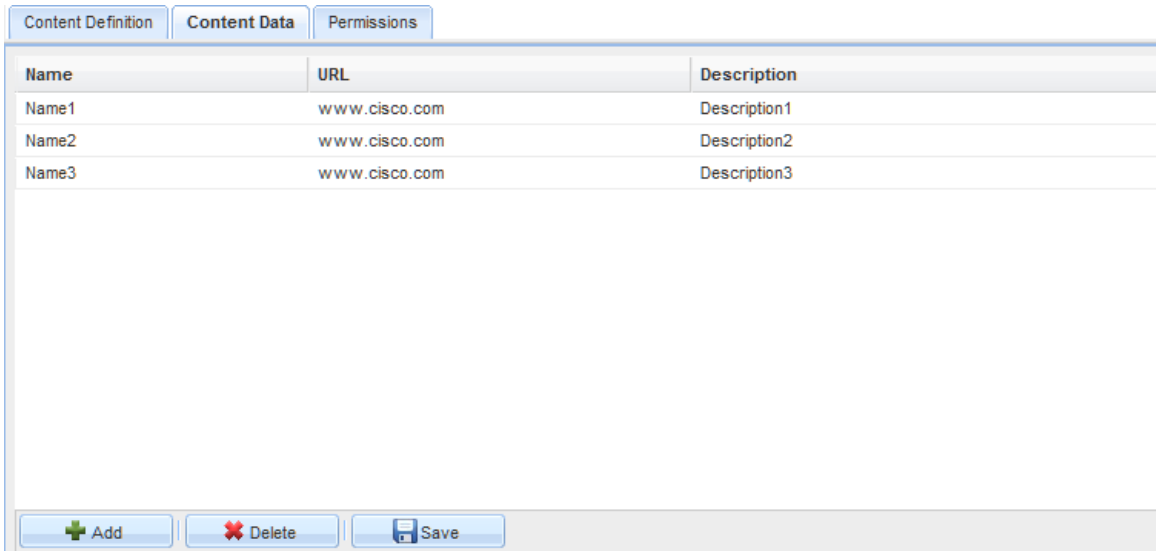
- Display Name – Label for the table column as displayed in the portlets
- Name – Internal name for the table column; should be unique within the same table
- Data Type – The type and maximum value/length allowed for data stored in the table column:

Data Type	Description
INTEGER	A positive or negative whole number or zero; in the range: –32,676 to 32,676
LONGINTEGER	A positive or negative whole number or zero
DATETIME	A date and time
MONEY	A positive or negative number with up to three digits of decimal precision or zero
DOUBLEFLOAT	A positive or negative number with decimal precision or zero
STRING(32)	A string (alphanumeric) value up to 32 characters long
STRING(128)	A string (alphanumeric) value up to 128 characters long
STRING(512)	A string (alphanumeric) value up to 512 characters long

- Unique Key – Whether the table column is used alone or along with other columns to uniquely identify a row of data in the table; used for validating rows entered into the table

Content Data

Content Data are the rows in the table, enterable through a grid-like user interface.



The values entered should conform to the data type and unique key restrictions specified in the Content Definition.

Content Permissions

Content access permissions can be controlled in a way similar to the portlets on the Permissions tab (see the "Portlet Permissions" section on page 4-11).

For users to view the content in Portal Manager, read permissions to both the content definition and data are required.

HTML and JavaScript Portlets

HTML and JavaScript portlets provide the ability to define portlets that are free-format and use those portlets within portal pages. Such portlets can be defined and maintained completely within Portal Designer. They only need to obey the coding rules described in this section.

For HTML and JavaScript portlets, all data displayed in the portlet is provided via the HTML or JavaScript code. Therefore, the View and Filter subtabs are disabled.

HTML Portlets

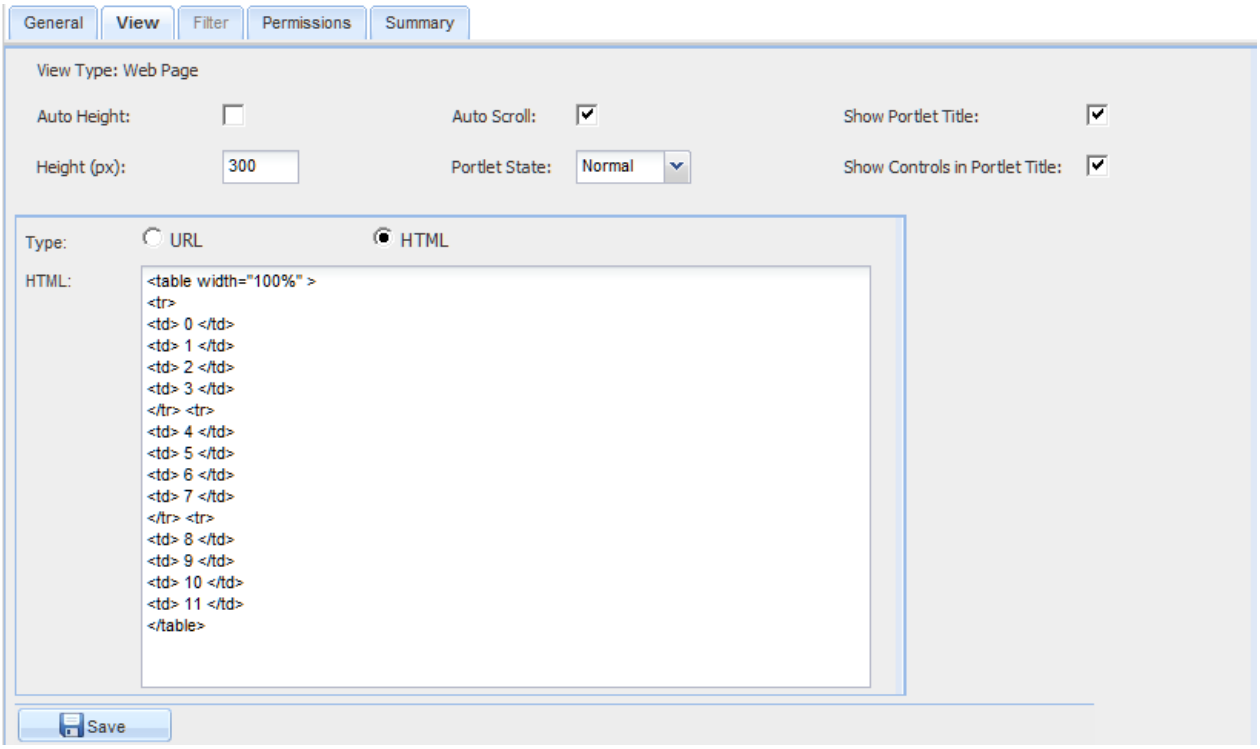
An HTML portlet consists of an HTML snippet or a URL.

When an HTML portlet is defined, the View subtab adjusts its contents so that:

- The View Type is by default set to Web Page.
- The designer can designate the portlet subtype (HTML or URL).

- An edit window for data entry of the appropriate type of text appears.

A text area is provided for entering HTML, as shown in the sample below.



The URL provides a hyperlink to the specified web page. It can be an absolute reference to an external web site or a relative reference to a Request Center page. For example:

/RequestCenter/myservices/navigate.do?query=orderform&sid=14).

The snippet cannot include <head> or <body> tags, since the portal is rendered as part of the page body. You may wish to use tags, so that styles can be applied to portions of the portlet, but this is not required. The snippet can include <script> definitions or invocations of JavaScript functions defined in local script, defined within the HTML snippet.

Authentication settings can be optionally associated with a URL-based portlet to allow automatic login to the external site. The common settings for the external site authentication are defined in the Portal Settings tab in Portal Designer. Credentials for individual users are maintained through the Edit Password tab in the portal front-end. See the "Authentication Settings" section on page 4-32 for more details regarding the different options for configuring external site authentication.

JavaScript Portlets

Overview

When a JavaScript portlet is defined, the View subtab is set to JavaScript and a text area is provided for entering the code.

JavaScript portlets can display dynamic content and use the full range of JavaScript functionality. The user interface of JavaScript portlets should be written using ExtJS functions, since ExtJS is the UI framework used for the portal front-end. The complete reference can be found at the Sencha website.

Like the content portlets, JavaScript portlets may consist of data available within Service Portal which is accessible through the use of the REST API. See the *Cisco Service Portal Integration Guide* for details regarding the APIs available.

The key concepts for using REST API and EXTJS to create a grid portlet are illustrated in the following sections with sample code snippets.

Invoking REST APIs

Follow the steps below to make use of the REST API to retrieve the data you want to show in the portlet:

Step 1 Identify the appropriate REST API to use based on the content type and filtering method required. For example, if the portlet is used to display all the orderable services for a particular category, the REST API to be used is:

/RequestCenter/nsapi/definition/servicedefs?categoryName=<categoryName>

Step 2 Define an array that contains all the attributes of the content type as defined in the REST API.

```
fieldList = [ "serviceId",
              "serviceName",
              "description",
              "topDescription",
              "middleDescription",
              "bottomDescription",
              "pricingScheme",
              "revisionNumber",
              "status",
              "statusId",
              "expectedDuration",
              "expectedDurationUnits",
              "price",
              "priceDisplaySchemaId",
              "priceDescription",
              "canStartLater",
              "isBundle",
              "dateQualityId",
              "serviceLevelDescription",
              "isOrderable",
              "isReportable",
              "serviceURL"];
```

Step 3 Create the proxy for REST Http GET calls.

```
var proxy = new Ext.data.HttpProxy({
    url: '/RequestCenter/nsapi/definition/servicedefs',
    method: 'GET'
});
```

Step 4 Create an XML data store for the result set, including an XML reader that defines the parameters.

```
var store = new Ext.data.XmlStore({
    autoDestroy: true,
    storeId: 'myStore',
    proxy: proxy,
    root : "services",
    record: 'service',
    idPath: 'rowId',
```

```
    totalProperty: '@totalCount',
    autoLoad: true,
    paramNames: {
      start: 1,
      limit: 10,
      catName : 'Sample Category'
    },
    fields: fieldList
```

Rendering Data in EXTJS Grids

Now that you have the data store ready, you can follow the steps below to create a simple grid for displaying the data in the portlet:

Step 1 Define an array for the columns and appearance of the grid to be used to display the content.

```
displayList = [
                {id: 'id', header: 'Id', width: 50, sortable: true, dataIndex:
'serviceId'},
                {header:'Service Id',dataIndex: 'serviceId',hidden:true},
                {header:'Service Name',dataIndex: 'serviceName'},
                {header:'Description',dataIndex: 'description'},
                {header:'Top Description',dataIndex: 'topDescription',hidden:true},
                {header:'Middle Description',dataIndex: 'middleDescription',hidden:true},
                {header:'Bottom Description',dataIndex: 'bottomDescription',hidden:true},
                {header:'Pricing Scheme',dataIndex: 'pricingScheme',hidden:true},
                {header:'Revision Number',dataIndex: 'revisionNumber',hidden:true},
                {header:'Status',dataIndex: 'status'},
                {header:'Status Id',dataIndex: 'statusId',hidden:true},
                {header:'Expected Duration',dataIndex: 'expectedDuration',hidden:true},
                {header:'Expected Duration Units',dataIndex:
'expectedDurationUnits',hidden:true},
                {header:'Price',dataIndex: 'price',hidden:true},
                {header:'Price Display Schema Id',dataIndex:
'priceDisplaySchemaId',hidden:true},
                {header:'Price Description',dataIndex: 'priceDescription',hidden:true},
                {header:'Can Start Later',dataIndex: 'canStartLater',hidden:true},
                {header:'Is Bundle',dataIndex: 'isBundle',hidden:true},
                {header:'Date Quality Id',dataIndex: 'dateQualityId',hidden:true},
                {header:'Service Level Description',dataIndex:
'serviceLevelDescription',hidden:true},
                {header:'Is Orderable',dataIndex: 'isOrderable',hidden:true},
                {header:'Is Reportable',dataIndex: 'isReportable',hidden:true},
                {header:'Service URL',dataIndex: 'serviceURL'}
];
```

Step 2 Create an EXTJS grid, using the column array and data store defined in the earlier steps.

```
var grid = new Ext.grid.GridPanel({
         store : store,
         columns : displayList,
         renderTo : '#divName#',
         width : "100%",
         autoHeight : true,
         layout : 'fit',
         viewConfig : {
            forceFit : true
         },
         tbar : [combo,filterButton],
```

```
        bbar : [new Ext.PagingToolbar({
                 store : store,
                 displayInfo : true,
                 pageSize : 5,
                 params:{
                          startRow: 1,
                          recordSize: 5
                        }
                 emptyMsg : "No record found"
               })]
});
```

Using AJAX Calls

In addition to REST APIs, AJAX calls can be invoked to retrieve data from other sources to provide the content for the portlets.

JSR Portlets

Overview

Portlets developed using APIs which meet the Java Portlet Specification (JSR168, JSR286) standards may be integrated into Portal Manager solution. These will appear in Service Portal as “Third-Party Portlets”. Vendor-specific implementations may include extensions to the approved APIs; these may not be supported.

JSR portlets can also be developed using Service Portal REST APIs for processing and displaying Service Portal entities. The information about these REST APIs, as well as the guidelines for developing JSR portlets to be used within the Portal Manager solution can be found in the *Cisco Service Portal Integration Guide*.

Deploying JSR Portlets

Prerequisites

Deployment Descriptor

The portal front-end uses Apache Pluto 1.1 libraries for the portal framework. To deploy a JSR portlet into Service Portal, the portlets must be assembled with Pluto-specific information for deployment. Specifically, a servlet and servlet mapping are added to the deployment descriptor (web.xml). This servlet (org.apache.pluto.container.driver.PortletServlet) is used to dispatch portlet requests to the portlet application. For more detailed information, see the deployment instructions in the Apache web site.

The following is a sample **web.xml** file for a portlet called “CategoryPortlet”:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
                         "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
```

```
  <display-name>CategoryPortlet</display-name>
  <description>Category Portlet</description>

<servlet>
  <servlet-name>CategoryPortlet</servlet-name>
  <servlet-class>org.apache.pluto.container.driver.PortletServlet</servlet-class>
  <init-param>
  <param-name>portlet-name</param-name>
  <param-value>CategoryPortlet</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>CategoryPortlet</servlet-name>
  <url-pattern>/PlutoInvoker/CategoryPortlet</url-pattern>
</servlet-mapping>

</web-app>
```

Dependencies

On the JBoss 7 application server, the tag for “renderSingleLine” in pluto.tld should be commented out.

```
<!--
<tag>
    <name>renderSingleLine</name>
    <tagclass>org.apache.pluto.driver.tags.PortletRenderSingleLineTag</tagclass>
    <bodycontent>empty</bodycontent>
</tag>
-->
```

In addition, the portlet WAR file should include a file named “jboss-deployment-structure.xml”, located under the WEB-INF folder in the portlet WAR file, to describe the dependencies on the JBoss modules.

Here is the sample content for the XML file:

```
<jboss-deployment-structure>
  <deployment>
     <dependencies>
          <module name="javax.portlet" slot="main" export="true"/>
          <module name="org.apache.pluto.container.om" export="true"/>
          <module name="org.apache.pluto.container.driver" export="true"/>
          <module name="org.apache.pluto.tags" export="true"/>
      </dependencies>
  </deployment>
</jboss-deployment-structure>
```

nsAPI Java Client

If the nsAPI java client is used in the portlets, the related Service Portal and third-party libraries need to be included in the application package. A complete list of those dependent libraries and their locations can be found in the *Cisco Service Portal Integration Guide*.

Deployment Procedure

JBoss

Step 1 Create a subdirectory with the portlet name under the "<*JBOSS_HOME*>\standalone\deployments" folder; for example: <*JBOSS_HOME*>\standalone\deployments\<*portlet_name*>.

Step 2 Extract the portlet WAR file into the <*portlet_name*> directory that you just created.

Step 3 If the deployment descriptor has not been configured for the Apache Pluto portal server, modify web.xml accordingly. (See the "Deployment Descriptor" procedure on page 4-21.)

Step 4 If the server is already running, create a text file named <*portlet_name*>.dodeploy.

WebLogic

Step 1 Create a subdirectory with the portlet name under the "<*your_domain*>\applications" folder; for example:

<*BEA_HOME*>\ user_projects\domain\<your_domain>\applications\<*portlet_name*>

Step 2 Extract the portlet WAR file into the <*portlet_name*> directory that you just created.

Step 3 If the deployment descriptor has not been configured for the Apache Pluto portal server, modify web.xml accordingly. (See the "Deployment Descriptor" procedure on page 4-21.)

Step 4 (For clustered WebLogic environment only) If your portlet references the URL for the Request Center application, then specify the URL in the jsrportlet.properties file as "http://localhost:<port>/RequestCenter" where <port> is the port number used by each WebLogic server in the cluster. In other words, do not specify the URL as "http://<host_name>/RequestCenter" where <host_name> is the computer name of the web server or a specific server in the cluster.

Step 5 Use the WebLogic Administration Console to deploy the portlet application to the same WebLogic server (or cluster) as the Request Center application.

Step 6 Restart the WebLogic server (or cluster).

WebSphere

Step 1 Use the WebSphere Administration Console to deploy the portlet application to the same WebSphere server (or cluster) as the Request Center application. Make sure you choose the same virtual host and specify an appropriate context root for the web application.

Step 2 If the deployment descriptor has not been configured for the Apache Pluto portal server, modify web.xml accordingly. (See the "Deployment Descriptor" procedure on page 4-21.)

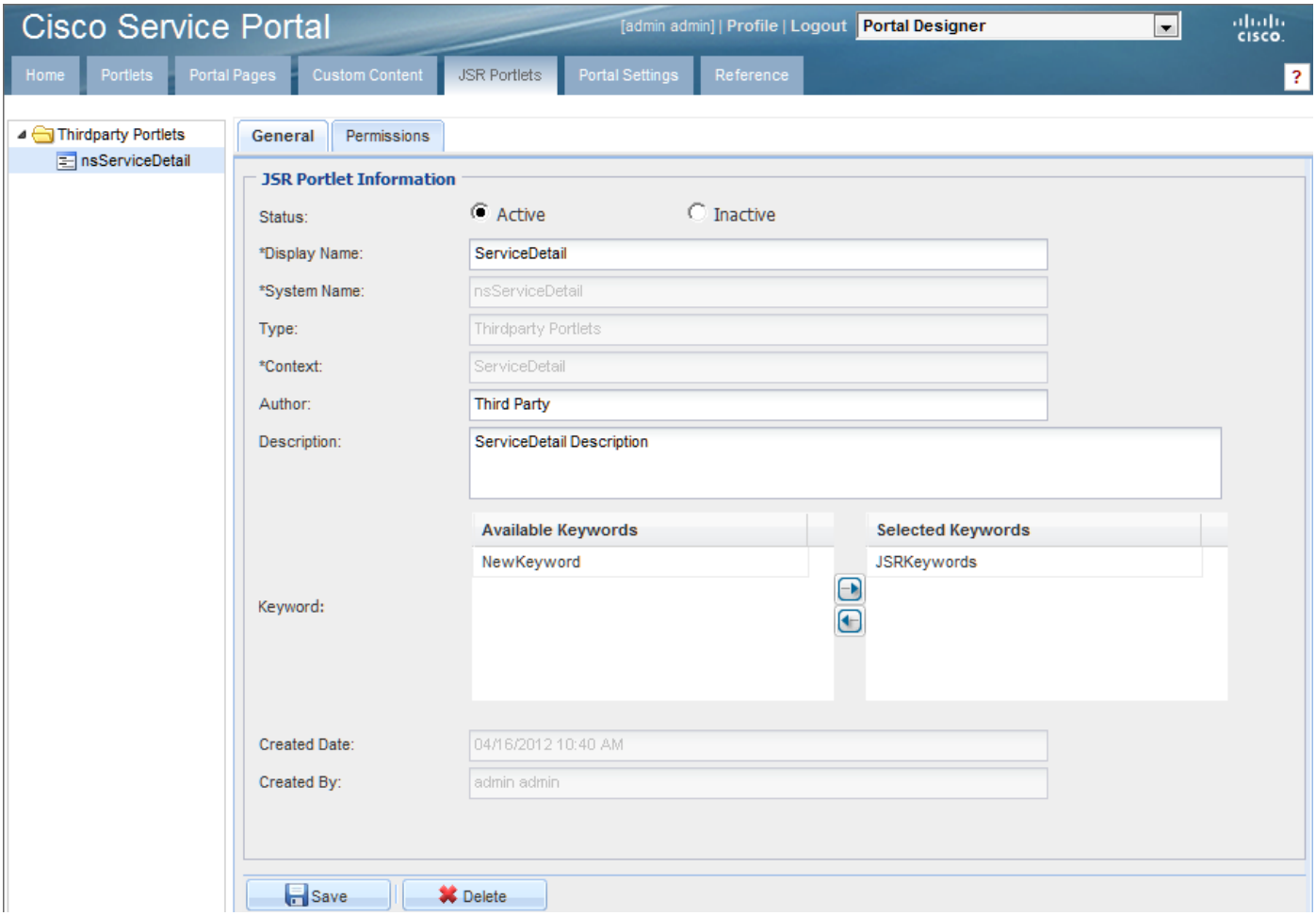
Step 3 (For clustered WebSphere environment only) If your portlet references the URL for the Request Center application, then specify the URL in the jsrportlet.properties file as "http://localhost:<port>/RequestCenter" where <port> is the port number used by each WebSphere server in the cluster. In other words, do not specify the URL as "http://<host_name>/RequestCenter" where <host_name> is the computer name of the web server or a specific server in the cluster.

Step 4 Restart the WebSphere server (or cluster).

Managing JSR Portlets

Adding JSR Portlets to the Portal

All successfully deployed JSR portlets will show up automatically on the JSR Portlets tab in the Portal Designer module. The portlets are placed into the "Third-Party Portlets" folder and their statuses are initially set to Inactive.



As with content portlets, access permissions are applied to the JSR portlets. To enable the portlet to be used on the portal, follow these steps:

Step 1 Set the portlet Status to **Active**.

Step 2 Modify the author and description as necessary for better documentation.

Step 3 Add appropriate keywords to the portlet to facilitate portlet search.

Step 4 **Save** the settings.

Step 5 On the Permissions subtab, grant the read permission to the appropriate entities.

Step 6 Edit the desired portal page and add the JSR portlet to the page, just as you would for content portlets.

Removing JSR Portlets from the Portal

Before a JSR portlet is made obsolete and permanently removed from the application server, all dependencies and associations to the portlet should be removed. To do this, you should remove the portlet from all portal pages that contain the portlet, and delete the portlet from the JSR Portlets tab. This would allow permissions and subscriptions to be dropped for the portlet. Finally the portlet can be undeployed from the application server.

Migrating JSR Portlets between Portals

The import/export of JSR portlets configurations is not supported at this time. The steps for entering general information and permissions for JSR portlets need to be repeated manually when the portlets are deployed to a Service Portal environment for the first time.

Portal Pages

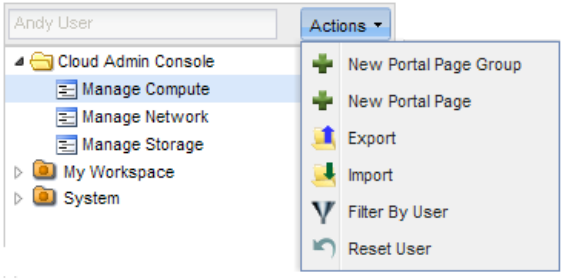
Once a portlet has been defined, made active, and made available to users via the appropriate permissions, it can be incorporated into portal pages. Two approaches are available for configuring portal pages:

- The portal designer can preconfigure the page by specifying its layout characteristics and including portlets as appropriate.
- Portal end users can dynamically incorporate portlets for which they have permission into their own pages and optionally save the portal pages.

Creating a Portal Page

To create or edit a portal page, click the Portal Page tab of Portal Designer.

All portal pages must be in a Portal Page Group. Both portal pages and page groups can be created by clicking on the **Actions** drop-down arrow and choosing the appropriate option.



Page groups serve as containers of portal pages for easier navigation and control of access permissions in Portal Designer. Page groups can also be exposed as modules in the Service Portal submenu to allow users to locate pages based on their grouping. Only users who have read permissions to those page groups can see them in the Service Portal submenu.

Service Portal includes two preconfigured page groups, both of which are displayed as modules in the module menu:

- System – This portal page group is reserved for site-wide information. The Site Homepage is located in this page group.
- My Workspace – This portal page group is accessible by all users and is available for users to place on their portal pages.

To mark a portal page group as a module, check the check box “Display As Module” on the portal page general information. Once the user has logged out and relogged in, the new module will appear in the Service Portal section of the module menu, placed beneath My Workspace and System, as the Cloud Admin Console module appears as shown below.

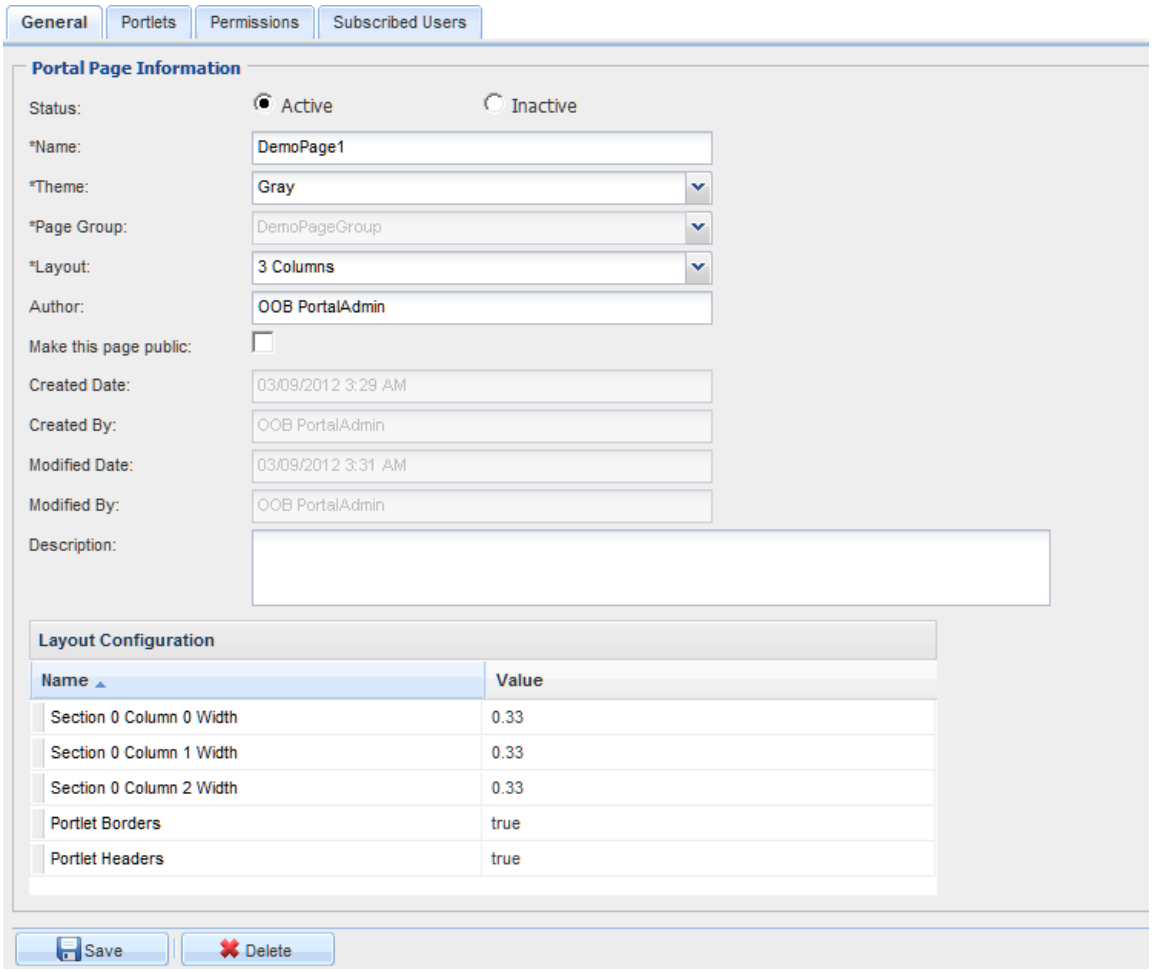
Portal Page General Information

A portal page must be created by specifying its name and page group. You can then use the General tab to view or modify the page configuration. The page group for a portal page cannot be modified once it has been specified because of the permissions already associated with the group.

The General information determines the overall look-and-feel of the page, including its color scheme (“Theme”) and layout.

All portal pages are created with an “Active” status. When the page status is set to “Inactive”, the portal page is hidden from Service Portal. Users who currently subscribe to the page will remain in the subscription record until the page is deleted. If an inactive page is still marked as the landing page for a person or an organizational unit, the setting is ignored, and users will land on their organizational unit homepage (if one is defined) or the Site Homepage instead when they first navigate to Service Portal.

The Portal Manager solution is distributed with a set of "Themes", color schemes, and styles. Portal pages are set to use the "Gray" theme by default. They can be configured to use other preconfigured themes both by portal designers and by portal users who have the "Manage Portal Page Theme" capability. For more information on themes and the styles used by the Portal, see the *Cisco Service Portal Configuration Guide.*

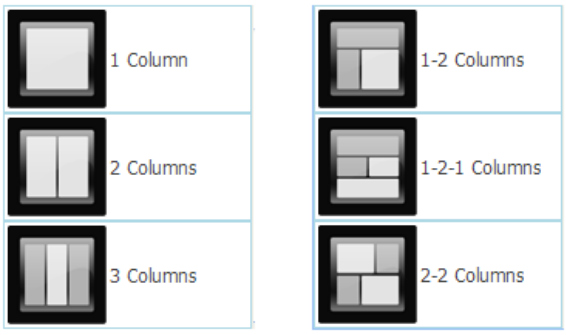


Fields used to specify general information about a portal page are summarized in the table below.

Field	Description
Status	The current status of the portal page; values are Active and Inactive.
Name	The name assigned to the page.
Theme	The default theme with which the portal page is displayed.
Page Group	The page group to which the portal page was assigned.
Layout	The layout for the portal page. Details on available layouts are given below.
Author	A text field containing the name of the author or other appropriate comments about the author; default value is the name of the user who created the portal page.
Make this page public	Making a portal page public makes it visible to other users. Users having appropriate permissions are able to subscribe to the page.

Layout Configurations

The portal page can be divided into sections and columns either vertically or a combination of vertically and horizontally. Available page layout formats are summarized below:



Additional properties of the Layout Configuration are inherited from the portlet's definition, and can be overridden on a page-by-page basis:

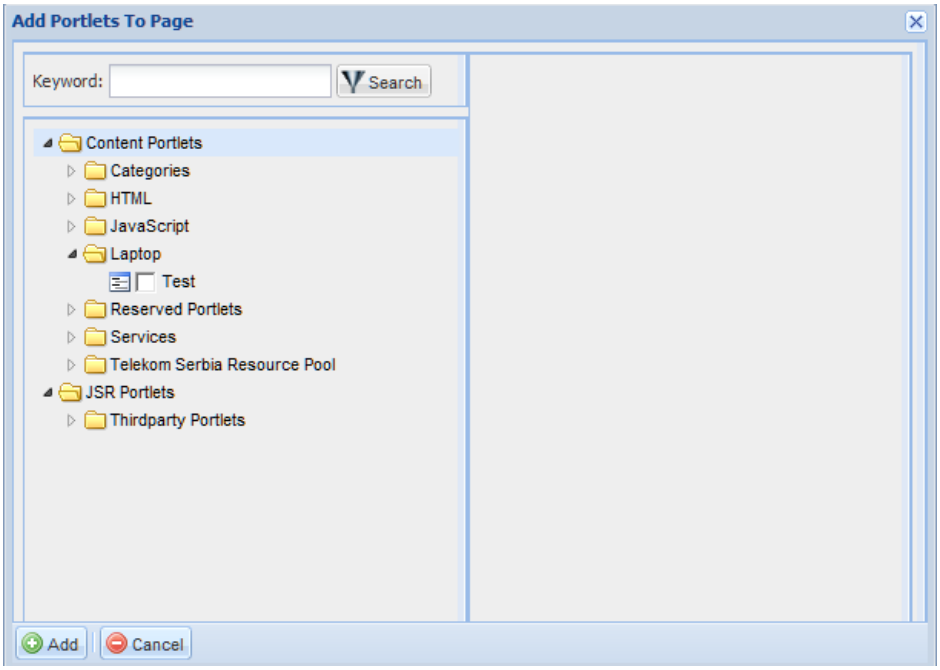
Field	Description
Section 0 Column <n> Width	For each column in the Layout, the user can specify the percentage of the browser width that the column should take up. The percentages should not exceed 100 percent. The number of sections available varies with the layout chosen.
Portlet Borders	True if each portlet should have a border around it; false otherwise.
Portlet Headers	True if the grid column headers should be displayed; false otherwise.

Portal Page Content

Use the Portlets subtab to include portlets on a portal page, configure their appearance, change the portlet configuration on a page, or remove a portlet from a page.

To add one or more portlets to a page:

Step 1 At the bottom of the page, click **Add Portlets to Page.** The "Add Portlets to Page" popup window appears, as shown below.

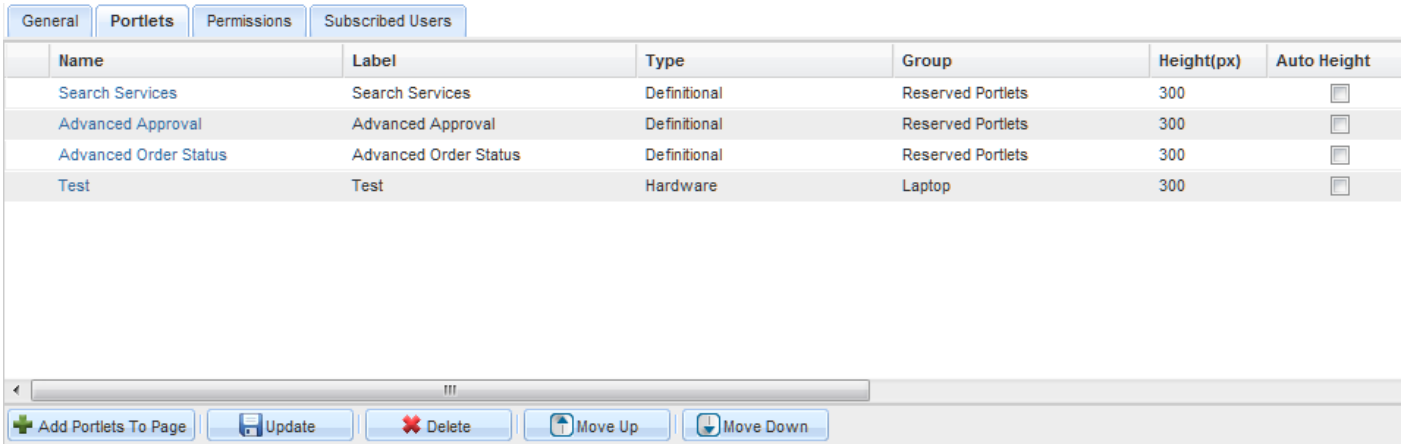


Step 2 If desired, you can filter the portlets displayed by entering a keyword and clicking **Search**.

Step 3 Expand the portlet groups displayed, until you find the portlets of interest. Highlight the name of a portlet to display a summary/description in the right-hand pane. Check the check box to the left of the portlet name to include the portlet on the page.

Step 4 Click **Add** to add the chosen portlets to the page.

The chosen portlets are added to the Portlets grid, as shown below.



When a portlet is added to a page, it is placed into the first section by default and set to display last on the page. The Section, Row and Column information in the grid indicates the location of the portlet on the page. The values in these fields cannot be modified directly but designers can change the position of a portlet by highlighting the portlet and clicking the “Move Up” or “Move Down” buttons.

If the portlet position needs to be substantially rearranged, designers may choose to do so in the portal front-end by adding the page to their portals, and using the mouse controls to drag the portlets to the desired location on the page.

To place a portlet into the second section or third section of a page with multisection layout (for example, 1-2, 1-2-1 or 2-2 columns), you need to edit the page in the portal front-end, use the mouse to drag the portlet to the bottom of the page and drop it when the second section outline appears. The third section, if available for the page layout, shows up only when one or more portlets have been placed into the second section.

Many of the properties shown are inherited from the portlet definition. Some of these (Name, Label, Type, Group) can be changed only by changing the portlet's configuration. Click on the Name to go to the Portlets tab to modify these properties. The remaining inherited properties can be overridden on a page-by-page basis.

Portal Page Permissions

The user who creates a portal page group or a portal page is automatically granted all access permissions to the object. For portal page groups, apart from the read and write permissions, the following permissions are also granted to the user:

- Read all pages in the group – Allows the user to view all the pages in the page group in Portal Designer. Also allows the user to subscribe to all the public pages in the page group in the portal front-end.
- Write all pages in the group – Allows the user to edit the settings and definition of all the pages in the page group in Portal Designer. Also allows the user to enter edit mode of the pages in the portal front-end.

Users without these permissions can only access individual pages for which they have read/write permissions. As with portlets, the Permissions subtabs for portal page group and page allow designers to designate which users should be able to access the portal object and the type of access to grant. See the "Portlet Permissions" procedure on page 4-11 for more information.

Subscribed Users

The Subscribed Users subtab provides a read-only view of users who have included the current page in their portal. Designers cannot remove user subscriptions but can prohibit users from accessing the page by setting the status of the page to "Inactive", or by removing the read access permissions.

Site Homepage

The Site Homepage is automatically provided within the System portal page group. Users with the Site Administrator role are granted read and write permissions to this portal page in Portal Designer. They can edit the page or grant access to the page to other users so that the page can be configured to include site-wide information that is of interest to the portal users.

The read permission to the page is granted to the system roles "Request Self-Service Roles" and "Request Governance Roles". This page is always displayed in Service Portal for users who are granted any child role of the above two container roles. The portal page also serves as the landing page for a portal user when the user's home organizational unit does not have a default landing page defined and the user has not set his/her own landing page preference.

Portal Settings

The Portal Settings tab allows designers to specify global data and settings for use in all portlets and portal pages. Available options are summarized below.

Option	Description
General > Common Settings	Site-wide settings for portal operations
Organizational Unit Settings	Default settings for individual organizational units
Keywords	Keywords that can be associated with a portlet to help users find portlets when designing a portal page
Authentication Settings	Authentication details for use with external sites to allow Single Sign-On (SSO) from enterprise login accounts to individual portlets, or to use group account credentials to automatically login to those sites

Common Settings

Common settings establish parameters for portal operations. The default settings are the recommended configurations. Changes to higher values may affect the application performance.

- Maximum Number of Tabs in Portal – The maximum number of portal pages that can be displayed in My Workspace or any other portal module. The default setting is 6 and applies to all portal users. The highest value allowed for this setting is 10.
- Maximum Number of Portlets on a Tab – The maximum number of portlets that can be included on a portal page. The default setting is 6 and applies to all portal users. The highest value allowed for this setting is 10.
- Maximum Number of Grid Portlets on a Tab – The maximum number of grid portlets that can be included on a portal page. The default setting is 4 and applies to all portal users. The highest value allowed for this setting is 6.
- Maximum Number of Private Pages in Portal – The maximum number of private pages a user can maintain in the portal. The default setting is 2. This restriction is not applied to users with the "Override Private Portal Page Limit" capability. The highest value allowed for this setting is 10.
- nsAPI Page Size for Transactional Data – The default number of records returned by portlets and Service Portal API clients for Requisitions, Requisition Entries, Authorizations, and Tasks (when the page limit is not specified in the API call). The default setting is 10. The highest value allowed for this setting is 50.
- nsAPI Page Size for Directory Data – The default number of records returned by portlets and Service Portal API clients for People, OUs, Groups, and Accounts (when the page limit is not specified in the API call). The default setting is 10. The highest value allowed for this setting is 50.
- nsAPI Page Size for Service Item and Standard Data – The default number of records returned by portlets and Service Portal API clients for Standards and Service Items (when the page limit is not specified in the API call). The default setting is 10. The highest value allowed for this setting is 50.
- nsAPI Page Size for Definitional and Custom Data – The default number of records returned by portlets and Service Portal API clients for Categories, Services, Offerings, Agents, Agreements, and User-defined Contents (when the page limit is not specified in the API call). The default setting is 10. The highest value allowed for this setting is 50.

General

Setting	Value	Description
Maximum Number of Tabs in Portal	6	The maximum number of portal pages allowed for display in the Service Portal. Default = 6
Maximum Number of Portlets on a Tab	6	The maximum number of portlets allowed to be included on a portal page. Default = 6
Maximum Number of Grid Portlets on a Tab	4	The maximum number of grid portlets allowed to be included on a portal page. Default = 4
Maximum Number of Private Pages in Portal	2	The maximum number of private pages an end user can maintain in Service Portal. Default = 2
nsAPI Page Size for Transactional Data	10	Default number of records returned by nsAPI and/or Portlets when the page limit is not specified. (Note: This setting is applicable for Requisitions, Requisition Entries, Authorizations and Tasks.)
nsAPI Page Size for Directory Data	10	Default number of records returned by nsAPI and/or Portlets when the page limit is not specified. (Note: This setting is applicable for People, OUs, Groups and Accounts.)
nsAPI Page Size for Service Item And Standard Data	10	Default number of records returned by nsAPI and/or Portlets when the page limit is not specified. (Note: This setting is applicable for Standards and Service Items.)
nsAPI Page Size for Definitional and Custom Data	10	Default number of records returned by nsAPI and/or Portlets when the page limit is not specified. (Note: This setting is applicable for Categories, Services, Offerings, Agents, Agreements, User-defined Contents.)

Update

Organizational Unit Settings

The Organizational Unit Settings page allows a home page to be configured for individual organizational units (OUs). A portal page so designated is always displayed in the portal for users who have the organizational unit as their Home OU.

Users who have a homepage defined for their Home OU will see two pages in My Workspace—the OU Homepage on the first tab, and the Site Homepage on the second tab. They can add/create more pages and choose to land on a different portal page by setting the desired page as the landing page for themselves. More information regarding the end user view of Service Portal can be found in the "An End User's View of the Portal" section on page 4-46.

Keywords

Keywords provide flexibility in portlet search when designers or portal users are looking for content to include in a portal page. Once defined in the Common Settings, keywords can be associated with a portlet on the Keyword page itself, or on the General Information subtab for portlets.

Keywords used for portlets are distinct/disjoint from the Keywords used/associated with services (service definitions).

Authentication Settings

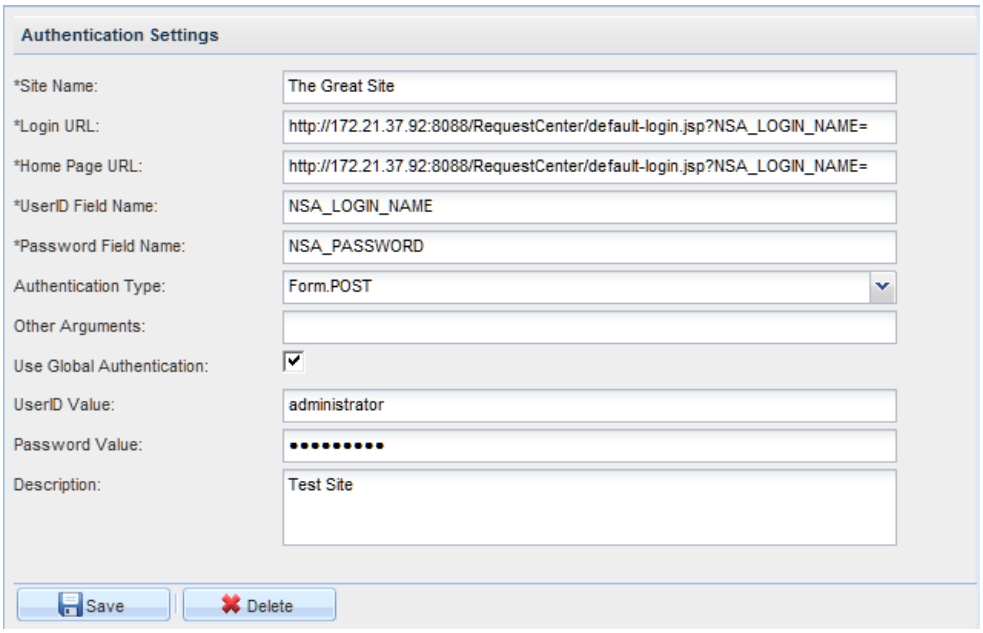
The Authentication Settings page allows you to specify Single Sign-On information for connecting to a site that requires a user name and password or other information to be supplied as part of a logon procedure. To connect to a site that requires a login:

- Create an External Site profile specifying the site name and the login URL, identifying the fields on the login page where the user is expected to enter a user name and password; and specifying the authentication type.

- If the same connection criteria can be used for all users accessing the site through the portal, specify global authentication criteria.
- Design a portlet that has automatic login enabled and uses the external site.
- The portlet can be included on a portal page. If global authentication arguments are specified, they will automatically be passed to the external site for authentication. If global authentication is not used, the user can enter site connection credential via the portal's Edit Password tab accessible in the portal's Edit Mode.

Entering Authentication Details

A sample Authentication Settings page is shown below.



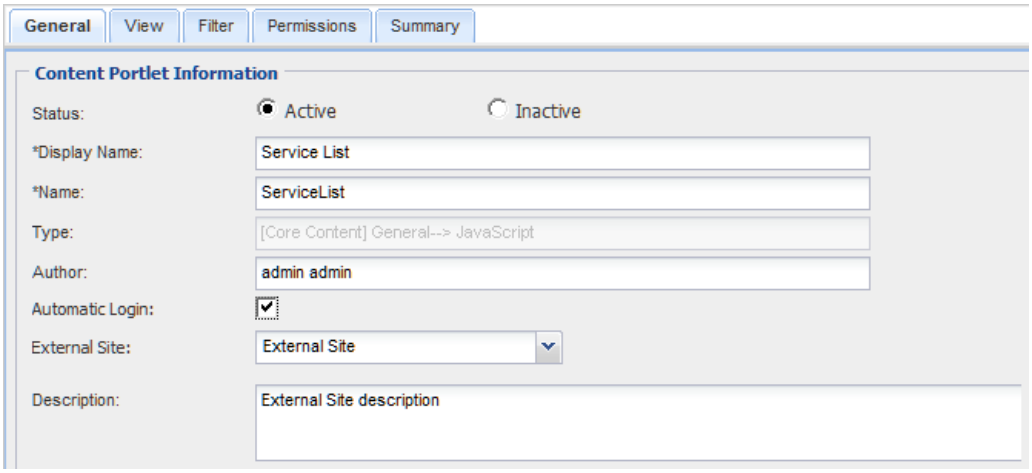
Fields used to define external site authentication settings are summarized in the table below.

Field	Description
Site Name	The name of the site.
Login URL	The URL that identifies the site's login page.
Home Page URL	The URL that identifies the landing page of the application to be displayed.
UserID Field Name	The field on the page specified by the Login URL which contains the user's ID/user name.
Password Field Name	The field on the page specified by the Login URL which contains the user's password.
Authentication Type	URL, Form.GET, or Form.POST.
Other Arguments	Additional arguments that must be passed to the site. These are passed in a format depending on the authentication type. The typical format is *arg1=value1&arg2=value2*.

Field	Description
Use Global Authentication	A check box that indicates that the User ID and Password values specified below are used for all users to connect to the site.
UserID Value	The value of the User ID to be passed to the site for global authentication.
Password Value	The value of the Password to be passed to the site for global authentication.
Description	Free-format text which describes this site; documentation only.

Defining a Portlet with Automatic Authentication

You can associate a portlet that makes use of automatic authentication with the external site by choosing the site name and checking the **Automatic Login** check box.

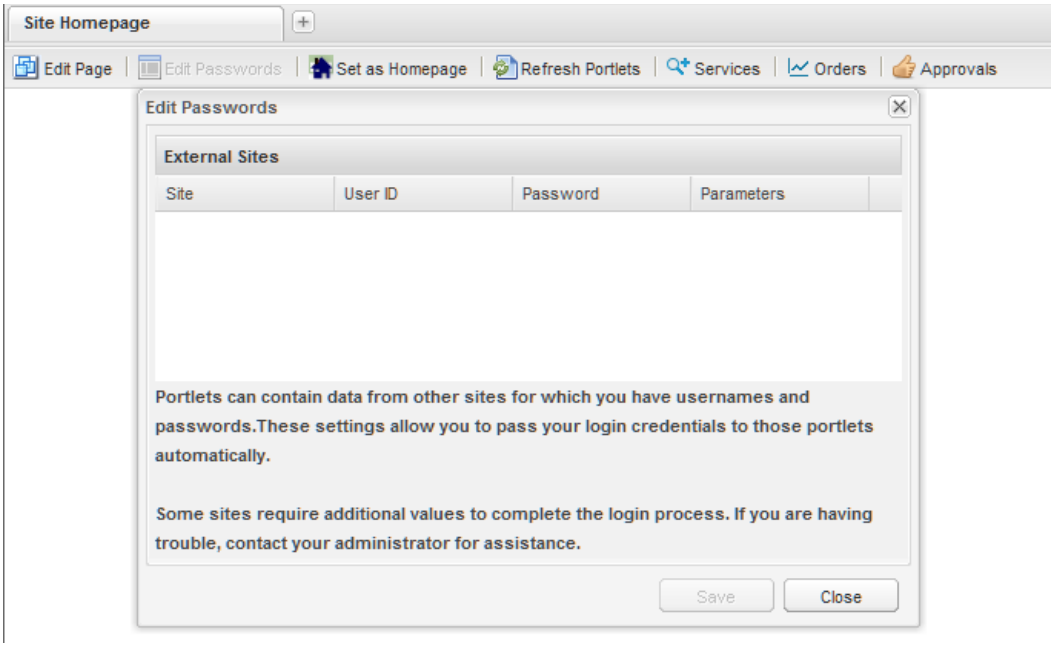


Accessing an External Site in the Portal

Portlets with automatic authentication enabled for external sites can be included on any portal pages.

If the included site has been defined with global authentication, the application will pass the preset credentials to the external site for authentication and display the site's portlet on the current portal page.

If the included site does not have global authentication enabled, the application will attempt to connect to the portlet with the site authentication credentials that the user has previously defined through Edit Password settings in the portal front-end:



The Edit Passwords page lists all sites associated with portlets displayed on the user's portal page that are enabled for automatic authentication but do not use global authentication. The user may enter connection information for all external sites and save the settings for use in other portal pages. The saved settings are preserved for the current session in which the portal page is displayed or until the associated site times out. If the user navigates to another page or module and returns to the portal page, automatic authentication will take place again.

Users are prompted for login or authentication exceptions by the external site if:

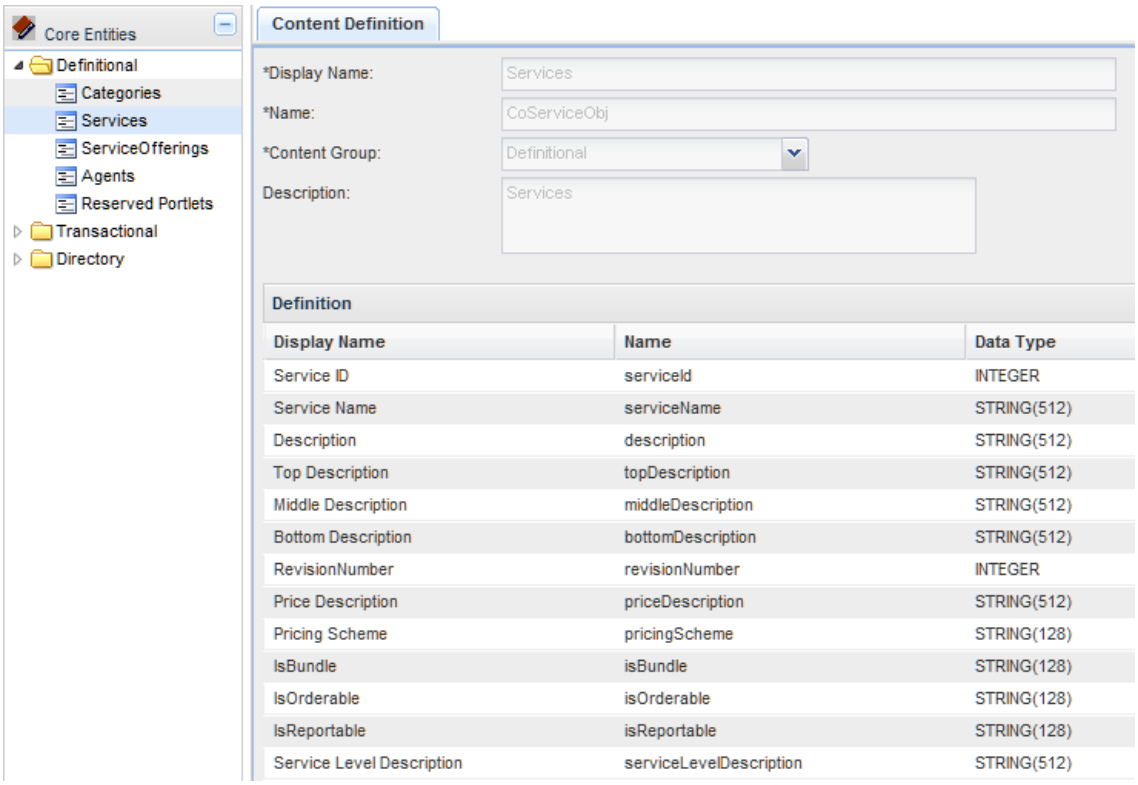
- the connection to the external site has already timed out when the user attempts to refresh the data in the portlet, or
- the credentials defined for the user have expired or are incorrect, or
- no credentials are found that match the site name referenced in the portlet.

Reference Data

The Portal Manager solution allows you to integrate views of Service Portal entities (objects) into portlets. Such reference data cover the following types of objects:

- **Core Entities** represent the definitions of Request Center and Demand Center entities, such as categories, and services, as well as directory data (people and organizations) and actual transactional data on tasks and requisitions.
- **HTML/JavaScript** reference data refers to portlets defined in Portal Designer of the corresponding type.
- **Service Items** defined in the Service Item Manager module of Lifecycle Center, used to track corporate assets that have been ordered or updated via service requests.
- **Standards** defined in the Service Item Manager module of Lifecycle Center, used to enforce data entry rules or otherwise standardize the configuration of service items or other orderable assets.

Reference data serves as a quick reference to the definitions of Service portal entities. A list of the attributes in each object that comprises the reference data is given in the "Content Definition" tab displayed when that object is chosen from the Reference Data list panel, as shown in the sample below.



Display Name	Name	Data Type
Service ID	serviceId	INTEGER
Service Name	serviceName	STRING(512)
Description	description	STRING(512)
Top Description	topDescription	STRING(512)
Middle Description	middleDescription	STRING(512)
Bottom Description	bottomDescription	STRING(512)
RevisionNumber	revisionNumber	INTEGER
Price Description	priceDescription	STRING(512)
Pricing Scheme	pricingScheme	STRING(128)
IsBundle	isBundle	STRING(128)
IsOrderable	isOrderable	STRING(128)
IsReportable	isReportable	STRING(128)
Service Level Description	serviceLevelDescription	STRING(512)

All attributes listed in the "Definition" section are available for inclusion in the portlet views. Detailed descriptions of these columns are given in the sections below.

Content Definition

The content definition subtab of the Reference Data tab lists details on each of the entities that can be included in a portlet. This subtab is read-only, allowing portal designers to review entity definitions before including them in a portlet.

Field	Description
Display Name	The name of the object displayed in Portal Designer.
Name	The system name of the entity as stored in the portlet content metadata tables.
Content Group	The categorization of content type; for core entities, the entity type (definitional, directory, or transactional); for service items and standards, the service item group or standards group, respectively.
Description	A description of the entity.
Definition	
Display Name	The name of the attributes that comprise the entity. This is the column name displayed by Portal Designer and within portlets.
Name	The name of the database column containing the attribute data.
Data Type	The data type of the attribute/column.

In addition to the attributes which comprise the entity, the content definition includes one or more "URL" as the last attribute.

In a portlet, the URL attribute generates a clickable link which takes the user to the Request Center, Demand Center or Lifecycle Center module to bring up the entity details or the actionable view of the entity on a popup page. The user interface is the same as the one that user would see by navigating through the search views in those modules. The only difference is that the user stays in the portal and does not lose the context once the review/action is completed for the entity and the popup page is closed.

Users must have the required RBAC capabilities to navigate to other modules via the URL links; otherwise, an insufficient permission error appears.

Core Entities

Core entities allow portal designers to expose information on application transactional, definitional, and directory data to portal users. In general, the attributes available correspond to the fields displayed on the corresponding user interfaces in Request Center, Demand Center and Lifecycle Center, and should be familiar to users of those modules.

Categories

Categories are used to group services in the service catalog for presentation to end users who may wish to browse or order those services.

Column (Display Name)	Description
Category ID	Internal ID of the category
Category Name	Name of the category
Description	Description of the category
isRoot	Whether the category is a root category, that is, “Consumer Services” or “Service Offerings”
TopDescription Enabled	Whether the top section of category details is enabled in the category presentation
Top Description	HTML defined in the top section of category details in the category presentation
TopDescription URL	URL defined in the top section of category details in the category presentation
MiddleDescription Enabled	Whether the middle section of category details is enabled in the category presentation
Middle Description	HTML defined in the middle section of category details in the category presentation
MiddleDescription URL	URL defined in the middle section of category details in the category presentation
BottomDescription Enabled	Whether the bottom section of category details is enabled in the category presentation
Bottom Description	HTML defined in the bottom section of category details in the category presentation
BottomDescription URL	URL defined in the bottom section of category details in the category presentation
Category Image	Relative URL for the category image within Request Center.war
CatalogType ID	Internal ID of the category type: 1-Consumer Service category, 2-Service Offering category
Description URL	(Not Used)
Category URL (My Services)	Relative URL link for accessing the category in the My Services – Category Overview tab
Category URL (Relationship Manager)	Relative URL link for accessing the category in the Relationship Manager Category Overview tab
Category URL (MYSE)	Relative URL link for accessing the category in the My Services Executive Category Overview tab

Services

Column (Display Name)	Description
Service ID	Internal ID of the service
Service Name	Name of the service
Description	Description of the service
Top Description	HTML defined in the Overview section of service details in the service presentation, shown only when the display is set to Show
Middle Description	HTML defined in the More Details section of service details in the service presentation, shown only when the display is set to Show
Bottom Description	HTML defined in the Service Form section of service details in the service presentation, shown only when the display is set to Show
RevisionNumber	Internal version number of the service
Price Description	Description of how the service is priced, shown only when the Pricing Summary display is set to “Display both cost and price” or “Display only price”
Pricing Scheme	Pricing scheme of the service, shown only when the Pricing Summary display is set to “Display both cost and price” or “Display only price”
IsBundle	Whether the service is a bundle
IsOrderable	Whether the service is orderable
IsReportable	Whether the service is reportable in the Advanced Reporting module
Service Level Description	Description of the service level as defined in the service general information
Status	Status of the service; possible values are Active and Inactive
Status ID	Internal ID for the status of the service; possible values are: 1 (Active), 2 (Inactive)
Expected Duration	Expected duration of the service in hours
Expected Duration Units	Units of measure to be used when displaying the service; possible values are “Business Days” and “Hours”
Price	Price of the service, shown only when the Pricing Summary display is set to “Display both cost and price” or “Display only price”
Can Start Later	Whether future delivery is allowed for the service
Date Quality ID	Forecasting method defined for the service; possible values are: 2 (Estimate Due Date from task durations) 3 (Approximate Due Date using Standard Duration) 4 (Do not forecast Due Date)
Service Image	Relative URL link for the service image in the form of servlet reference within RequestCenter.war
Service URL	Relative URL link for accessing the service in the My Services Service Overview page
Service Order URL	Relative URL link for accessing the service in the My Services Service Order page

Service Offerings

Column (Display Name)	Description
Offering ID	Internal ID of the service offering
Offering Name	Name of the service offering
Status ID	Internal ID of the service offering status; possible values are: 1 (Draft), 2 (Active), 3 (Inactive)
Status	Status of the service offering; possible values are: Draft, Active, Inactive
Description	Description of the service offering in the form of HTML code
Price Description	Price description of the service offering in the form of HTML code
Service Level Description	Service level description of the service offering in the form of HTML code
Component Services Description	Component services description of the service offering in the form of HTML code
Fiscal Year	Fiscal year for the service offering
Owner ID	PersonID of the owner (author) of the offering
Owner Name	First and last name of the owner (author) of the offering
Creation Date	Creation date of the service offering
Expiration Date	Expiration date of the service offering, which is the last day of the fiscal year
Offering Price	External price of the offering
Offering URL (Relationship Manager)	Relative URL link for accessing the service offering in the Relationship Manager Home tab (View Service Offerings)
Offering URL (Service Level Manager)	Relative URLlink for accessing the service offering in the Service Level Manager Home tab (View Service Offerings)

Agents

Agents are used by Service Link, the integration hub of Service Portal, to provide an interface between Service Portal service requests and third-party systems such as help desks, inventory control systems, purchasing systems, or other external applications.

Column (Display Name)	Description
Agent ID	Internal ID of the agent
Agent Name	Name of the agent
Description	Description of the agent
Status ID	Internal ID of the agent status; possible values are: 1 (Active), 2 (Inactive)
Action	Action to be performed by the agent, as seen in the task general information in Service Designer
Context Type ID	Internal ID for the context of external task; possible values are: 1 (Service Task), 2 (Service Item Task)
Inbound Adapter Name	Name of the adapter used for inbound action
Outbound Adapter Name	Name of the adapter used for outbound action
InboundTransformation	Name of the transformation used for inbound document

Column (Display Name)	Description
OutboundTransformation	Name of the transformation used for outbound document
Failed Email	Name of the email notification used for failure
Status	Status of the agent; possible values are: Active, Inactive
Agent URL	Relative URL link for accessing the agent in the Service Link Manager Integration tab

Agreements

Column (Display Name)	Description
Agreement ID	Internal ID of the agreement
Agreement Name	Name of the agreement
Offering ID	Internal ID of the offering from which the agreement was created
Offering Name	Name of the offering from which the agreement was created
Status ID	Internal ID of the agreement status; possible values are: 1 (Active), 2 (Expired), 3 (Business Manager Approved), 4 (Draft), 5 (Relationship Manager Approved), 6 (Inactive)
Status	Status of the agreement; possible values are: Active, Expired, Business Manager Approved, Draft, Relationship Manager Approved, Inactive
Account ID	Internal ID of the account for which the agreement is made
Account Name	Name of the account for which the agreement is made
Description	Description of the underlying offering for the agreement
Price Description	Price description of the underlying offering for the agreement
Service Level Description	Service level description of the underlying offering for the agreement
Owner ID	PersonID of the owner (author) of the agreement
Owner Name	First and last name of the owner (author) of the agreement
Effective Date	Effective date of the agreement, which is also the creation date
Start Date	Start date of the agreement, which is the day on which the agreement was made active.
Expiration Date	Expiration date of the agreement, which is the last day of the fiscal year
Total Price Estimate	Total price estimate of the agreement
Actual To Date	Total actual charges (YTD) for the agreement
Budget To Date	Total estimated charges (YTD) to the agreement
Total Agreement Budget	(Not Used)
Fiscal Year	Fiscal year for which the agreement was created
Agreement URL (Relationship Manager)	Relative URL link for accessing the agreement in the Relationship Manager Agreement tab
Agreement URL (Service Level Manager)	Relative URL link for accessing the agreement in the Service Level Manager Agreement tab

Requisitions

Requisitions are the service requests that have been submitted by Request Center users.

Column (Display Name)	Description
Requisition	Internal ID of the requisition
Name	Display name of the requisition; normally the first service included in the requisition
Initiator Id	PersonID of the initiator of the requisition
Customer Id	PersonID of the customer of the requisition
Expected Duration	(Not used)
Actual Duration	Actual duration, measured in hours, to complete the requisition
Due Date	Date on which the requisition fulfillment is expected to complete
Closed Date	Date on which the requisition was actually set to Closed status
Expected Cost	Price of the requisition
Status	Status of the requisition; possible values are: Ongoing, Closed, Rejected, Cancelled, Delivery Cancelled
Initiator	First and last name of the initiator of the requisition
Customer	First and last name of the customer of the requisition
Bill To	Name of the home organizational unit of the customer of the requisition
Submit Date	Date on which the requisition was submitted
Requisition URL	Relative URL link for accessing the requisition details page in My Services

Authorizations

Authorizations are any approvals or reviews required in conjunction with completing fulfillment of a service request. The columns cover those that are presented in the Authorization tab in My Services.

Column (Display Name)	Description
Requisition	Requisition ID associated with the task
Total Price	Total price of the requisition associated with the task
Due On	Due date of the task
Task Name	Name of the task
Service Name	Name of the service associated with the authorization. When there are multiple services for the authorization, only the first service name is shown.
Customer	Name and home organizational unit of the customer for the associated requisition, presented in the format {FirstName LastName} : {OU Name}
Performer	First and last name of the performer of the task
Status	Status of the authorization task; possible values are: Under review, Being approved, Reviewed, Approved, Rejected
Priority	Priority of the authorization task; possible values are: High, Normal, Low
Authorization ID	Internal ID (Task ID or Activity ID) of the task
Authorization URL	Relative URL link for accessing the task data page in Service Manager

Tasks

Tasks are activities associated with a request, including reviews, authorizations and fulfillment tasks. The columns cover those that are presented in the Home tab of Service Manager.

Column (Display Name)	Description
Task Id	Internal ID of the task (aka Activity ID)
Task Name	Name of the task
Requisition	Requisition ID associated with the task
Due Date	Due date of the authorization task
Service Name	Name of the service associated with the task
Initiator	First and last names of the initiator of the requisition associated with the task
Customer OU	Name of the home organizational unit of the customer for the associated requisition
Customer Name	First and last names of the initiator of the requisition associated with the task
Performer	First and last names of the performer of the task, if the task is assigned to a person
Queue	Name of the queue assigned to perform the task, if the task is assigned to a queue
Status	Status of the task; possible values are: New, Ongoing, Under review, Being approved, Completed, Reviewed, Approved, Rejected, Skipped, Cancelled, Scheduled, Review Submitted, Approval Submitted
Scheduled Start Date	Scheduled start date of the task
Effort	Effort estimated for the task
Task URL	Relative URL link for accessing the task data page in Service Manager

Organizational Units

Organizations are the business units and service teams into which users are organized.

Column (Display Name)	Description
OrganizationUnit Name	Name of the organizational unit
Description	Description of the organizational unit
Organizational Unit ID	Internal ID of the organizational unit
Parent ID	Internal ID of the parent organizational unit
Parent Name	Name of the parent organizational unit
Organizational Unit Type ID	Internal ID of the organizational unit type; possible values are: 1 (Business Unit), 2 (Service Team)
Status ID	Internal ID of the status of the organizational unit; possible values are: 1 (Active), 2 (Inactive)
Status	Status of the organizational unit; possible values are: Active, Inactive

Column (Display Name)	Description
Manager ID	Person ID of the person assigned to the organizational unit manager functional position
Manager Name	First and last names of the person assigned to the organizational unit manager functional position
isBillable	Whether the organizational unit is marked as billable
Organizational Unit URL	Relative URL link for accessing the organizational unit general information page in Organization Designer

Persons

Persons are individual users as defined in Organization Designer.

Column (Display Name)	Description
Person ID	Internal ID of the person
First Name	First name of the person
Last Name	Last name of the person
Email	Email address of the person
HomeOrganizationalUnit ID	Internal ID of the home organizational unit of the person
HomeOrganizationalUnit Name	Name of the home organizational unit of the person
TimeZone ID	Internal ID of the time zone of the person
TimeZone Name	Name of the time zone of the person
Login Name	Login name of the person
Birth Date	Birth date of the person
Hire Date	Hire date of the person
Title	Title of the person
Employee Code	Employee code of the person
Locale ID	Internal ID of the locale of the person
Language Code	Internal ID of the preferred language for the person
Language Name	Preferred language for the person
Supervisor ID	Person ID of the supervisor of the person
Supervisor Name	Name of the supervisor of the person
Status	Status of the person; possible values are: Active, Inactive
Person URL	Relative URL link for accessing the person general information page in Organization Designer

Groups

Groups are a user-defined grouping of OUs or people that can be used in the assignment of work, roles and permissions.

Column (Display Name)	Description
Group ID	Internal ID of the group
Group Name	Name of the group
Description	Description of the group
Status ID	Internal ID of the status of the group; possible values are: 1 (Active), 2 (Inactive)
Status	Status of the group; possible values are: Active, Inactive
Parent ID	Internal ID of the parent of the group
Parent Name	Name of the parent of the group
Group URL	Relative URL link for accessing the group general information page in Organization Designer

Accounts

Accounts are user-defined grouping of OUs for use to establish agreements in Demand Center.

Column (Display Name)	Description
Account ID	Internal ID of the account
Account Name	Name of the account
Description	Description of the account
Created Date	Date on which the account was created
Owner ID	Person ID of the owner (author) of the account
Owner	First and last name of the owner (author) of the account
Account URL	Relative URL link for accessing the account general information page in Relationship Manager

HTML/JavaScripts

The HTML/JavaScripts objects are those HTML/Java script portlets that have been designed in Portal Designer.

Service Items

Both system- and user-defined service items are available for display in portlets. For user-defined service items, the attribute names and data types correspond to those defined at your site—see your service catalog design team for more information. For the system-defined service items for Virtual Hardware, attributes are described in detail in Chapter 3, “Lifecycle Center”.

Standards

Both system- and user-defined standards are available for display in portlets. For user-defined standards, the attribute names and data types correspond to those defined at your site—see your service catalog design team for more information. For system-defined standards (related to Virtual Hardware), attributes are described in detail in Chapter 3, "Lifecycle Center".

An End User's View of the Portal

Overview

Until now, the default home page for Service Portal end users has been the Request Center My Services module. All Request Center users can browse the service catalog, viewing services which they have permission to order, and request services for themselves. They can also view any requisitions they have previously ordered. In addition, the My Services home page is configurable so that designated users or groups of users can order on behalf of other users; view their authorizations or perform authorizations themselves; and view their service items or service items for others in their organizational units.

Unfortunately, the My Services page is not customizable by its users. It is customizable only to the extent that administrators can configure the page to allow or disallow some of the capabilities described above and can set up organization-specific style sheets to change the colors, fonts, or graphics displayed on the page.

The Portal Manager solution provides an alternative to replace the My Services page with an end-user customizable page, customizable within the constraints set up and maintained by administrators. Through Portal Designer capabilities described previously in this chapter, administrators can define any number of portlets and grant permission to users to read or edit data displayed in these portlets.

Portal Modules

The Service Portal option section in the module menu represents a collection of submodules available in the portal. My Workspace and System modules are always listed on top, followed by the user-defined modules. A module is visible to users only if the corresponding portal page group has the "Display As Module" check box checked in the definition and the users have read permission to that portal page group. Once the user has logged out and relogged in, the new module will appear in the Service Portal section of the module menu, placed beneath My Workspace and System, as the Cloud Admin Console module appears as shown below.

My Workspace Module

My Workspace, which is a system-defined portal page group, is the group in which users create and maintain their own portal pages. It is also the default landing module when user selects the Service Portal menu option. For these reasons, all service portal users must be granted the read/write permission to the My Workspace portal page group. Users who have any of the Request Self-Service or Request Governance system roles will automatically be given the read/write access to this portal page group.

System Module

System module is another system-defined portal page group that is accessible to all users who have any of the Request Self-Service or Request Governance system roles. The Site Homepage is located in this module. More portal pages can be created by the administrators in this group for displaying site-wide information.

Portal Home Pages

A user's preconfigured portal pages within a module are accessible as tabs along the top of the window. The home page is the preferred landing page for a module. Users can set any of the portal pages within the module as the home page. This action will automatically move the page to be the first tab.

Site Homepage

The Site Homepage is the default landing page for My Workspace when no other homepage has been configured. As a good practice, all portal users should be given read access to this portal page. Users who have the Request Self-Service and Request Governance system roles will automatically have read permission to the page.

Organizational Unit Homepage

In a distributed portal administration model, portal designers may prefer to set up different landing pages for different organizational units to allow end users to access the organization-specific content. Users whose home organizational units are configured with a homepage in the Portal Designer settings will have that page as the first tab in My Workspace, followed by the Site Homepage.

User Homepage

When a user sets a portal page other than the Site or OU homepages as his default landing page, the chosen page becomes the first tab in My Workspace, followed by OU homepage (if one is defined) and then the Site Homepage.

View Mode of a Portal Page

The Portal is originally displayed in View Mode which allows users to perform a limited set of actions on the page.

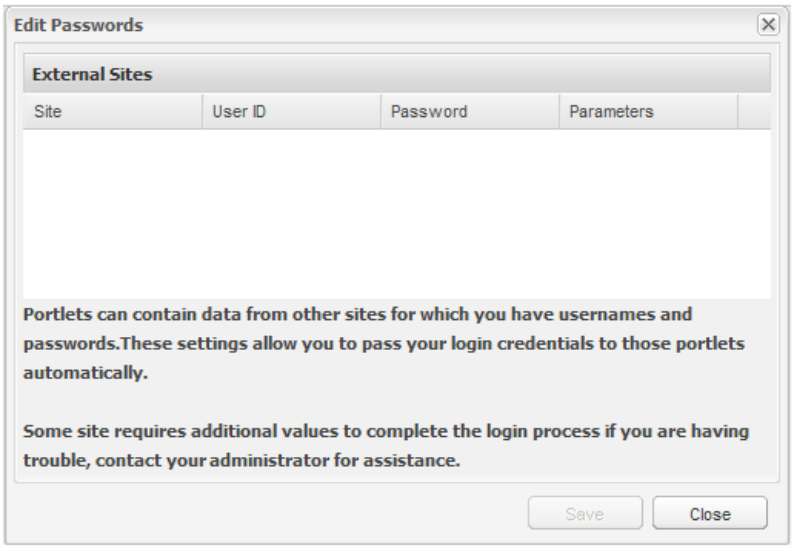
Edit Page

Users who are granted write permissions to the portal page are able to edit the content and appearance of the page by clicking **Edit Page**. The only exception is when the page contains inactive portlets or portlets to which the user has no access permissions.

Once in Edit Mode, a different set of actions is available (see the "Edit Mode of a Portal Page" section on page 4-49).

Edit Passwords

External sites that require individual authentication settings can be maintained to allow automatic authentication every time the portlet is accessed. The credentials maintained here are not accessible by other users, including the Portal Designer users.



Set as Homepage

The action marks the current active tab as the home page and moves it to the first tab position. When there is only one portal page in the current module, the button is disabled.

Refresh Portlets

The Refresh Portlets button reloads the content of all portlets on the page.

Search, Orders, and Approvals

Clicking the **Search**, **Orders**, or **Approvals** button is a shortcut to add the Search, Order Status, and Approvals portlet, respectively, to the page, rather than using the Add Content button (see the "Add Content" section on page 4-49). See the "Reserved Portlets" section on page 4-12 for information on these Reserved portlets.

Once a portlet is added to the page in View Mode, it is automatically saved to the page, and cannot be removed unless you enter Edit Mode (see the "Edit Mode of a Portal Page" section on page 4-49).

Note These buttons are disabled if the portlet has already been added to the current page. It is hidden if the user does not have Page Write access or appropriate RBAC permissions.

Edit Mode of a Portal Page

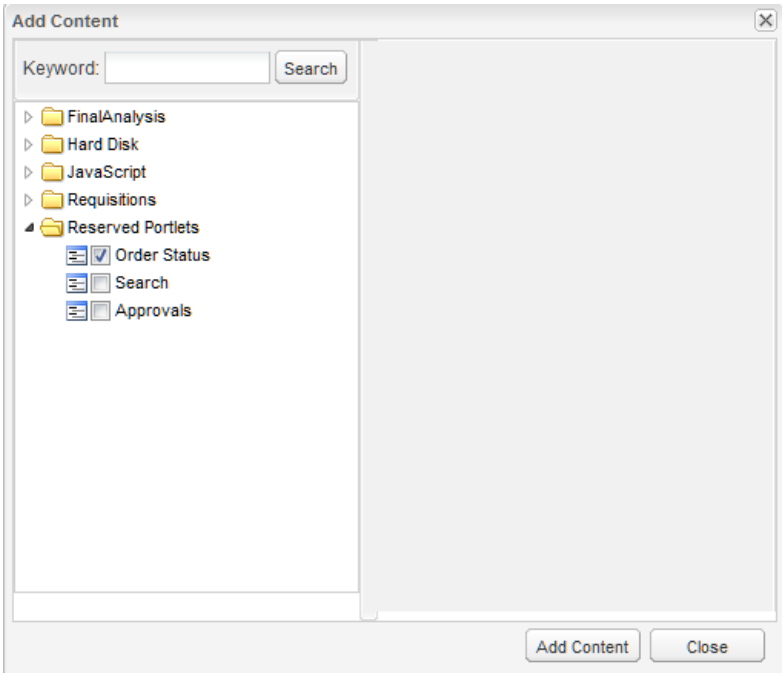
Once in Edit Mode, the toolbar shows the following actions:

Finished Editing Page

The Finished Editing Page button takes users back to View Mode (see the "View Mode of a Portal Page" section on page 4-47). If you have made any changes, be sure to click **Save Page** to save your edits before you click **Finished Editing Page**.

Add Content

The Add Content button allows users to add one or more portlets to the current page by choosing the appropriate check boxes and clicking **Add Content**. The portlets displayed are limited to those for which the user has read permissions.

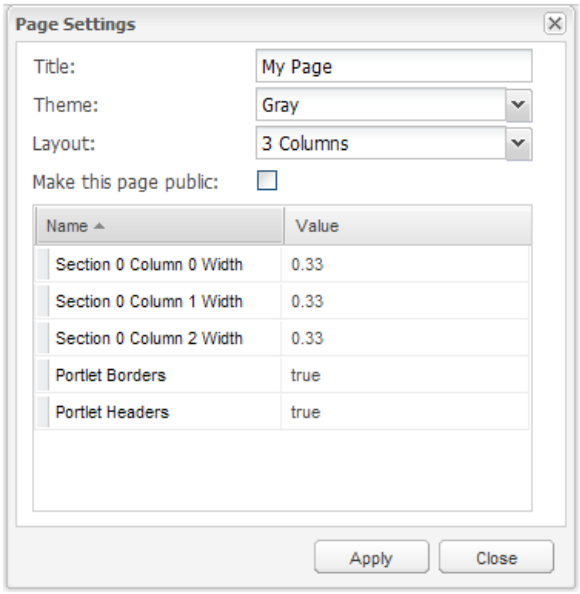


Note The Reserved Portlets are more easily added by just clicking on their buttons from the toolbar.

The number of portlets that can be added to a page is set on the Common Settings subtab (**Portal Settings > General > Common Settings**)—the "Maximum Number of Portlets on a Tab" and "Maximum Number of Grid Portlets on a Tab" settings (see the "Common Settings" section on page 4-31). When an attempt is made to add a portlet to the page that has already reached one or more of those limits, an error message appears to the user.

Page Settings

The Page Settings tab allows portal users to modify the appearance of the current portal page. The initial settings displayed are inherited from the page as designed in Portal Designer. You may adjust individual settings then click **Apply** to see the effects of your adjustments. Click **Close** when you are done.



Field	Description
Title	The title of the page, displayed on the tabs at the top of the portal page.
Theme	The color scheme and styles to be used to display the portal page contents. The drop-down box is enabled only for users who have the Service Portal "Manage Portal Page Theme" capability.
Layout	The number of sections and columns to be used to display the portlets that comprise the current page.
Make this page public	Making a portal page public makes it visible to other users. The check box is enabled only for users who have the Service Portal "Make Portal Pages Public" capability.
Column <n>Width	For each column in the Layout, the user can specify the percentage of the browser width that the column should take up. The percentages should not exceed 100 percent.
Portlet Borders	True if each portlet should have a border around it; false otherwise.
Portlet Headers	True if the grid column headers should be displayed; false otherwise.

Save Page

Click **Save Page** to save your edits to the page. The changes are seen by all users who subscribe to the page.

Search, Orders, and Approvals

Clicking the **Search**, **Orders**, or **Approvals** button is a shortcut to add the Search, Order Status, and Approvals portlet, respectively, to the page, rather than using the Add Content button (see the "Add Content" section on page 4-49). You must click **Save Page** to save the portlets on the page.

See the "Reserved Portlets" section on page 4-12 for information on these Reserved portlets.

In a Reserved portlet, click the button to edit settings for that portlet.

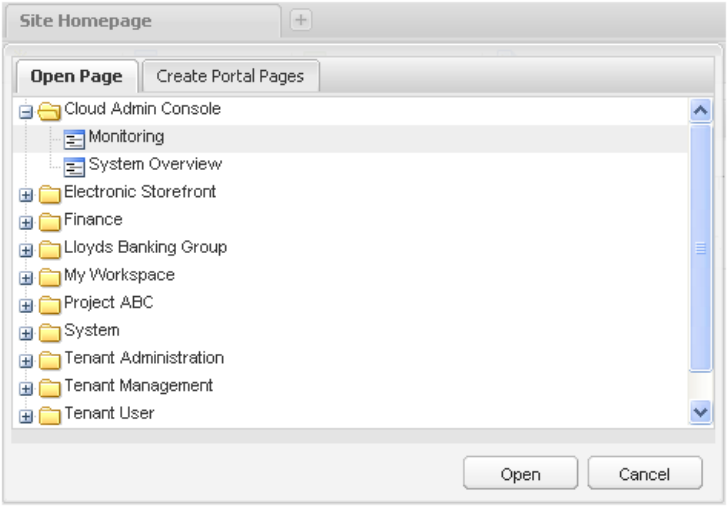
In a Reserved portlet, click the button to remove the portlet from the page.

Adding/Creating a Portal Page

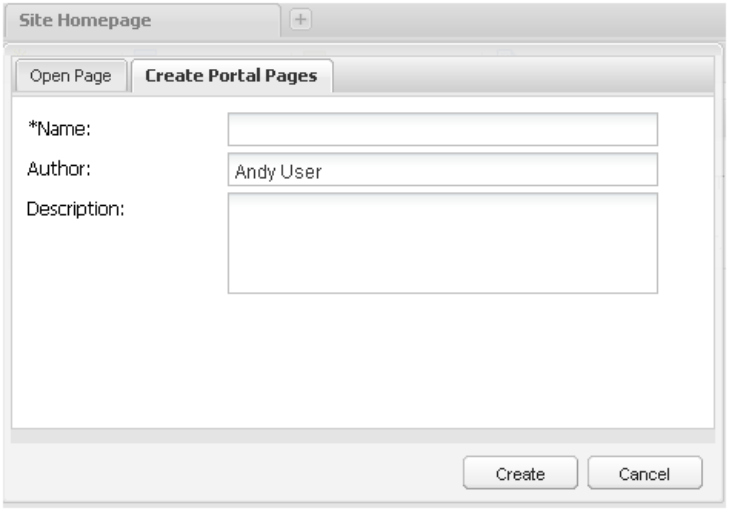
To the right of the last tab, clicking the button opens a popup window to allow users to add an existing portal page or to create a new page. The new page is added as the last tab in the module by default. It can be moved to a different tab position by mouse drag-and-drop actions as long as it is located after the homepages for the currently chosen module.

The maximum number of tabs allowed in each module is governed by the portal common settings. When the user is about to exceed the page limit, an error message appears.

The Open Page subtab shows a list of portal pages for which the user has read permissions. To add an existing page, highlight the page and at the bottom of the popup window, click **Open**.



The Create Portal Pages subtab is enabled for users who have the Service Portal "Manage Portal Pages" capability and write permission to the corresponding portal page group for the currently chosen module.



New portal pages are not marked as public by default. The number of such private pages allowed per user is also controlled by the portal common settings, except for those users who have the Service Portal "Override Private Portal Page Limit".

Importing and Exporting Portal Content

Overview

Portal content and portlet definitions are developed in a Service Portal instance, and the metadata underlying these objects is stored in the Content Management repository. You may want to backup an object definition to a source code control system or other file-based storage. You may want to develop content in a Development system, then transfer the content to a Test/QA system for testing or validation, and then to a Production system for everyday use by end users.

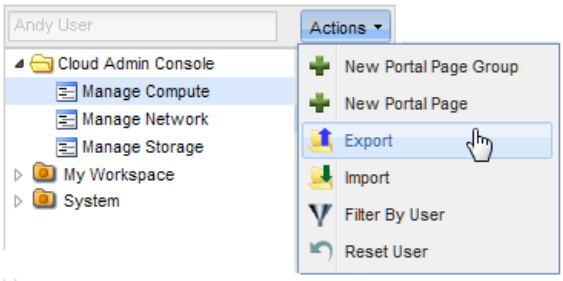
To do this, you need to use the Import/Export facilities provided by Portal Designer.

Portal Designer includes the following import/export options:

- Export specified content to the local file system.
- Import a previously exported file containing Portal Designer content into a new Service Portal instance.

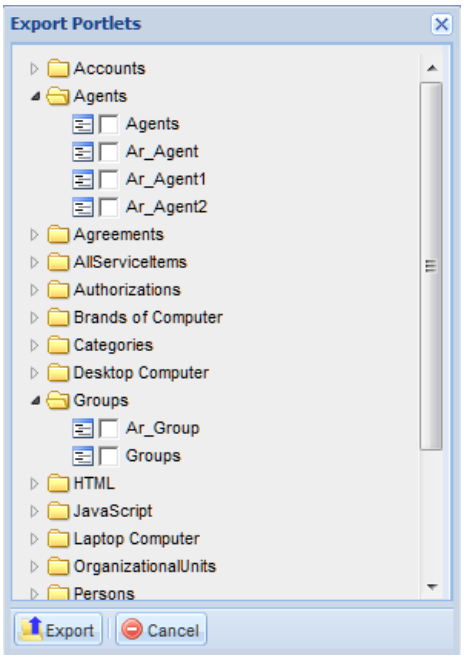
Exporting Portal Content

Use the **Export** action to export portal definitions to the local file system. The action is available for portal pages as well as non-JSR portlets.



Export Portlets

Portlets can be exported in bulk by checking the check boxes in front of the portlets shown in the Export Portlets popup window.



The export file is an XML file in an industry-standard CIM (Common Information Model) compatible format, version 2.3.1. CIM is based on an object-oriented model and uses terminology adapted from Unified Modeling Language (UML).

What is included:

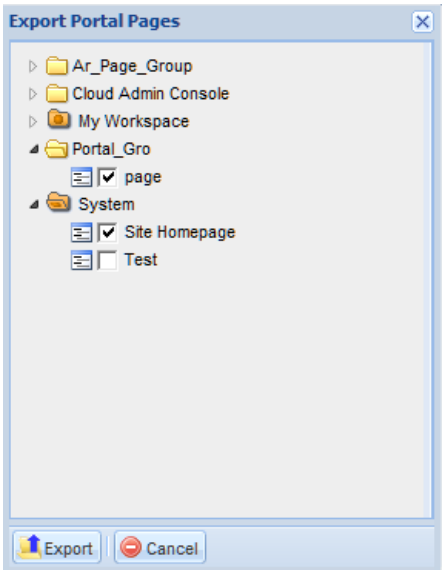
- Portlet general information, view and filter definitions, including HTML and JavaScript code
- Associated custom content definition and data
- Associated keywords
- Associated authentication settings

What is **not** included:

- All object permissions
- Definition and data for other content types (core entities, service items, standards)

Export Portal Pages

Portal page export works in a similar way and includes all associated definitions as part of the export XML file.



What is included:

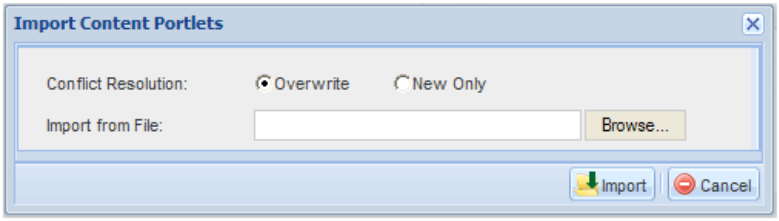
- Portal page general information, settings and content definitions.
- Associated portal page groups
- Associated portlet definitions
- Associated custom content definition and data
- Associated keywords
- Associated authentication settings

What is **not** included:

- All object permissions
- Definition and data for other content types (core entities, service items, standards)

Importing Portal Content

Use the **Import** action to import portal objects into the same environment in which they are exported, or another environment which does not contain those objects.

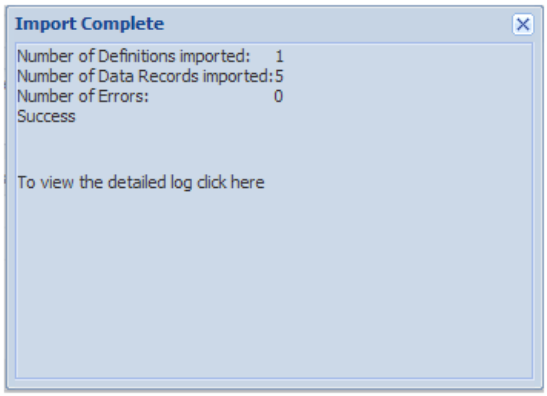


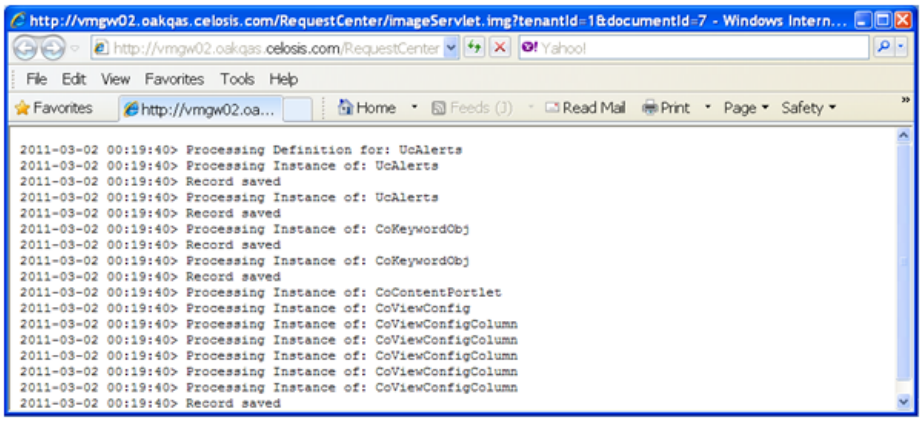
Import accepts only XML files created using the Export Portlets and Export Portal Pages features.

Two modes of conflict resolution are available:

- Overwrite: The import replaces the existing definitions of the objects with the same names in the environment with the definition contained in the XML file.
- New Only: The import fails if the portal object already exists.

When the import is complete, a summary page appears. A detailed log is also available to show the names of the portal objects created/updated.





The ability to create portal objects through the import utility is still controlled by the corresponding capabilities and permissions for such actions. The user who performs the file import should be granted appropriate roles in order to successfully execute the import.

An import may fail due to insufficient permissions. For example:

- The user does not have “Manage Portlets” capability when importing a portal page that contains new portlets. In this case, the portlets will not be created, and the import of the portal page containing the new portlets will also fail.
- The user does not have “Manage Custom Content” capability when importing a portlet that is based on new custom content. In this case, the custom content table will not be created, and import of the portlet making use of the custom content will also fail.
- The user does not have write permission to the portal page group that is specified for a new portal page. In this case, the portal page will not be created.

Portal Access Control

Overview

Site administrators can use the Organization Designer module to grant users access to the Portal Manager modules.

Roles for the Portal Designers

Roles and capabilities, managed with the Organization Designer module, allow site administrators to control access to functionality provided by Portal Designer.

Standard roles relating to Portal Designer and the capabilities included in each are summarized in the table below.

Capability	Description	Role		
		Portal Administrator and Designer	Distributed Portal Designer	Portal Content Provider
Access Portal Settings	Allows the user to view portal settings such as common settings, OU settings, keywords and authentication settings.	X	X	
Access Entity Reference	Allows the user to view the definition of core entities, service items and standards that are available as data source for portlets.	X	X	X
Manage Common Settings	Allows the user to view/modify the site-wide settings for the portal.	X		
Manage Org Unit Settings	Allows the user to view/modify the portal settings of organizational units.	X	X	
Manage Portal Keywords	Allows the user to add/modify/delete keywords and their associations with portlets.	X		
Manage Authentication Settings	Allows the user to define authentication method and connection parameters for use in portlets with external sites, and to associate the site settings with HTML portlets.	X		
Access/Manage Portlets	Allows the user to create portlets.	X	X	X

Capability	Description	Role		
		Portal Administrator and Designer	Distributed Portal Designer	Portal Content Provider
Access/Manage JSR Portlets	Allows the user to edit JSR portlets for which he has write permissions.	X	X	X
Manage Portal Page Groups	Allows the user to create portal page groups.	X	X	
Access/Manage Portal Pages	Allows the user to create portal pages.	X	X	
Access/Manage Custom Content	Allows the user to create custom content groups and custom content definitions.	X	X	X
Search All Portal Pages	Allows the user to search for portal pages created by other users.	X		
Import/Export Portal Definitions	Allows the user to import/export portal pages and portlet definitions.	X	X	

To use the Distributed Portal Designer role effectively, create child roles to it, and grant each child role read/write permissions to appropriate portal and organization objects.

Roles for the Portal End Users

Standard roles support end-users accessing the portal front-end:

Role	Description
Portal Basic User	Enables users to access the portal front-end and view portal pages defined by the portal administrators.
Portal Advanced User	Enables users to access the portal front-end and manage the content and presentation of their portal pages.
Portal Professional User	Enables users to manage portal pages and make them available to other users. This user can initiate and track service requests, authorizations, and service items on behalf of others in their business units and access those transactions in portlets.

Users of these roles require read permissions to particular portal pages and portlets in order to put the pages on their portal and view the portlets.

Users may need to be granted additional capabilities and permissions if they need to access other modules through hyperlinks on the portlets and to see the content in the portlets.

CHAPTER 5

Catalog Deployer

- Overview, page 5-1
- Catalog Deployer and Configuration Management, page 5-4
- Configuring Catalog Deployer, page 5-9
- Running Catalog Deployer, page 5-19
- Sample Deployment Scenarios, page 5-39
- Branded Content Libraries, page 5-46

Overview

Overview

Catalog Deployer is a content deployment and configuration management tool that can be used by service designers, catalog publishers, and organization building resources to migrate application entities between development, test, and production sites.

Catalog Deployer provides customers with a change management process and history, allowing for reliable control over Request Center content changes, and changes to the organizational entities used by all modules of Service Portal.

Catalog Deployer also supports the deployment of preconfigured services packaged within a branded content library by Cisco. Such services can be used as-is or as the templates for customizing the service definition to the organization's requirements, significantly reducing the time and effort required to implement actionable service catalogs.

Catalog Deployer Operation

Catalog Deployer offers two methods of content deployment. Source sites can use Catalog Deployer to “extract” or assemble a package of service definitions and entities for transmission to, and deployment on, a target site. In this case, all operations can be performed within the Catalog Deployer module, and there is no need for external programs. Alternatively, or in addition, a package can be produced for export. Exported files are imported via Catalog Deployer into the target site’s instance of Service Portal. Since these are XML files, in text format, they can also be stored in a configuration management or source code control system. Branded content libraries are delivered in the form of an export file, so standard Catalog Deployer facilities may be used to import and deploy services available in the library.

Like all other Service Portal modules, Catalog Deployer can be permissioned for use. Users may have abilities to view deployment history; to create and assemble a package for deployment; to import or deploy a package to a particular site; or the combination of these capabilities that best fit the users' responsibilities. All such capabilities may be assigned via standard roles or custom roles.

Catalog Deployer Features and Functionality

Key features include:

- Simplicity of use via the user interface. Packages for deployment are created, and can be populated, transmitted to a target site, or exported to the file system.
- Two methods of content transfer: XML package transmission across sites or file-based transmission via export/import functionality.
- Service deployment (inclusive of related entities) on one or more target sites.
- Organizational deployment (OUs, groups, queues, roles, people, functional positions) on target sites.
- Email template and Service Link agent deployment on the target sites.
- Change management support (optional segregation of duties between content developers/managers and catalog publishers).
- Hot deployment—the ability to deploy content while the application is available to users.
- Online options to view extraction and deployment history.
- Support of site protection levels and entity homes.
- Ability to preview a service definition before it is deployed to a target site from a branded content library.

Catalog Deployer Usage

Catalog Deployer may be used at the beginning of a service catalog effort to provide template content which can form the basis of a customized service catalog. In addition, Catalog Deployer may be used in the Build and Maintenance phases of a development effort, to promote tested content from development to other environments and to synchronize multiple environments.

Configuration Management

The following sections of this chapter discuss capabilities relevant to using Catalog Deployer for release and configuration management:

- Catalog Deployer and Configuration Management, page 5-4
- Configuring Catalog Deployer, page 5-9
- Running Catalog Deployer, page 5-19
- Sample Deployment Scenarios, page 5-39

Service Catalog and Portfolio Development

The following sections of this chapter discuss capabilities relevant to using Catalog Deployer to install Cisco content libraries to provide the basis for a service catalog and service portfolio:

- Running Catalog Deployer, page 5-19 (only the sections on importing and deploying content.)
- Branded Content Libraries, page 5-46

Terminology

Key terms used in this chapter are defined in Table 5-1 below.

Table 5-1 Key Terms

Term	Definition
Assemble a Package	Add content to a package. Assembling a package extracts the current definitions of the objects which have been included in the package from the repository and writes them to the package. Any subsequent changes to these objects will not be reflected in the assembled package.
Associated Entity	An entity that is related to the primary entity and required in the target site for a deployment to be successful. For example, a service definition (primary entity) may refer to several organizations, service teams to whom tasks are assigned (the organizational unit is the associated entity).
Catalog Deployer	The Service Portal module responsible for deployment of catalog content and directory information from one site to another.
Component Entity	Entities that are automatically deployed when a service is deployed. Service component entities include categories, presentation elements, dictionaries, dictionary groups, service groups, keywords and objectives.
Data Source	A data source, defined in the JDBC data source page on the application server console, which specifies connection information for all sites which become targets for Catalog Deployer direct site-to-site deployment.
Deployment Package	The primary object managed by Catalog Deployer. The deployment package contains the chosen entities that are to be deployed (such as service definitions or queues and groups) and the deployment activity history. A deployment package can be transmitted or exported/imported into another site for deployment of its contents into the target site. A deployment package does not contain entity content until it is “assembled,” at which point the included data is extracted from the source site.
Entity	One of the objects created and maintained within Service Portal software. Examples are: organization units, service definitions, queues, and email templates.
Entity Home	The page of the Administration module's settings where administrators specify which site is the site of record for each supported entity type.
Export a Package	Create an XML file containing the contents of a previously assembled package and write the file to the file system. Exporting a package from a source site to a target site is an alternative to transmitting the package.
Implementation	A group of one or more Service Portal sites, either directly or indirectly connected.
Import a Package	Create a package by importing a previously exported XML deployment package into a target site. The package is created with the status “Received for Deployment”.
Library Package	A package from Cisco that has been preassembled and is available as a file import. The package content consists of template services or associated entities that can be used to provide the basis for a service catalog. Unlike deployment packages, chosen contents of a library can be deployed.
Primary Entity	An entity that can be chosen for inclusion in a Catalog Deployer package.

Table 5-1 Key Terms

Term	Definition
Site	A collection of one or many computer systems (single computer or cluster) that share a database and an http address, and can be categorized by function. For instance: a development site, a testing site, and a production site.
Source Site	The site in which a package is created. This is the site which transmits or exports a package for deployment on another site.
Target Site	The site in which a package is deployed. This is the site which receives a package via transmission or import.
Transmit	Send a package from one site to another via a database connection. Once a job has been transmitted, it cannot be edited (reassembled and have its content changed) on the source site. The target site receives the transmission via Catalog Deployer.

Additional Resources

Documentation	Description
Catalog Deployer Online Help	Provides an overview of the features and functionality available through Catalog Deployer and detailed steps for performing deployment tasks.
Cisco Service Portal Installation Guide	Describes the requirements and prerequisites necessary for installing Service Portal and all associated modules, including Catalog Deployer.

Catalog Deployer and Configuration Management

Overview

This section describes how to design a Service Portal implementation and the associated processes that support industry-standard configuration (or change) management practices.

Typical methodologies are reviewed and compared with those available for Service Portal applications, including Catalog Deployer configuration management. The section discusses different approaches to configuration management, as they are matched to the stage of deployment—from initial development, through testing, deployment, and maintenance.

Typical Configuration Management

Catalog Deployer implements best practices embodied by IT industry standard configuration management methodologies and technologies. Therefore, it is useful to review these practices.

- Changes to software configuration items (that is, the individual software modules or components that comprise the IT application) are made in a development environment.
- In the same (development) environment, the changed software typically undergoes preliminary testing (unit testing).

- Once the software has passed unit tests, a copy of the “source” for the software is checked into a source code control system and labeled as the release candidate. “Source” may consist of several types of artifacts, including code written and maintained in a text editor, or, increasingly common, specifications stored in XML files or within a metadata repository that is part of an integrated development environment.
- The saved source is deployed to a tightly controlled test environment, where it undergoes rigorous testing. The test environment may be reinitialized before each set of tests, to ensure that results in different runs are comparable.
- The testers are responsible for finding problems, not diagnosing or fixing them. All problems are reported to the development team, which uses its development environment to fix the problems, retest the code, and save a copy of the revised source.
- The fix, extract, deploy, and test steps are repeated until the testing team certifies that the software meets all test criteria—these may be performance measures or functional requirements or a combination.
- The same source that was deployed to the testing environment (and tested!) is deployed to the production environment.

Configuration Management

For Service Portal applications, such as Request Center, the software that is developed is typically a service definition with related elements such as categories, groups, tasks, and checklists. The typical configuration management scenario, and the role that Catalog Deployer plays in this, looks like:

- A new or enhanced service definition is developed and unit tested in a development environment.
- Catalog Deployer is used to extract the new or updated service definition from the development environment. The resultant deployment package may be exported and placed under source code control if desired.
- Any other code resources (such as JavaScript libraries) that do not reside in the Service Portal database may also be placed under source code control.
- Catalog Deployer is used to deploy the service definition in a test or quality assurance (QA) environment.
- The service is tested. If problems are encountered, the previous steps are repeated—code is fixed in development, extracted, and redeployed to the test environment—until the service is certified as having met the stated requirements.
- Once the service is accepted in the Test environment, it is deployed to the production environment, using the same procedure that originally deployed the code to the test environment.

This scenario adheres to industry standards in that the development environment is the only place changes are made to the service definition, and an automated process is used to deploy a controlled set of source code to test and production environments.

However, this scenario is incomplete. In most implementations, people and business units (a type of organization) are dynamically added to the Production environment, as people log in to Request Center or Demand Center for the first time, have a service ordered on their behalf, or are assigned to perform a review or authorization. Therefore, Catalog Deployer must also be used to migrate these entities from the production site back to development, so they are available for use in service definitions and other Request Center configuration items, such as authorizations, that are still under development. Further, the service definition may refer to related entities such as email templates, groups, queues, service team-organizations, and roles. If any of these do not exist in the target environment, they must be deployed to that environment before the service's deployment package can successfully be deployed.

Catalog Deployer can be used by Request Center customers, as illustrated in the scenario above, to deploy content such as service definitions, as well as any associated services.

Catalog Deployer Architecture

Catalog Deployer provides database-neutral data-transfer infrastructure and interfaces, organized around the logical entities found in the application. Examples of the logical entities are Service Definitions, Data Dictionaries, People and Organizational Units. A complete listing is available later in this section. There is however limited support for custom content across different databases and application environments. See the "What Catalog Deployer Does Not Do" section on page 5-7 for the list of entities not supported by Catalog Deployer.

An understanding of Implementation, Site, and Entity Home, described in Table 5-1 on page 5-3, is critical to the operation of Catalog Deployer.

In particular, Logical Entity Home Sites are an integral part of configuration management with Catalog Deployer. Though optional with Catalog Deployer, creating a system of Home sites for logical entities enables management of both the referential integrity of logical entities across sites, and the integrity of configuration information across sites.

The idea behind Logical Entity Home sites is that certain sites will have a more authoritative version of the data for a logical entity type than others. For example, for customers using their LDAP directories to authenticate and gather information about users, data on People and Organizational Units are most accurate on Production. Therefore, the Production site would be the Home site, the site of record, for the People and Organizational Units. If People can be created, or Organizational Units edited, on sites other than Production, deploying this data to other systems will get them out of sync with the system of record, and the quality of data across the implementation will degrade.

With an eye to these problems, the application framework is designed to allow protection levels to be assigned to logical entities to keep users from modifying these entities on sites other than the entities' Home sites. The site administrator can choose how much protection to provide for logical entities by choosing one of four settings, as described in Configuring Catalog Deployer, page 5-9.

Catalog Deployer Capabilities

Entity Permissions

Catalog Deployer deploys permissions for entities as part of the entity. When permissions are removed from the entity in its Home site, the application does not leave behind deletion stubs (or a transaction log) that Catalog Deployer may use to replicate this removal. For example, when the permission to order a service is removed from an Organizational Unit on a Service Definition, Service Portal does not retain this fact, it simply removes the permission data.

When you deploy a service whose permissions have been changed, all associations between the service definition and its permissions are dropped in the target site and recreated according to the permissions in effect in the source site. Any deleted permissions are reflected. However, if the permission was granted to a custom role or group, and the role or group was deleted from the source site, the role or group will still exist in the target site. Catalog Deployer cannot propagate entity deletions.

What Catalog Deployer Does Not Do

Catalog Deployer migrates logical entities which are stored in the transactional database between two sites which are implemented using the same version of Service Portal. The entities are extracted to a text stream, formatted in .xml, which is part of a deployment package which also includes the specifications (options) used to create the package. That package may be extracted from the source database, to serve as a backup mechanism as well as the source for deploying the entity definitions into the target database of another site. The entity definitions are deployed unchanged into the target environment.

Catalog Deployer has no effect on any components of Service Portal other than those logical entities. Changes to some of these components are not part of the configuration management scenario handled by Service Link—the synchronization of customer-designed and configured configuration items across an implementation—so need not be considered further here. For example:

- Software components must be installed and configured via the Service Portal Installer.
- The contents of the data mart and reporting tables must be created via Service Portal Installer and populated via Extract-Transform-Load (ETL) processes.
- The schema is upgraded as part of the Service Portal installation.
- Any customized Service Portal components, or additional components, need to be installed on all servers via the "Customizations" option in the Service Portal Installer and the procedures documented in the *Cisco Service Portal Configuration Guide*. Such customizations typically include support for custom mappings used in directory integrations, and any APIs provided by the Advanced Services organization.

However, additional configuration items may have to be deployed in conjunction with changes to the logical entities which are handled by Catalog Deployer. These are summarized in the table below and discussed in more detail in conjunction with the logical entity affected.

Configuration Item	Additional Artifacts to be Controlled and Migrated
External Dictionary	DML and DDL scripts run in the database
Datasource	Datasource specification to reference an external dictionary, a SQL-based option list or a table referenced in a data retrieval rule
Data Retrieval Rules	SQL Statement directly entered into the rules (may not be compatible across different database types)
ISF Script Library	Library (JavaScript) text file deployed on the application server
Custom Adapter	Deployment file produced by the Service Link Adapter Development Kit (ADK)

Entities Supported by Catalog Deployer

The table below lists all logical entities supported by Catalog Deployer and their associated application module.

Module	Supported Entities
Service Designer	Service definitions (Offer, Form, Form Sections, Plan, Authorizations, Permissions) Component entities automatically deployed with service definitions: • Service Groups • Dictionaries and Dictionary Groups • Active Form Components and Component Groups • Keywords, Categories, and Presentation Elements • Script Functions and Libraries Entities referenced by service definitions: • Email Templates • Organization Designer entities • Service Link agents and transformations
Service Item Manager	Component entities automatically deployed with service definitions: • Service Items (if referenced by a Service Item-Based Dictionary or by a table-based data retrieval rule) • Standards (if referenced by a table-based data retrieval rule)
Service Link	• Agents • Transformations associated with the chosen agent are also deployed.
Organization Designer	• Queues • Organizational Units • Groups • Roles • Functional Positions • People
Administration	Email Templates

Catalog Deployer copies the entities listed above from the source site database to the target site database. Catalog Deployer does not move or copy information stored in the file system. Catalog Deployer copies the definition of libraries associated with JavaScript functions.

Note Catalog Deployer **does not support** deploying images (presentation elements) in .bmp format. Service Designer now prevents such images from being specified. Any legacy images should be converted to an alternate format, such as .jpg or .gif, optimized for presentation on the web.

Configuring Catalog Deployer

Overview

Configuring Catalog Deployer involves:

- Meeting the prerequisites for installing Service Portal and for configuring client workstations to support Catalog Deployer.
- Installing a version of Service Portal that includes Catalog Deployer.
- Configuring implementations and sites within the development and production Service Portal instances.
- Configuring application server JDBC data sources on source and target sites.
- Using the Administration and Organization Designer modules to ensure that personnel have access to Catalog Deployer capabilities appropriate to their functions in the implementation.

Prerequisites

See the *Cisco Service Portal Installation Guide* for Service Portal prerequisite information.

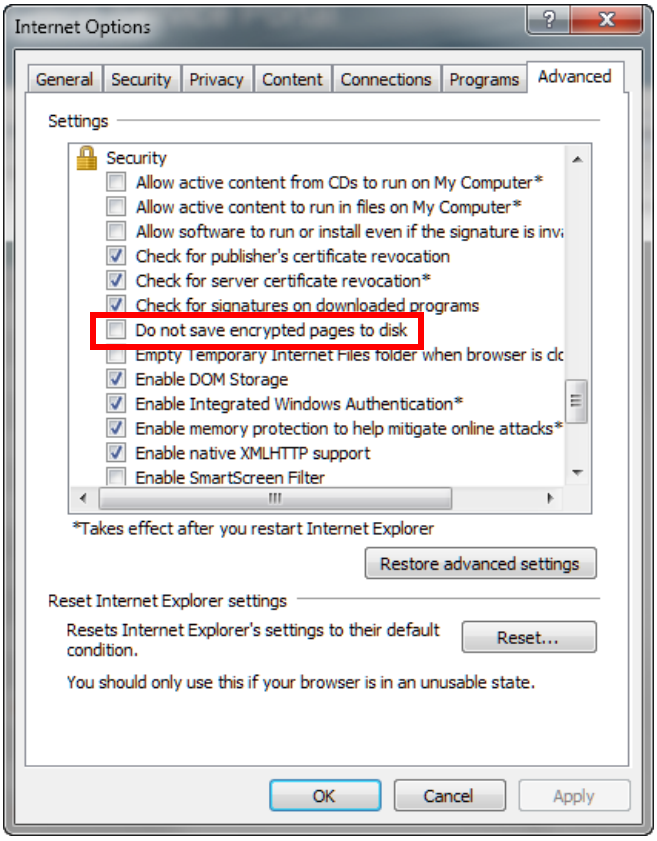
Installing Catalog Deployer

Catalog Deployer is automatically installed as part of all Service Portal sites.

All sites should have the same version of Service Portal installed, including Service Packs.

Configuring Client Workstations

The deployment package produced by Catalog Deployer contains a compressed XML representation of the definitions of the included entities. Catalog Deployer can use file-based transmission of packages, in which package contents are transferred from one site to another via the export of the package and its subsequent import into the target site. To support this mode of transmission, the user's browser must be configured to allow encrypted pages to be saved to disk, as shown below in the Security section of the Advanced Internet Options for Internet Explorer.



This configuration is not required if all deployments are via direct site-to-site transmission.

Overview of Configuring Implementations and Sites

Before using Catalog Deployer, you need to use the development and production instances to configure your implementations. An implementation is the collection of sites between which Catalog Deployer migrates Request Center service definitions and other Service Portal entities.

Configuring implementations and sites consists of the steps summarized below and described in detail in the following section.

Name the Implementation

Assign a name to the implementation. This name typically reflects the company or the project. This name is for documentation only, and is not used by Catalog Deployer.

Name the Sites within the Implementation

Within an implementation, each site must have a unique name. Service Portal uses this name to identify the site for purposes of assigning protection levels for the Service Designer and Organization Designer pages that allow users to change (add, modify, and delete) entities.

Two sites are typically required: Development (DEV) and Production (PROD). Additional sites, for example, STAGE, TEST, or QA, may be used if configuration management and migration plans call for their use.

Specify the Home Site for Each Logical Entity

Specify the (single) Home site for each logical entity. A few rules of thumb are useful in making a decision about the Home sites for logical entities:

- A logical entity should typically be Home at one and only one site. This will allow you to better manage changes to the entity instances.
- Any logical entity involved in the definition of a service should clearly be home in the development instance.
- Any logical entity created through production use of the system should be Home at the Production site. If Single Sign-On (SSO) or Directory Integration which includes an Import Person event is enabled, logical entities home in Production will include people (and thus queues, as they are in the same table). If the automated user creation facility creates Organizational Units, then Organizational Units should also be Home at the production site.

Note Logical entities are almost never Home at a test or stage site. This is because such sites are typically rebuilt with production data and newly developed code to be tested.

Create a Data Source for Each Site

For Catalog Deployer to deploy content to a site, a JDBC data source must be configured for that site in the application server running the Service Portal application. For example, in order to deploy services developed on the development site to test and production, the development site must include data sources for both the production and test sites; to deploy organizational entities from the production site to development, the production instance must include a data source for the development instance.

For clustered sites, the data source must be accessible to each node that comprises the site.

Repeat the Process for Each Site

The data that specifies implementations, sites, and logical entities is, in turn, stored within logical entities.

These specifications must exist in all Service Portal instances which comprise the implementation.

You also need to create appropriate data sources at all sites.

Instructions for Configuring Implementations and Sites

You must configure your implementation environment settings in the Administration module to begin using Catalog Deployer. The values you set in Administration allow source sites to recognize target sites.

These steps must be performed after you have defined your data sources in the JDBC data source page on the application server console.

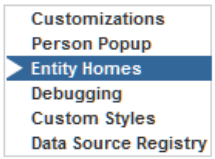
To configure an implementation:

Step 1 Log in to the development instance as a user with Administration privileges.

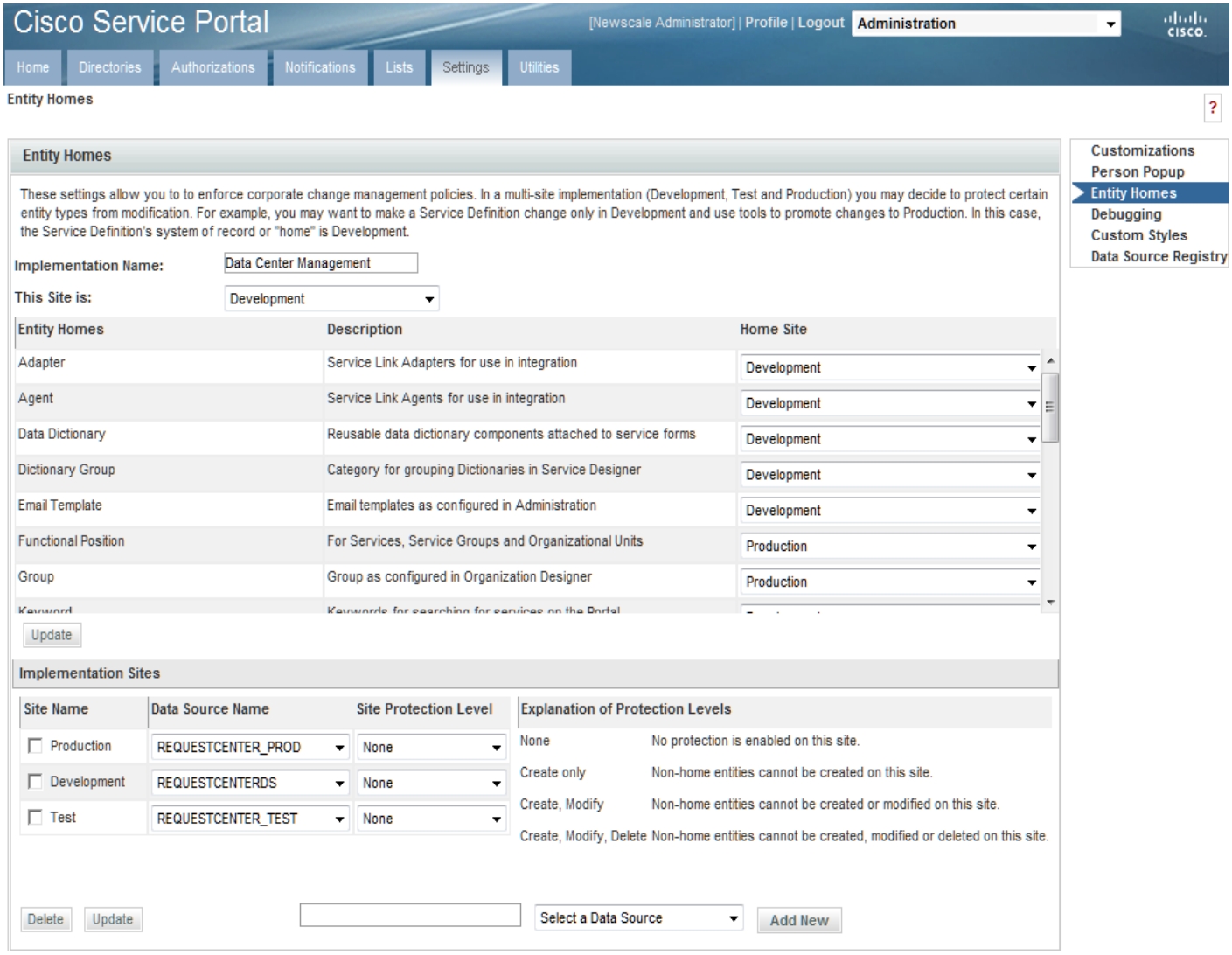
Step 2 From the module drop-down menu, choose **Administration**, as shown below.

Step 3 Click the **Settings** tab.

Use the menu on the right-hand side, as shown below, to choose **Entity Homes**.



The Logical Entity Home Specification page appears, as shown below.



These settings influence the behavior of Catalog Deployer as well as the Service Designer and Organization Designer modules. If they are set improperly, incorrect data could be written to Service Portal application sites, including production. Only systems administrators should change these settings.

Step 4 In the "Implementation Sites" section, in the text field at the bottom of the page, enter a site name; for example, "Development".

Step 5 From the "Select a Data Source" drop-down menu, choose a data source.

Step 6 Click **Add New**.

The site name is added to the list of site names.

Step 7 Add the name for your production site as another site.

Step 8 If the implementation includes other sites which need to be refreshed via Catalog Deployer, such as QA or Test, add these as well.

Step 9 At the top of the page, in the **Implementation Name** field, enter an implementation name; for example, "My Cloud" or "Data Center Management".

Step 10 From the "This Site is" drop-down menu, choose the development site to identify the current site.

Step 11 Click **Update**.

Step 12 Review the **Home Site** assignments for the entities, changing any that do not fit your requirements. Click **Update** when finished.

Step 13 Assign the appropriate **Site Protection Level** to each site you have defined. Click **Update** when finished. These protection levels alter the behavior of the Service Designer, Organization Designer, and Administration pages through which the corresponding logical entity is maintained.

Protection Level	Effect on User Interface at Nonhome Sites
None	Nothing: all UI elements for creating and modifying entities remain available.
Create only	Controls for creating new logical entities are disabled.
Create, Modify	Controls for creating and updating logical entities are disabled.
Create, Modify, Delete	All controls for creating, editing and deleting logical entities are disabled.

The same site names, entity home settings, and protection levels should be specified at all sites in an implementation. This will prevent users from inadvertently creating or modifying an entity at the wrong site, where it could be overwritten by the next deployment of the "same" entity, developed and maintained at a different site.

Guidelines for Assigning Site Protection Levels

Development. A protection level of **"**none**"** should only be used on a development site, in the initial phases of an implementation. This allows all entities to be created, modified, or deleted within the site by users who have appropriate roles.

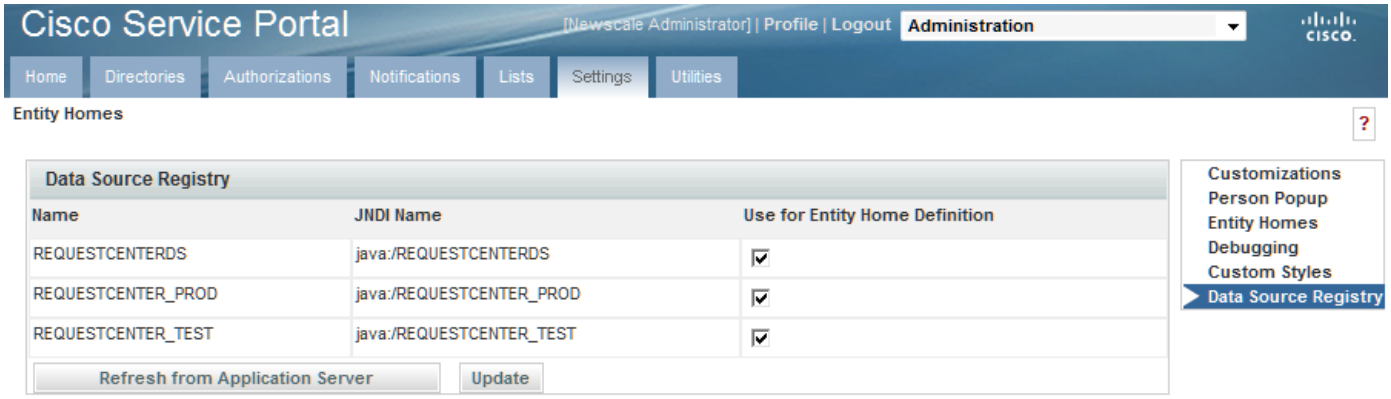
Test. A protection level of **"**Create, Modify, Delete**"** should typically be applied to any test, staging, or QA sites. These sites would typically be refreshed by copying a complete database from production (for example, to support performance or volume testing) or by deploying services from development for functional testing prior to promotion to production.

Production. A protection level of **"**Create, Modify, Delete**"** should typically be applied to the production site. A protection level of "Create only" would allow minor modifications to be applied to protected entities, such as changing an entity name.

Configuring Data Sources for Sites

In order for Catalog Deployer to transmit a package to another site, the JDBC data source that corresponds to the target site must be configured in the same application server as the source site. The procedures for configuring the JDBC data source are exactly the same as those for configuring the RequestCenter data source, as outlined in the *Cisco Service Portal Installation Guide*.

The same JNDI name prefix should be used to allow Service Portal to discover the data source. The list of discovered data sources appears in Administration > Settings > Data Source Registry, as shown below.

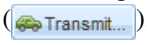


Certain data sources are used for reporting purposes and are not meant to be seen by Service Designer and Catalog Deployer users. To limit the data sources to just the ones these users need, check the "Use for Entity Home Definition" check boxes for those data sources and then click **Update**.

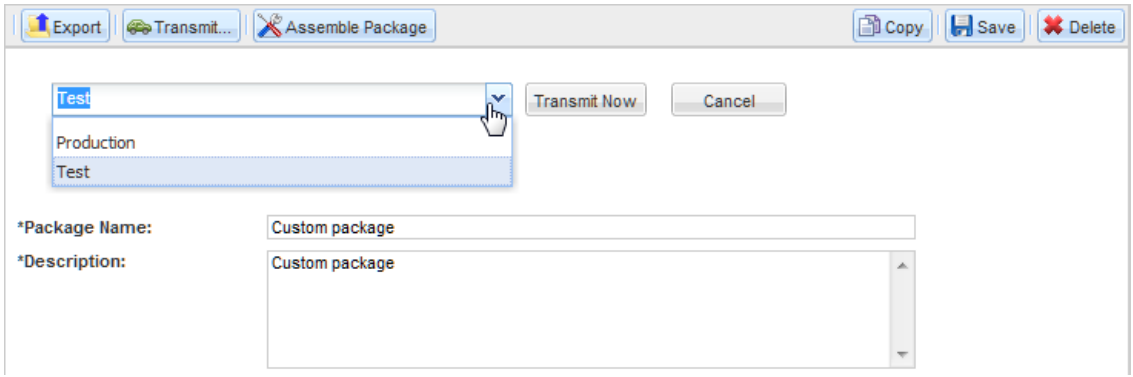
Configure All Sites

The steps above guide you through configuring one site—you will typically start with the development site. The development site is the source site for service definitions that are migrated to other target sites that become visible to the source site. Any site can be configured as a source or a target site, or potentially both, depending on the configuration of your overall implementation.

For example, you might configure your development site to see your test and production sites, and configure the production site to see the development and test sites. In this scenario you would deploy entities from development to test, and then from development to production, assuming that testing was successful. You would migrate organizational information from production to both test and development sites.

Repeat the configuration of potential target sites on all source sites, including the definition of required data sources. To verify that configuration is successful, log in at each site, navigate to Catalog Deployer, and in the Action pane, attempt transmitting a package to one of the target sites you just configured by clicking **Transmit** (

).You should see your site in the drop-down menu as one of the target sites you can choose.



Who Uses Configuration Management Tools?

Service Portal expects Service Designers, Configuration Managers, Catalog Publishers, and Organization Builders to use Configuration Management tools. These key roles are defined below.

Role	Definition
Catalog Publisher	Creates and maintains service catalogs and is responsible for deploying catalog content to the runtime application, configuring the look and feel and structure of catalogs, and for updating the deployed catalog content on an ongoing basis as the service definitions and delivery plans change.
Service Designer	Designs service definitions at a customer site. Service designers have significant subject matter knowledge of the services provided to the customers of the customer IT team, and are proficient in the usage of Service Designer and Organization Designer. Services designers should be moderately technical, but are typically business analysts rather than engineers. Service designers may frequently need to create and modify service definitions within their development site. Some are allowed to use the Catalog Deployer for publishing their work to a staging or production site.
Organization Builder	Designs Organization Designer organizational entities at a customer site. Organization builders have significant subject matter knowledge of the organization's needs with regard to configuring the organizational units, groups, and roles necessary to successfully deploy and use Service Portal. Organization builders should be proficient in the use of the Organization Designer module. Organization builders do not need to be highly technical, but should be an application administration IT resource.
Site Administrator	Performs application permission administration at a customer site. This person may be the first user of the system and is able to access all facets of the back end of the application, such as setting Global Configurations; managing lists such as languages and billing categories.
Change Manager	Approves change requests for the implementation at a customer site. This person is typically responsible for the stability of the production site and needs to understand all changes prior to their deployment to the production site. This role is optional, but is required in those customer sites that have established formal change control processes.

Application Roles and Capabilities

The table below describes the system-defined roles along with the default capabilities granted to each role, relevant to the Catalog Deployer module.

The Catalog Publisher and Licensed Content Publisher roles are unique to the Catalog Deployer module. The Catalog Publisher role enables users to manage deployments of the Service Catalog and other organization changes between applications sites, and also to publish content libraries provided by Cisco. The Licensed Content Publisher role includes the capability to "Package Branded Content Libraries" and is reserved for use by Cisco only.

	Capability					
Predefined Roles	**Manage Basic Service Deployments**	**Manage Advanced Service Deployments**	**Manage Custom Deployments**	**Import Deployments**	**Deploy Deployment Packages**	**Package Branded Content Libraries**
Catalog Designer & Administrator	X	X	X			
Organization Designer			X			
Site Administrator	X	X	X	X	X	X
Catalog Publisher	X	X	X	X	X	
Licensed Content Publisher	X	X	X	X	X	X

Catalog Deployer Performance Considerations

Catalog Deployer must be run when the source or target site is online and in use. However, it may be advisable to apply some restrictions to this usage.

Concurrent Usage of Catalog Deployer

Catalog Deployer consumes a significant amount of memory for assembling and deploying packages, as well as previewing services. To avoid any issues resulting from memory consumption, Catalog Deployer prevents more than five users from assembling, importing, or previewing a package at the same time. If a sixth user attempts to assemble, import, or preview a package, Catalog Deployer displays an alert to that effect, telling the user to try again later. While this may be an unlikely scenario in a Production environment, since the Production deployment will likely be handled by a single person, it may well occur in Development or Test environments if multiple service designers are packaging their respective content for deployment.

Browser Session Time-out

Package assembly or deployment is still considered interaction with the browser. Hence the session time-out for inactivity configured in the Administration module does not cause large package assembly or deployment to time out.

Package Size

Catalog Deployer only creates/updates associated entities a single time within a package if the entity is required for multiple primary entities. For example, if the “IT Software Configuration” dictionary is necessary for 20 services within a deployment package, Catalog Deployer only creates/updates the dictionary once on the target site. In grouping services with shared components in the same package, the size of the package and the number of redundant creations/updates Catalog Deployer must perform can be reduced.

Users can expect the performance of a large package (as defined by its size) to be slower. Users should also be aware that assembly or deployment of such packages may occasionally fail. If this occurs, the solution is to simply break the package up into a smaller set of primary entities; for example, creating two packages of 10 services each instead of one consisting of 20 services.

Hot Deployment

In principle, Catalog Deployer can deploy a complete package in one database transaction. In such a case, failure to deploy any one component (for example, a service definition in an advanced package cannot be deployed because a specified queue does not exist in the target) would result in the entire package contents being rolled back. The target site's repository would remain as it was before the deployment was started.

In practice, however, this all-or-nothing deployment may cause problems. Once an entity has been deployed, it is not available to online users until the complete package has been deployed and the database transaction committed. If, during this time frame, a user attempts to order a service that has been updated by the ongoing deployment, the user's session would hang, waiting for the service definition to be available. Eventually, the user would receive an error (worst case) or, after a delay, be able to order to service (best case). (This error may occur only when a user is initially ordering a service, not when task performers or authorizers are working with this service.)

One sure way to avoid this scenario is to not allow deployments while a production system is in use. This complies with the industry-accepted best practice of performing updates to a production system during a regularly scheduled maintenance window. However, waiting to deploy until a maintenance window roles around may not be possible. To minimize the probability of problems arising if a deployment must be run when Service Portal is operational, it is advisable to group services into smaller logical packages. The service designer can still take advantage of reduced package size for shared components by grouping related services into the same package.

Running Catalog Deployer

This section provides a general overview of Catalog Deployer functionality. See the *Catalog Deployer Online Help* for more detailed user steps.

Overview

The following types of deployment packages are available:

Basic Services: Basic Services deployment packages work in a similar way as importing service definitions in Service Designer. If any associated entities (for example, a queue or organizational unit) are not found on the target site, the deployment would simply skip the entities not found. Basic packages cannot include bundled services.

Note You do not need permissions to view or edit services in Service Designer to view and choose services for deployment in Catalog Deployer.

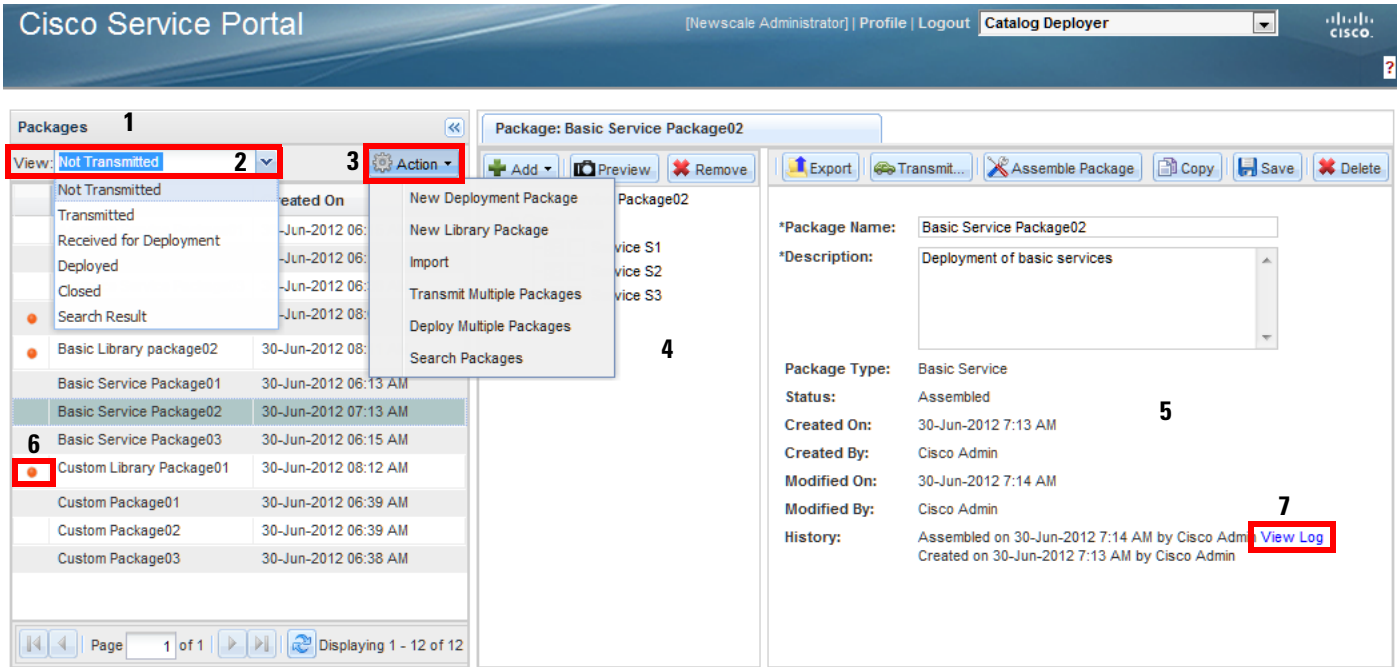
Advanced Services: Advanced Services deployment packages are for migrating service definitions, including bundled services. This package type provides the ability to control options for how to process the associated entities of a service definition during the deployment on the target site.

Custom: Custom deployment packages allow you to choose entities individually and control options for how to process associated entities during deployment. Custom deployment is typically used for organizational entities, such as people, OUs, and groups. Custom packages cannot include services.

Using Catalog Deployer

Catalog Deployer is a module of Service Portal. It will appear on the module drop-down menu for any user who has been granted a role that allows access to Catalog Deployer (as specified in the "Application Roles and Capabilities" section on page 16):

The Catalog Deployer Home Page provides access to all options:



1	View and Search pane	**4**	Content pane	**7**	View Log link
2	View drop-down menu button	**5**	Action pane		
3	Action drop-down menu button	**6**	Library package icon		

The home page consists of the following sections:

- The **View and Search** pane allows users to view all packages with a specified status using the View drop-down menu. The list of packages is initially sorted by date, with the most recently created package at the top of the list, but can be changed (see "Hiding/Adjusting the View and Search Pane" below). The Action drop-menu contains functions to create, import, and search for packages, as well as transmit and deploy multiple packages.
- The **Content** pane displays information about the package currently selected in the View and Search Pane. Content may be added, previewed, and removed with the buttons on this pane.
- The **Action** pane displays information about the package currently selected in the View and Search Pane and Catalog Deployer action buttons. Available actions depend on the status of the package. Only those actions that may be applied to the package currently selected are enabled.

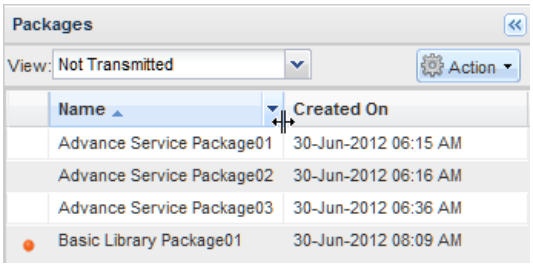
You can resize the panes by moving your mouse between two panes until the Column resize cursor appears ( ). Click and drag it to the desired position.

Hiding/Adjusting the View and Search Pane

As illustrated below, in the View and Search pane, you can resize, move, sort, and hide columns. In addition, you can hide the View and Search pane by clicking the hide button ( ). Click the show button ( ) to show the pane.

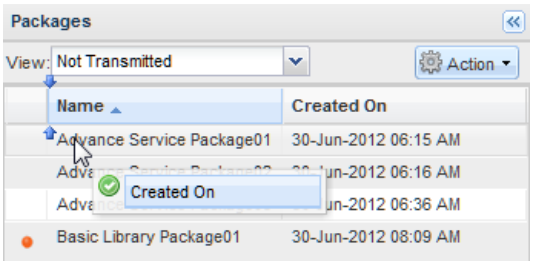
Resizing a Column

Move your mouse between two columns until the Column resize cursor appears ( ). Click and drag it to the desired position.



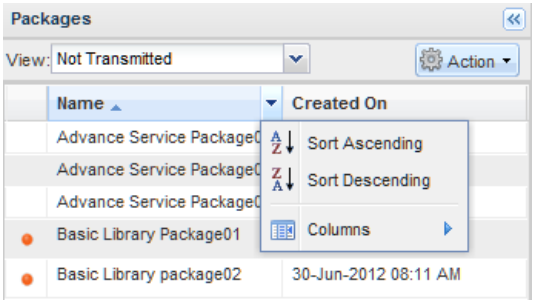
Moving a Column

Click a column with your mouse and drag it to the desired position. In the example below, the user is dragging the Created On column to the left of the Name column.



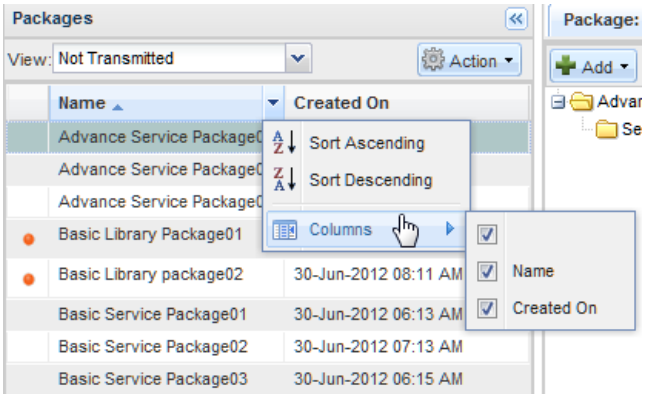
Sorting a Column

Move your mouse over a column until the Column sort button appears ( ). Click the Column sort button and then choose **Sort Ascending** or **Sort Descending**.



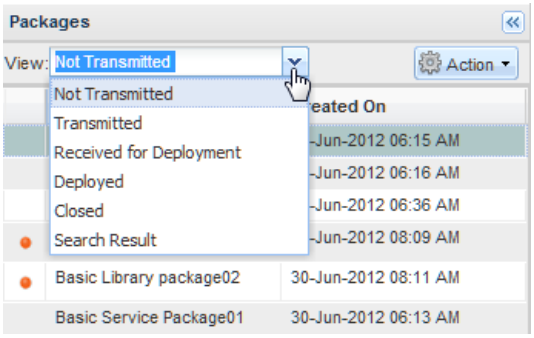
Hiding or Showing Columns

Move your mouse over a column until the Column sort button appears ( ). Click the Column sort button, choose **Columns**, and then check the check boxes of the columns you want to display.

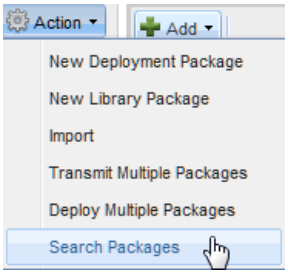


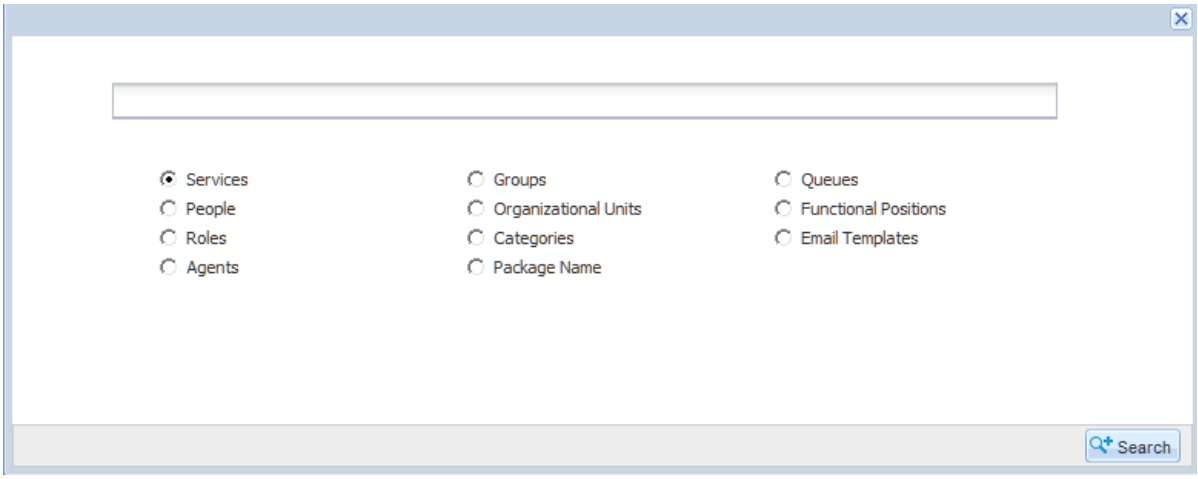
Searching Deployment Packages

From the **View** drop-down menu on the View and Search pane, choose a status to filter packages by status:



The View and Search pane not only allows you to choose the status of the packages you would like to work on, but to search for a particular package by choosing **Action > Search Packages** to display the Search dialog box.





You may search by package content (the name of any entity of the specified type included in the package) or by Package Name by clicking its radio button in the search list and clicking **Search** (Search). The search text may contain the wildcard of '*' to search based on partial words; for example, Email*. The search finds any packages, regardless of their status, that meet the specified criteria, and lists the package names in the View and Search pane under the **Search Result** view selection of the View drop-down menu. You can scroll through these, choose the one of interest, examine its contents, and, if its status allows, update those contents.

Note The search results only return packages where the entity type or search value entered were primary entities within the package. Primary entities are those entities (service definitions, organizational units, groups, queues, people, functional positions, roles, categories or email templates) that were chosen for inclusion in the package.

Creating and Deploying a Deployment Package

A deployment package follows this flow in Catalog Deployer:

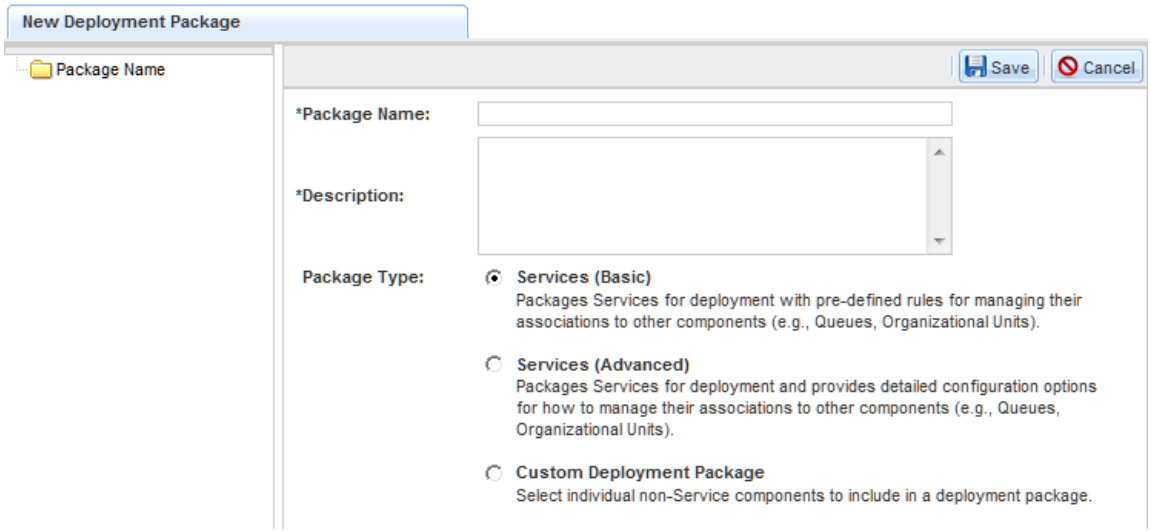
- **Create a Deployment Package**. Persons granted permission to access and use Catalog Deployer can create a deployment package and choose the entities to be included in the package. See the "Creating a Deployment Package" section on page 5-24.
- **Assemble the Package**. When the underlying content is considered ready for deployment, a deployment package is assembled by clicking **Assemble Package** (Assemble Package) in the Action pane. At this time, Catalog Deployer extracts the content and creates a copy saved as part of the deployment package. Once package assembly has occurred, any changes to the entities included in the package are not captured unless the package is reassembled. See the "Assembling a Deployment Package" section on page 5-27.
- **Transmit the Package**. The assembled package is sent to a target site for deployment. Packages may be transmitted via Catalog Deployer, or, if there is no direct connectivity between source and target sites, you may perform this step "off-line" using the export and import process. See the "Transmitting a Deployment Package" section on page 5-27.
- **Deploy the Package**. A person with permission to deploy the package selects the received package on the target site and runs the deployment. See the "Deploying a Package" section on page 5-30.

Log files record all activity that occurs within Catalog Deployer on each site. See the "Log Files" section on page 5-37.

Note It is only necessary to save packages when the package definition is modified. Package content is automatically saved.

Creating a Deployment Package

In the View and Search pane, create a new package by choosing **Action > New Deployment Package**.



Creating a deployment package consists of giving the package a name and description, specifying the type of package, and then clicking **Save** (Save).

- Standards for naming packages should be developed, to allow users to infer the package contents from its name. For example, the name could consist of a designation of the type of package, the source site, a description of the content, and the build number or date the package was created.
- You cannot create two packages with the same name in the source or target site.

Note The application will not allow the user to enter Package Names that use special characters (for example, no "\ /:*?"<>()[]" and so on). Spaces are allowed, but they are replaced with an underscore (_) in the Log XML Output file.

- Package description is required. Both the package name and description can be modified after the package has been created.

Note If needed, click the package name in the View and Search or Content pane to see the Package Name and Description fields.

- The Package Type cannot be changed once the package has been created. See the "Deployment Packages in Detail" section on page 5-34 for more information on Package Types.

Adding Content to a Deployment Package

When a package is created, it is assigned a status of "Not Transmitted". When it is clicked in the View and Search pane, its contents appears in the Content pane.

To add content to a package:

Step 1 In the Content pane, click the **Add** drop-down menu (Add) and choose the type of entity you wish to add from the list of available entities for that type of package.

A Search dialog box appears where you can search for and choose content.

Step 2 Choose the content you want by checking its check box.

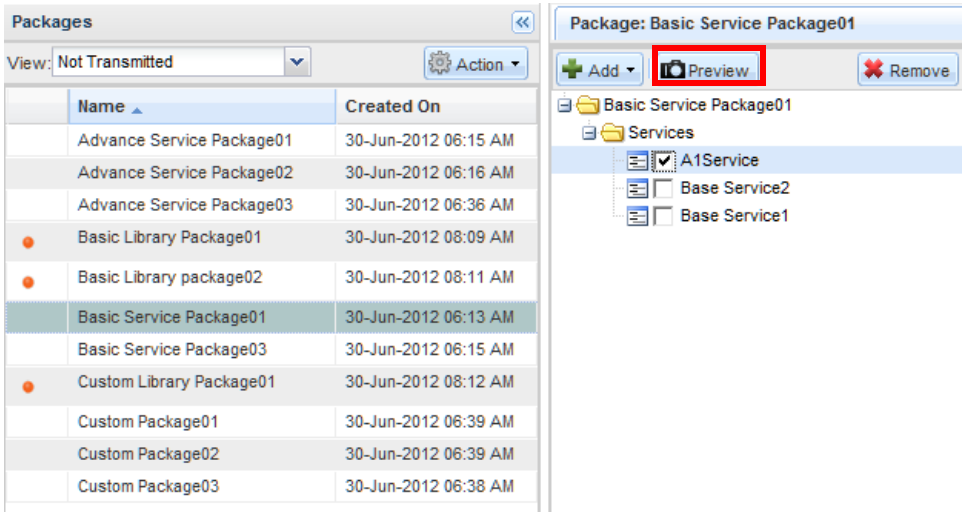
Step 3 Click **Add** to add the chosen content.

The content appears in the Content pane and is automatically saved. If you need to remove an entity, check its check box, click **Remove** (Remove) and then **Yes**. Content may be added and removed at any time until the package has been transmitted. If content is changed after the package has been assembled, the package must be reassembled.

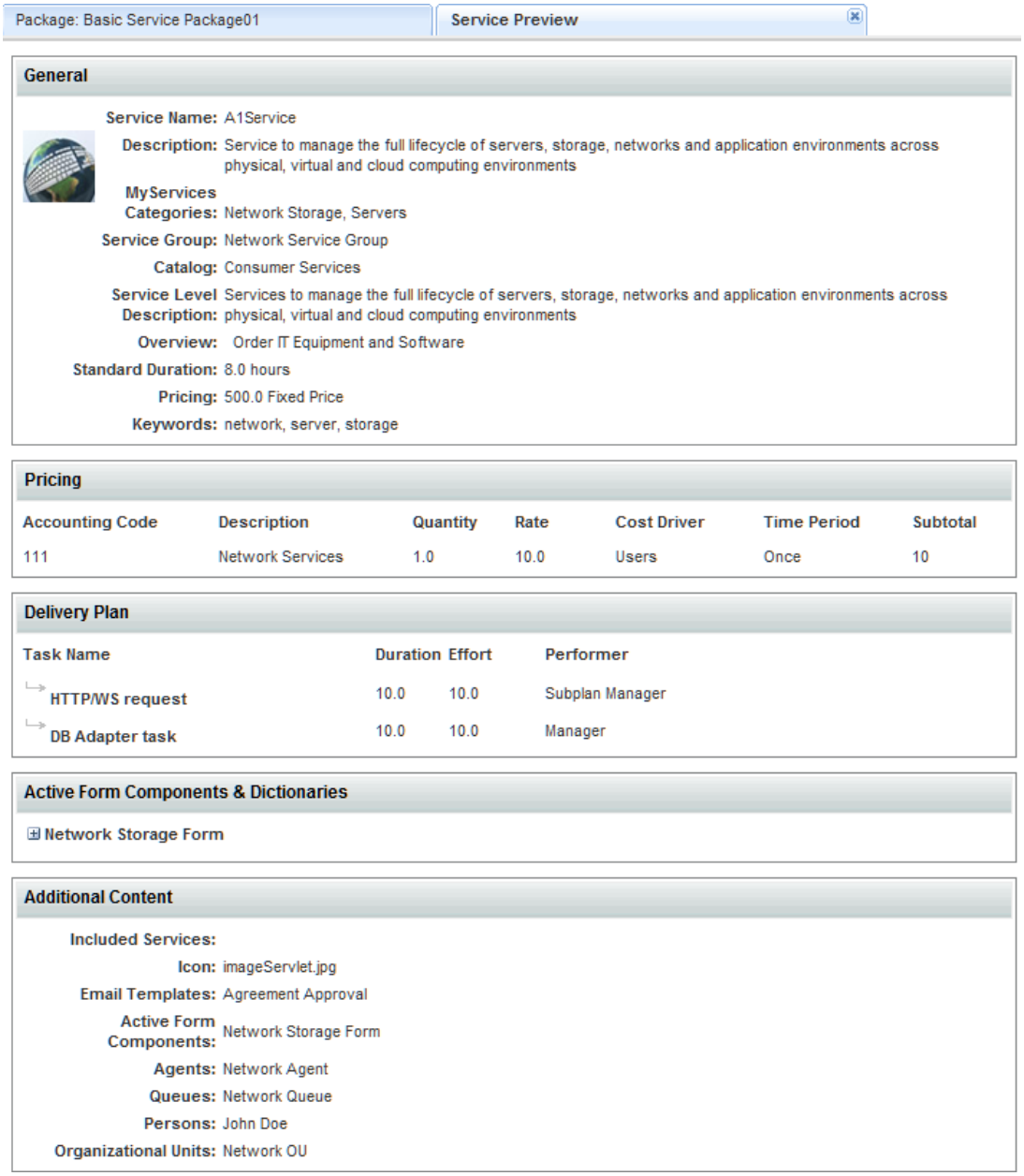
Previewing Package Contents

Basic and Advanced Services Deployment Packages allow you to preview the definition of a service.

For a service that is already been included in a package, go to the package's Content pane, choose the service to be previewed by checking its check box, as shown below, and then click **Preview** (Preview).



The preview appears in its own tab, as shown below.



A service preview consists of a summary of the service definition, as well as a rendering of the service form. All entity references are summarized in the “Additional Content” section at the end of the preview. This may help you ensure that these entities are present in the target environment before you deploy the service.

Assembling a Deployment Package

In the Action pane, click **Assemble Package** (Assemble Package) to assemble a package. Click **OK** to confirm that package assembly was successful. Once a package has been assembled, its status remains "Not Transmitted". It can be reassembled if the definition of any of its components changes, or if you want to add or delete entities to be deployed. The log entry for package assembly includes both the primary entities specified and any component entities that will also be deployed. In the History section of the Action pane, the log may be viewed by clicking the **View Log** link.

Transmitting a Deployment Package

Content can be transmitted directly to another site if:

- The target site has been defined in Administration > Settings > Entity Homes.
- A datasource corresponding to the target site has been defined in the JDBC data source page on the application server console.

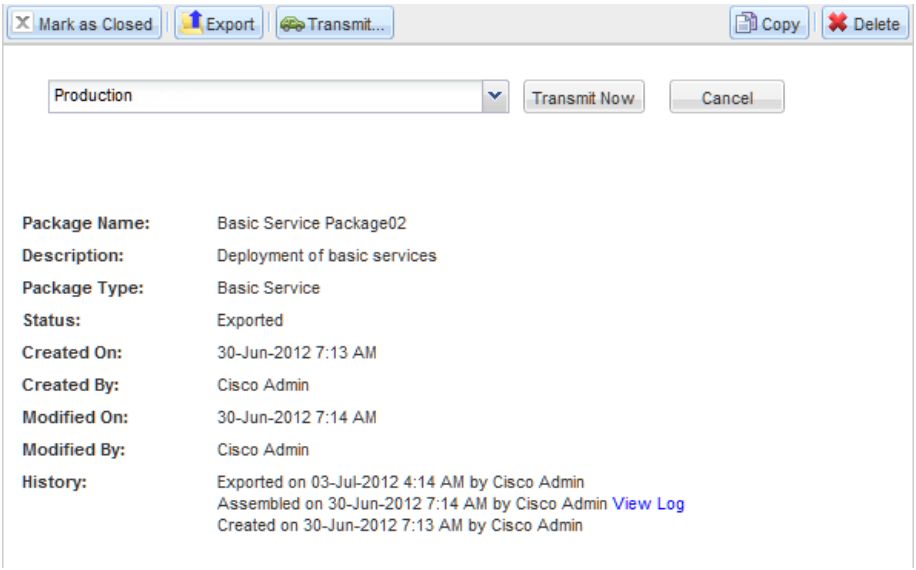
To transmit a package:

Step 1 In the View and Search pane, use the View drop-down menu to view packages with the status of: **Not Transmitted**.

Step 2 Locate the package within the list. Click the package name to view its information in the other panes.

Step 3 If the package has not been assembled, in the Action pane, click **Assemble Package** (Assemble Package). Click **OK** to confirm that package assembly was successful.

Step 4 In the Action pane, click **Transmit** (Transmit...).



Step 5 Choose the target site from the drop-down menu.

Note You can choose only those data sources for which your site administrators have configured. See the *Cisco Service Portal Designer Guide* for instructions. In situations where no sites are listed, the site may be configured to only use the export/import functionality of Catalog Deployer.

Step 6 Click **Transmit Now** (Transmit Now).

Step 7 Click **OK** to confirm that the transmission was successful.

The deployment package is transmitted to the target site. The status of the current package in the source site is changed to "Transmitted". A transmitted package may be transmitted to additional sites or exported.

Exporting a Deployment Package

A package can be exported, instead of or in addition to being transmitted. Exporting a package produces an XML file consisting of the content of the assembled package. The export file can then be imported into a target site and deployed.

Exporting a package provides a textual representation of the package. It can be used as offline archival or checked into a corporate source code control system.

To export a package:

Step 1 In the View and Search pane, use the View drop-down menu to view packages with the status of: **Not Transmitted** or **Transmitted**.

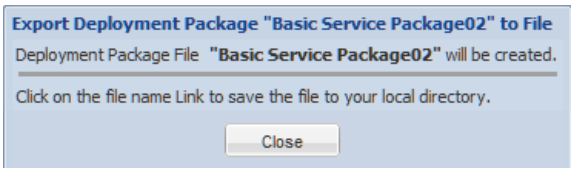
Note Packages which have been received for deployment, or deployed, cannot be exported.

Step 2 Locate the package within the list. Click the package name to view its information in the other panes.

Step 3 If the package has not been assembled, in the Action pane, click **Assemble Package** (Assemble Package). Click **OK** to confirm that package assembly was successful.

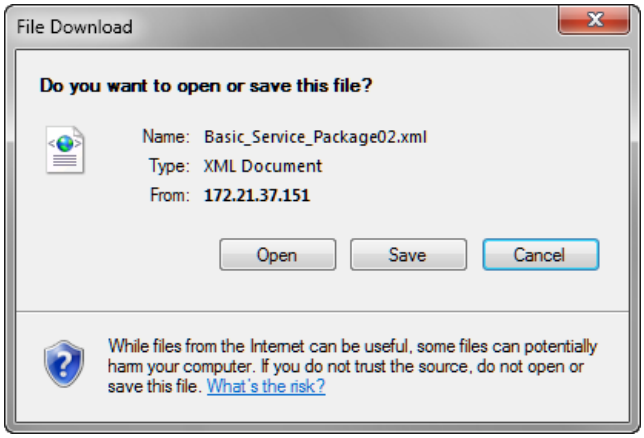
Step 4 In the Action pane, click **Export** (Export).

An export dialog box appears, as shown in the example below.



Step 5 In the dialog box, click the file name link (in the example above, the file name link is **"Basic Service Package02"**).

A File Download dialog box appears, as shown below.



Step 6 Click **Save**.

A Save As dialog box appears.

Step 7 Rename the file if desired and choose your desired destination.

The destination would typically be on a shared drive, accessible to all users with Catalog Deployer capabilities. Standards for naming directories and structuring subdirectories should be established to facilitate tracking packages. For example, a new subdirectory could be created for all packages deployed as part of the same change request.

Step 8 Click **Save**.

Step 9 Click **Close** to close the export dialog box.

Export/import can be used instead of transmitting a package in cases where security or other concerns do not allow directly transmitting a package from the source to a target site. The results are identical—the package is created in the target site in the "Received for Deployment" status, and can then be deployed.

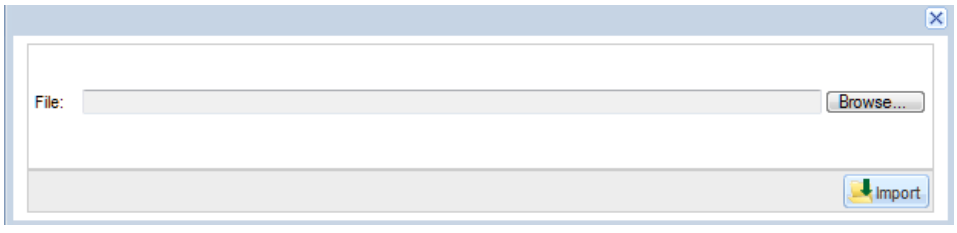
Importing a Deployment Package

An exported package needs to be imported into the target system.

To import a package:

Step 1 In the View and Search pane, choose **Action > Import**.

An import dialog box appears, asking you to browse for the file to be imported.



Step 2 Click **Browse**.

Step 3 Find and choose the package file.

Step 4 Click **Open**.

Step 5 Click **Import** ( ).

The deployment package is imported, assigned a status of "Received for Deployment", and opened. It can now be deployed.

Step 6 Close the import dialog box.

Note To import a deployment package back into the source site from which it was originally exported, you must first delete the identical package from the source site. The application will recognize duplicate files even if the filename has been changed.

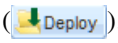
Deploying a Package

To deploy a package that has been transmitted to or imported into the target site:

Step 1 In the View and Search pane, use the View drop-down menu to view packages with the status of: **Received for Deployment**.

Step 2 Locate the package within the list. Click the package name to view its information in the other panes.

Note The deployment runs according to the deployment options and associated entity rules that were chosen on the source site. Users at the target site can click through the tabs and view the entities chosen, but cannot modify the package in any way.

Step 3 In the Action pane, click **Deploy** ().

Step 4 Click **OK** to confirm that deployment was successful.

All deployment actions are logged in the log file accessible by clicking the View Log link in the package's History section of the Action pane. The deployment log lists each entity deployed on the target site. The history will note, for example, if an entity was created or updated in the target site. After the deployment completes successfully, the status of the package changes to "Deployed".

If the deployment fails (for example, if an associated entity is not present in the target site), an error message appears and the package status remains "Received for Deployment" on the target site. You may fix the problem (for example, by creating the missing entity or, more likely, deploying a custom deployment package that contains the missing entity) and deploy the same package again. If successful, the package moves to the status of "Deployed" on the target site.

The assembled content of the package to be deployed must match the release level of the target site. Catalog Deployer tracks the application version under which the package was assembled, and will not allow deployment across different versions.

A Basic or Advanced Services package includes all the component design elements used by the services deployed. However, design components which are not stored in the database are not included in the deployment package and must be deployed separately. These elements include:

- JavaScript libraries referenced by any ISF Scripts

- Data sources added to the environment and referenced by data retrieval rules or option lists

For a complete list, see the *Cisco Service Portal Configuration Guide*.

In principle, deployment does not affect the service and component definitions used by requests that were in-flight when the deployment occurred. However, because of the dynamic nature of the requisition process, previously submitted requests are affected by:

- Changes to the content of rules or JavaScript functions and libraries
- Changes to the delivery plan of service requests that have not yet passed their final approval step

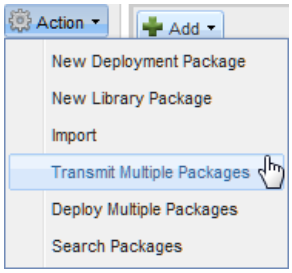
The deployment schedule and service designers need to take into account that in-flight requests based on the previous version of the service definition are affected by changes to the above elements.

Transmit and Deploy Multiple Packages

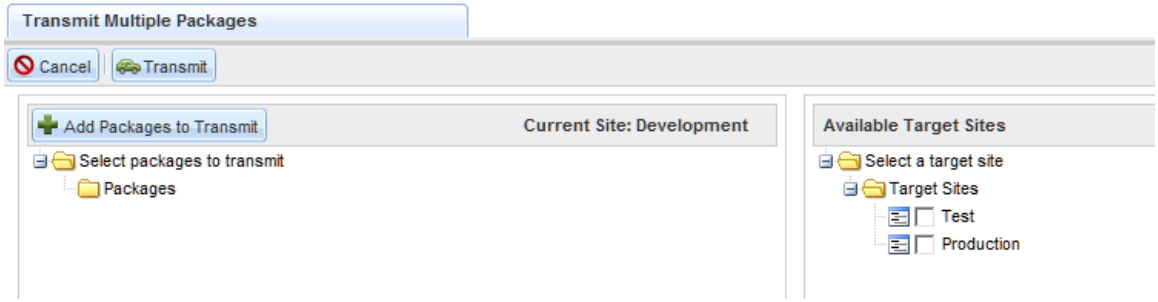
You may transmit and deploy multiple packages in one function. Catalog Deployer processes each package in turn, and provides the status for each. This procedure makes it much easier to deploy a large number of packages with limited user intervention.

To transmit multiple packages:

Step 1 In the View and Search pane, choose **Action > Transmit Multiple Packages**.



The Transmit Multiple Packages window appears, as shown below.



Step 2 Click **Add Packages to Transmit** (Add Packages to Transmit) to open a Search dialog box where you can search for and choose packages to be transmitted. The Search dialog box only allows you to choose packages with a status of "Not Transmitted" (which have been assembled) and "Transmitted".

Step 3 Choose the packages you want by checking their check boxes.

Step 4 Click **Add** to add the chosen packages.

The packages appear below the Packages folder in alphabetical order. All chosen packages are transmitted. If you need to remove a package, check its check box, click **Remove** (Remove) and then **Yes**.

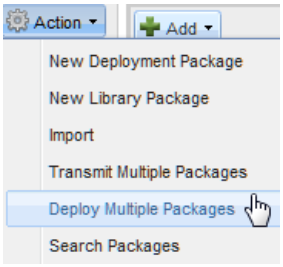
Step 5 Under the "Select a target site" folder, choose the target site by checking its check box.

Step 6 Click **Transmit** (Transmit...) to start the transmission.

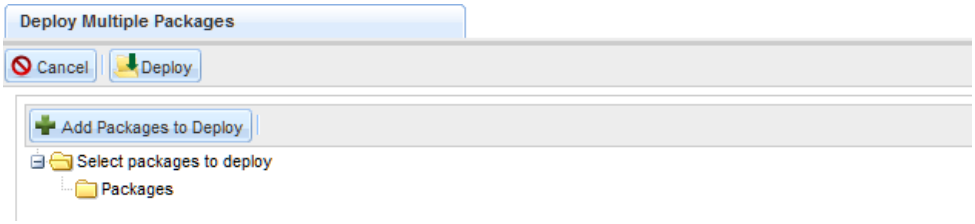
The packages are transmitted in the order in which they are listed (alphabetically). After the transmission process is complete, a status message appears, showing the transmission success or failure for each package. (The most common reason for a transmission failure is a version mismatch between the source and target sites.) The log file for each package is also updated.

A similar interface is available to deploy multiple packages:

Step 1 In the View and Search pane, choose **Action > Deploy Multiple Packages**.



The Deploy Multiple Packages window appears, as shown below.



Step 2 Click **Add Packages to Deploy** (Add Packages to Deploy) to open a Search dialog box where you can search for and choose packages to be deployed. Any package with a status of "Received for Deployment" or "Deployed" may be chosen.

Step 3 Choose the packages to be deployed by checking their check boxes.

Step 4 Click **Add** to add the chosen packages.

The packages appear below the Packages folder in alphabetical order. All chosen packages are deployed. If you need to remove a package, check its check box, click **Remove** (Remove) and then **Yes**.

Step 5 Click **Deploy** (Deploy) to start the deployment.

The packages are deployed in the order in which they are listed (alphabetically). Like the multiple transmission, multiple deployment indicates the success or failure for the deployment of each package; that information is available in the package's log file as well.

Closing/Reopening a Deployment Package

A deployment package in any status can be closed in the site in which it was created by clicking it in the View and Search pane, and in the Action pane, clicking **Mark as Closed** (Mark as Closed) Closing a package simply changes its status to "Closed", so that it appears only in the view of "Closed" packages, rather than in other views. This may make working with other packages easier, since there are fewer active packages to search through to find the one you want.

A closed package can be reopened at any time, for example, if you need to transmit it to another site or wish to export its contents. To reopen a package, click it in the View and Search pane, and in the Action pane, click **Reopen** (Reopen).

Deleting a Deployment Package

To delete a package, click it in the View and Search pane, and in the Action pane, click **Delete** (Delete). Click **Yes** to confirm the deletion. Deleting a deployment package permanently removes the package and its deployment history from the current site. Although package contents are compressed, the XML required to represent the package components may be quite large. Therefore, deleting packages will recover usable space in the repository/database. Package contents could be recovered if the package was previously exported; however, the complete deployment history of the package at the current site could not be recovered.

Copying a Deployment Package

Unless you're very lucky, you will need to deploy the same entities multiple times as part of the build process. For example, you deploy one or more services from the development to the test environment; the testers find some defects that need to be repaired; the service definitions are repaired in development and redeployed, so that the fixes can be verified in test before being deployed to production.

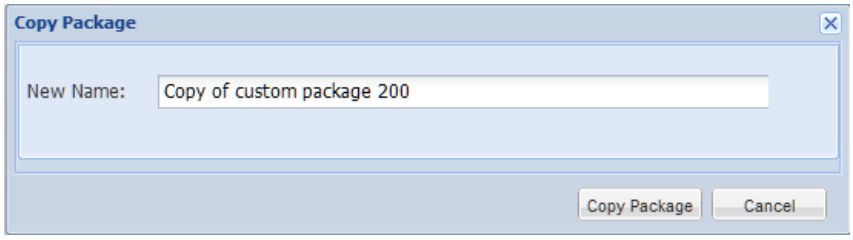
The ability to Copy a deployment package facilitates this work flow. Once an initial package, with the desired content, is produced, you can copy the package.

To copy a package:

Step 1 Choose the package in the View and Search pane.

Step 2 In the Action pane, click **Copy** (Copy).

A Copy Package dialog box appears, as shown below.



Step 3 Rename the package.

Step 4 Click **Copy Package**.

Step 5 Click **OK**.

The new package is created and opened with a status of "Not transmitted"—it contains the entities specified in the Content pane, not the assembled package content. You can then reassemble the package as required.

Deployment Packages in Detail

The deployment package types—Basic Services, Advanced Services, and Custom—differ in the primary entities each can deploy and the options available for governing deployment of associated entities.

Basic Services Deployment Packages

A Basic Services deployment package deploys the specified service definitions and their components. The deployment will proceed and skip entity associations if any entities associated with the services are not present in the target site.

The content of a Basic Services deployment package and its associated entities are summarized below.

Content Type	Action	Entity Types/Description
Primary Entity	Chosen via the Services content tab	Service Definition—All aspects of the service definition as defined in the Service Catalog option of Service Designer
Component Entities	Automatically deployed	All entities referenced by the Service Definition: • Categories • Data Dictionaries • Dictionary Groups • Active Form Components • Active Form Component Groups • Keywords • Objectives • Presentation Elements • Script Functions • Script Libraries • Service Groups • Service Items and Service Item Groups • Standards and Standard Groups

Content Type	Action	Entity Types/Description
Associated Entities	Deployment will skip the entity association if any is not present in the target site	All entities referenced by the Service Definition: • Agents • Email Templates • Functional Positions • Groups • Organizational Units • People • Roles • Queues

By default, Catalog Deployer deploys Standards data as well as the Standard definition. You can override this behavior by disabling the Administration setting to "Deploy Entries (data) in Standards Tables". This might be desirable, for example, if a different set of data should be used in different environments or data is being provided via a file import from an external source.

Advanced Services Deployment Packages

An Advanced Services deployment package deploys the specified service definitions and their components. An Advanced Services deployment has two major differences from a Basic deployment:

- All services comprising a bundle can automatically be deployed by choosing the parent service and indicating it is a bundle.
- The user can specify which action to take if associated entities do not exist on the target site at the time of the deployment. This overrides the behavior of a Basic Services deployment, which automatically fails if any associated entity does not exist.

The options which govern behavior of Catalog Deployer in an Advanced Services deployment package when the associated entity does not exist in the target site are summarized below.

Associated Entity	Available Options	Result
Agent	Skip	Reference to the agent is removed from the delivery or authorization task
	Fail	Deployment fails
Email Template	Create	Create an empty template in the target site.
	Skip	Reference to the template is removed from the delivery or authorization task
	Fail	Deployment fails
Functional Position	Skip	Association to the functional position is removed; assignment is to the Default Service queue
	Fail	Deployment fails
Group	Skip	Association to the group is removed; assignment is to the Default Service queue
	Fail	Deployment fails

Associated Entity	Available Options	Result
Organizational Unit	Skip	Assign the task to the Default Service Queue
	Fail	Deployment fails
People	Skip	Assign the task to the Default Service Queue
	Fail	Deployment fails
Queues	Create	The queue is created only if the organization for the queue exists in the target site; otherwise, deployment fails
	Skip	Assign the task to the Default Service Queue
	Fail	Deployment fails
Roles	Skip	Assign the task to the Default Service Queue
	Fail	Deployment fails

Since the Skip and Create (available only for selected entity types) options substantially change the definition of the service, they should be considered useful for “quick-and-dirty” deployments only. The service's form component would be transferred intact; however, potentially significant changes could be made to the delivery plan.

Custom Deployment Packages

Custom deployment packages allow you to deploy entities other than service definitions. The entities that can be included in a custom deployment package are:

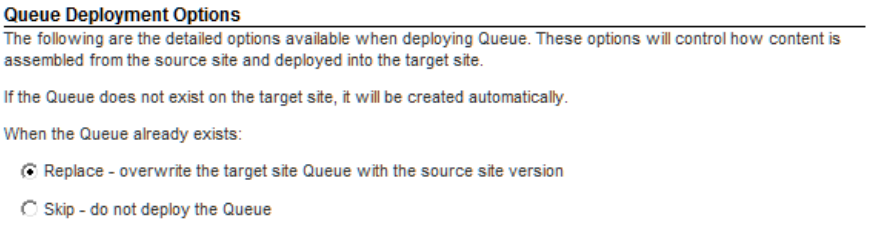
- Email Templates
- Functional Positions
- Groups
- Organizational Units
- People
- Roles
- Queues
- Categories
- Agents

Deploying categories via a custom deployment allows you to recreate all or part of the category structure of the source site in the target site. Chosen categories and their subcategories are deployed, and the relationship of the categories to services with which they are associated is re-established to match that in the source site. This supplements the capabilities provided by Basic deployment packages, where only those categories associated with the services being deployed are included in the package.

A package may contain more than one type of entity where there are interdependencies between the entities. For example, a person must be associated with an existing organization. Catalog Deployer automatically deploys entities in the correct order, so these interdependencies may be maintained.

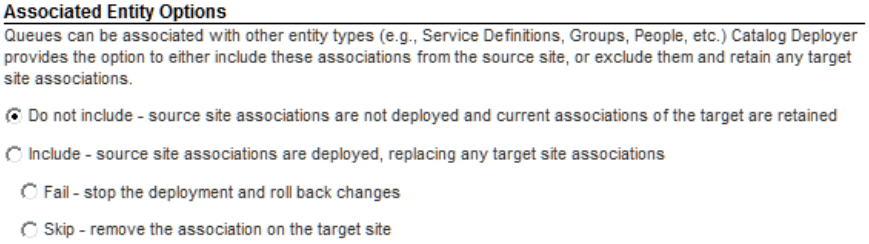
Further, there may be interdependencies among entities of the same type: a person may designate another person as a supervisor, or an organizational hierarchy may exist. In these cases, Catalog Deployer also deploys entities in the correct order (the supervisor entity first, for example) so the relationships may be re-established.

For each of the entities (except functional positions) to be deployed, you have an option to overwrite the entity definition at the target site with the new definition contained in the package, or to skip the deployment. This option is shown below for a queue:



Skipping entity deployment is risky since Catalog Deployer only checks for the existence of the entity in the target site, and does not inspect its content (for example, to see if the source is newer than the target). The Skip option can optimize performance by not reinstalling entities that already exist, but should be used in limited circumstances (for example, if you are deploying a large package for the first time into a target site where the entities are not known to exist). If any aspect of the deployment fails (for example, you attempt to deploy a person whose home OU is not in the target site and not in the current deployment package), you can fix the omission and redeploy the package. Only package contents that still do not exist in the target site would be deployed.

For each entity type (except email templates, functional positions, and agents), you also have the option to specify the behavior of Catalog Deployer with regards to re-establishing relationships to other related entities in the target site.



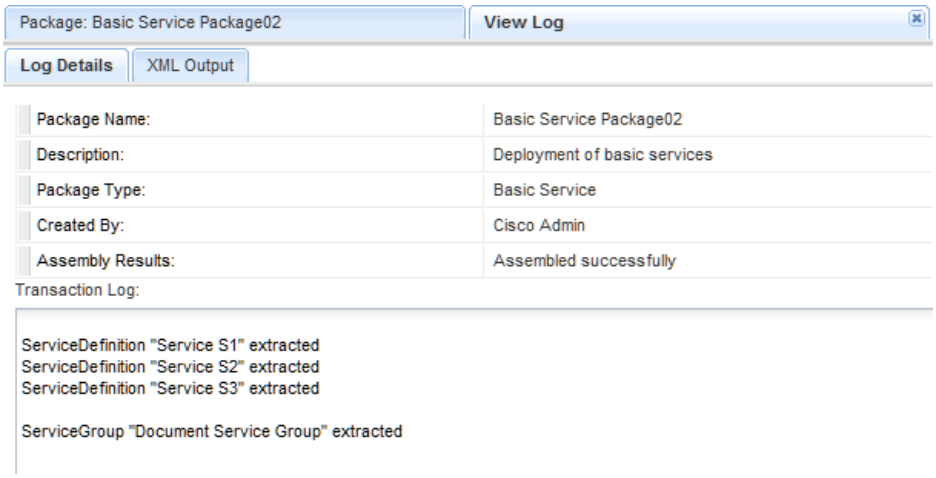
- Use “Do not include” to deploy only changes to the entity definition, without attempting to duplicate the relationships in the source site. This would be required if, for example, you want to deploy a new organization only, and not the relationships of this organization to numerous people, some of whom might not exist in the target site.
- Including source site associations will typically be required. This will duplicate the relationships in the source site at the target site. Using the “Skip” option will allow the deployment to continue if some of the associated entities are not present in the target site. For example, you may have assigned a queue to a service that is still in development and has not yet been deployed. If you use the “Skip” option, be sure to read the logs carefully, to ensure that no expected associations have been skipped.

Log Files

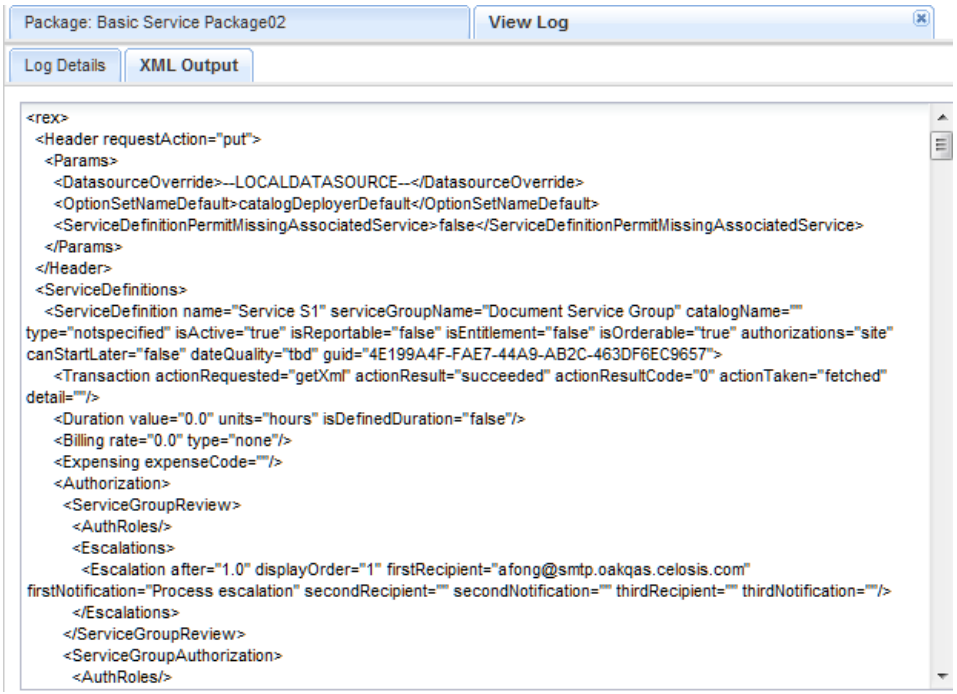
The **View Log** link in the History section of the Action pane displays the actions by Catalog Deployer in detail. Clicking on the **View Log** link opens a View Log tab with Log Details and XML Output subtabs. Logs record all activity that a package undergoes. Assembly logs list all entities included in the package. Deployment logs list all entities successfully extracted on the target site. If the package failed to deploy, the error also appears.

Logs include all package details (name, description), time/date stamps for the time of assembly or deployment, and all included entities.

A sample Log Details tab is shown below.



A sample XML Output tab is shown below.



Known Errors and Omissions

Catalog Deployer does not support renaming entities. Catalog Deployer will not behave correctly if you rename an entity at the source site and attempt to deploy it or deploy a service that references the renamed entity. In order to establish (or reestablish) an association with a related entity, Catalog

Deployer attempts to find the entity in the target site by name. For deployment purposes, an entity "name" is the name attribute of all entities except for people (identified by their login name) and organizations (identified by the combination of organization type and name). The result is that:

- For component entities of a service definition (dictionaries, form components), a new entity is created, and an association with this entity established for the service being deployed.
- For primary entities being deployed, a new entity, with the new name, is created.
- For associated (directory entities and email templates) entities, the deployment may fail or the entity association may be skipped, depending on the package type and deployment options chosen.

The workaround is to:

- Turn off site protection for the affected entities in the target site, if applicable.
- Rename the entity in the target site so the name matches that in the source site.
- Turn site protection back on.
- Deploy the entity.

During the development process, you should carefully track entity name changes, so they can be applied, as explained above, in the target sites before attempting to deploy the entity to those sites.

Sample Deployment Scenarios

Overview

This section gives details on recommended procedures to use in some frequent development and deployment scenarios.

Initial Deployment

You need to create a new Service Portal site. One option is to copy an existing Service Portal database to the site, and adjust the installation and configuration to use that database. (The procedure for doing this is documented in the *Cisco Service Portal Configuration Guide*.) However, another option is to perform initial configuration of the database and then use Catalog Deployer to deploy all entities that should populate the new site.

Perform Preliminary Configuration

Catalog Deployer does not deploy aspects of Service Portal configuration that typically do not change as the Service Catalog evolves. Therefore, you must define these elements as you did for the initial site.

- Re-enter Administration modules settings for settings, directory integration, authorizations, and entity homes. These settings should be identical to those in Development except for environment-specific configuration items. (For example, you may have separate LDAP directories for Development and Test.)
- Ensure that any Service Link custom adapters were included in the installation procedure, and that any changes you made to Service Portal adapters are reapplied.
- Re-enter agents and transformations (a known error and omission).

Deploy Service Foundation Entities

Entities maintained via Organization Designer (people, organizations, groups, roles, and functional positions) must be present in a site before a service definition can refer to them. Therefore, all such entities must be deployed before the service catalog can be deployed. Such entities should be included in one or more Custom deployment packages.

In addition, any custom email templates must be deployed. Email templates can be deployed at any time before the service that uses them is deployed.

Deploy Services

Once the foundation entities have been deployed, services may be deployed. For the initial deployment, keep the package size manageable (say, group things by service group) and be sure to review the log carefully. A basic deployment can and should be used, since all associated entities should have been deployed earlier.

Where Should Entities be Homed?

You should now have two (or more) sites in operation. All the services in Production should be identical to services in Development. (There may be additional services in Development that were part of the initial deployment, but that's okay.) In which environment will you be maintaining the entity definitions and membership (for groups, roles, and organizations)? In all cases, it is assumed that Entity Protection Levels are set to prevent any maintenance to entities in a nonhome environment.

All Directory-Related Entities Home Only in Production

The most straightforward approach is to home all entities that comprise a service definition in the Development instance and to home all entities related to people and organizations in the Production instance.

This makes using Catalog Deployer and maintaining the entities it deploys very easy—all aspects of all entities are always maintained in one and only one site. However, it complicates the work flow for creating and testing directory-related entities.

1. Before developing a new service, take an inventory of the queues, organizations, groups, roles, and functional units that are referenced in the delivery plan or other areas of the service.
2. If all of these associated entities already exist in the Development instance, it means they already also exist in Production (since they are homed in Production). Therefore, you can proceed with creating and testing the service.
3. If any of these directory-related entities do not already exist, log in to the Production instance and create them. Configure the entity definition, as well as its membership and roles.
4. Logged in to Production, create a Custom Deployment package containing the new entities and its associated entities. The package will need to use the option to “Use Source associations” for the new entities, since there are not yet associations in the target instance.
5. Deploy the custom package into Development.
6. Create the service using the new entities just deployed.
7. When the service is ready to be tested, create a Basic Services package containing the service.

8. Deploy (in this order) the custom package containing the directory entities to the Test site. Then deploy the Basic Services package.
9. Once the service passes testing, deploy the same Basic Services package from Development to Production.

In this scenario, all work regarding the directory-related entities (except actually referring to them in a service definition) is done in one place, the Production environment. This is good, because all work is done in one place and is easy to review and monitor. However, this process does have the following potential disadvantages:

- You are probably adding steps to the development process: if you didn't get the entity definitions correct the first (or second or third) time, you will have to go back to Production, fix the entities, repackage, redeploy and retest. When you repackage, you will need to use the option to “Use Target Associations” for the primary entity, since it has already been associated with a service that only exists on the target site.
- Another possible objection might be that you doing extensive development work in a Production environment, which some organizations see as a security risk.

Directory-Related Entities Home in Both Production and Development

An alternate approach is to home all entities that comprise a service definition in the Development instance (this should never change) and to home entities related to people and organizations in both the Production and Development instances.

Homing entities such as organizations, groups, roles, and people in both instances has the following advantages:

- You can work on the definition of these entities in the logical place, the Development environment. It may take a few passes, for example, to get a custom role definition just right. You wouldn't need to iteratively develop, deploy, and test—you could just develop and test, deploying when you are done.
- You can assign members to organizations or roles in the logical place, the Production environment, where all person information is automatically refreshed via directory integration and you are guaranteed that all Request Center users are represented.

Homing entities such as organizations, groups, roles, and people in both instances has the following disadvantages:

- Role-based access control currently allows access to a particular entity type. No further granularity is possible. For example, it is not possible to allow designers to define groups (and their roles) only in the Development environment, but only allow them to assign group membership in the Production environment.
- Since maintenance is happening in more than one environment, care must be taken when deploying packages. You must ensure that appropriate source and target associations to the entity are maintained.

As an example, assume you are developing a group whose members will consist of people across various service team organizations. How do you assign members to the group?

Scenario: People and Groups home in both Production and Development

- Create the group in Development, and assign some (test) members in order to verify that the capabilities and permissions granted via the group are correct. Not all people who must be in the group are in the Development environment, so the member list is incomplete and potentially incompatible with people in production (if you have some "test" people who do not have corresponding entries in Production).
- Use a Custom Deployment Package to deploy the group to Production, using source associations, but skipping an association if the referenced entity does not exist.
- Using Organization Designer in Production, associate the appropriate people with the group. You may do this either via the People or Group pages, whichever is more convenient, since both entities are home in Production and the entities could be maintained even with a "Create, Delete, Modify" protection level.
- If membership in the group changes, you only need to go to Organization Designer in the Production environment and make the appropriate changes. Further, directory integration has the ability to dynamically specify a list of groups in which a person is a member; if this capability is enabled (via updates to the enterprise directory and a corresponding directory mapping) no manual updates would be necessary.

Deploying a Service which needs a New Queue

You need to develop a new service or enhance an existing service to use a new queue. Following best practices, the queue should have a corresponding service team organization in which it is homed.

There are two possible scenarios here, depending on where you have "homed" queues and organizations. Each has pluses and minuses. (This is just a special, and very frequent case of the discussion in the previous section.)

OUs and Queues Homed in Production

This is a "clean" approach that places all work on organizations, people, and queues in Production. It was the only workable approach in versions on Service Portal prior to 2007, so people upgrading from those versions may prefer to keep using it.

- A disadvantage to this approach is that you are doing manual work in Production (creating queues and OUs), which some organizations may frown upon for security reasons.
- The advantage of this approach is that all work on the queues and OUs—not only defining them, but assigning their members —is done in one place.

1. Create the OU and queue in Production and assign appropriate roles.
2. In Production, assign members to the OU.
3. Deploy the new OU and queue from Production to Development. Include in the deployment package all service performers in the OU. You may wish to set the People deployed to skip existing people, to optimize deployment performance.
4. Create the service in Development.
5. Deploy the service from Development to Production.

OUs and Queues Homed in Both Development and Production

In this scenarios consider that the definition of the entity is home in Development, but its membership is home in Production. (Of course, this division cannot be enforced by entity homes, so both sites are designated as home for the entities to allow maintenance via Organization Designer).

1. Create the OU and queue in Development and assign appropriate roles. Assign enough members so you can thorough test the new configuration.
2. Create the service in Development.
3. Deploy the new OU and queue from Development to Production, using source associations, but skipping any that do not exist at the target site (to exclude “Test” people).
4. Use Organization Designer (or rely on Directory Integration) in the Production site to assign members to the OU.
5. Deploy the service from Development to Production.

Maintaining OUs and Queues after Initial Deployment

In both scenarios above, you are faced with possible changes to the initial queue and OU configuration after initial deployment.

- Changes in membership in the OU are handled as for initial deployment, that is, the OU membership is maintained in Production. It is not critical that all such changes be deployed back to Development, but if this is desired, a Custom Deployment package of the OU and new or affected People can be produced and deployed to Development, using source associations.
- Changes in the permissions assigned to the OU are applied in the site where the entity definition is home. The new OU definition is then deployed to the other site, using target associations.

Deploying Services that use a new Email Template

This is actually a fairly straightforward procedure. Two packages are needed:

1. In Development, create a Custom package containing the email templates. Use the default deployment option—replace the existing entity.
2. In Development, create a Basic Services package containing the revised services.
3. Deploy the Custom package to Production.
4. Deploy the Basic Services package to Production.

Of course, each package may be deployed as it is produced, provided the Custom package is deployed before the Basic Services package.

Renaming a Queue and Service Team

After several services have been deployed, which include tasks assigned to a particular queue, you receive feedback that the queue name is misleading to service performers, who would like it changed. Or perhaps the company has been restructured and organizations (including Service Teams) need to be renamed to reflect the new organization.

This runs smack up against a "Known Error and Omission" documented in the "Known Errors and Omissions" section on page 5-38. Since Catalog Deployer works by matching the names of entities across sites, it cannot match an entity that has been renamed. In some cases (as documented above), Catalog Deployer does fail and reports an error. In other cases, however, Catalog Deployer will create a new entity in the target. In the case of a renamed queue, any services previously deployed would still use the old queue. Only services deployed after the queue was renamed and deployed would reference the new queue. This is clearly not the intent of the designer.

The only way to prevent this behavior is to put in place processes that carefully monitor and control entities that need to be renamed. The processes would vary slightly, depending on whether queues and organizations are homed solely in Production, or in both Production and Development, but both involve manually renaming (via Organization Designer) the entities in both sites.

Scenario 1: Entities are Home in both Development and Production

1. In response to the requirement, the Organization Designer renames the queue and service team in the Development instance, through normal maintenance procedures.
2. An Administrator manually (via Organization Designer) renames the entities in the Production instance. Since the entities are homed in both instances, entity protection levels would normally allow this step.
3. Service Designers work on services using the renamed queues and service teams. Such services can be deployed through standard procedures.

Scenario 2: Entities are Home Only in Production

1. In response to the requirement, the Organization Designer renames the queue and service team in the Production instance, through normal maintenance procedures.
2. An Administrator relaxes entity home protection levels in the Development instance. This is necessary, since no manual changes to queue definition or membership is typically allowed, since the entity is homed in Production.
3. The Organization Designer renames the entities in the Development site.
4. The Administrator turns entity protection levels back on after the changes have been applied.
5. Service Designers work on services using the renamed queues and service teams. Such services can be deployed through standard procedures.

Changing a Category and its Icon

Categories are deployed as component entities as part of a basic or advanced service deployment. This is an effective strategy when your main objective is to deploy a new or revised service definition and to ensure that the associated categorization is also deployed. However, using a services package is not effective when the category hierarchy or images associated with categories have changed.

A custom deployment package offers the option to deploy all or a portion of the category hierarchy, independent of any services that may reference that hierarchy. When the category structure is deployed to the target environment, Catalog Deployer automatically re-establishes the associations between each category and its services.

As always, the warning against simply renaming an entity, in this case, a category, applies. If a category no longer applies, you should delete its association to services and create a new, replacement category. The alternative is to rename the category in the source and all target instances:

1. The Catalog Designer renames the category in Development (where categories are home).
2. An Administrator turns off entity protection in the Production site.
3. The Catalog Designer renames the categories in the Production site.
4. The Administrator turns entity protection levels back on after the changes have been applied.
5. Service Designers work on services using the renamed categories. Such services can be deployed through standard procedures.

Renaming Entities after a Service Portal Upgrade

Upgrading from versions of Service Portal prior to 2008 to 9.1 produces a set of new entities for each service definition. These entities are assigned distinctive names, so Service Designers can recognize them as having been produced by the upgrade process:

- Every service definition has a corresponding active form component. The name of the form component is: “UPGD: Form”, followed by the service name.
- All form components are assigned to a form component group. The name of the group is: “UPGD: Form”, followed by the name of the service group where the service definition corresponding to the form resides.

Service Designers will want to, at a minimum, rename the artifacts produced by the upgrade process, so the names are more user friendly. (They may also want to refactor the structure of the form components, to promote reuse, but that is another issue.) This is a “standard” Rename scenario:

1. The Service Designer renames the form component and form component group in Development (where these entities are home).
2. An Administrator turns off entity protection in the Production site.
3. The Catalog Designer renames the form component and group in the Production site.
4. The Administrator turns entity protection levels back on after the changes have been applied.
5. Service Designers work on services using the renamed form components. Such services can be deployed through standard procedures.

Adding a Custom Functional Position

You defined a Functional Position in Development (related to an OU) and specified the person holding that Functional Position for one or two organizations. You’ve changed the service in Development to route tasks to that Functional Position. How do you deploy the new and changed entities?

1. Create a Custom package in Development.
2. The Custom package contains the functional position you created and the people who are assigned to that position. The deployment option for the people is to use source associations.
3. Deploy the Custom package to Production.
4. Create a Basic Services package containing the revised service.
5. Deploy the Basic Services package to Production.

Deploying to an Environment with Browser Cache Enabled

If the Browser Cache setting is enabled in the Administration Settings, changes made to icons and embedded images in the service catalog presentation will not take effect until the browser cache has been deleted. To prompt the application users to delete their browser cache, follow the instructions in the *Cisco Service Portal Configuration Guide* to increment the browser cache version.

Branded Content Libraries

Overview

Cisco distributes content in the form of Branded Content Libraries in certain product releases. Consult the Cisco Technical Assistance Center (TAC) if you want to find out the availability of such content libraries for the release you are using. Two types of library packages are available:

- Service Library, containing service definitions and their component entities
- Custom Library, containing entities associated with a Service Library

The services in these libraries offer models for commonly required services for End User Management, Access Management, Data Center Management and other areas. Library contents can be previewed and desired items deployed to a development environment. This provides a head-start for designing, configuring, and customizing these services to enterprise requirements.

Custom libraries contain entities such as queues and service teams (organizations), which are referenced in a service. These libraries can optionally be installed in conjunction with the corresponding service library. If the Custom Library is deployed, the service will use the predefined references. If the custom library is not installed, the service will refer to the "Default Service Queue" or remove the reference, as appropriate. Service designers are responsible for completing the task (delivery) plan configuration.

Deploying a Branded Library

Typically, a site administrator is responsible for administering and deploying content from branded libraries. If desired, a custom role which includes the capability to Deploy Branded Content Libraries may be created. See the *Cisco Service Portal Configuration Guide* or *Organization Designer Online Help* for details on creating and assigning roles.

Use the procedure below to deploy branded libraries:

- Obtain the libraries. Libraries and corresponding user guides are available for download from the Cisco software site.
- Install the library. The library must be installed on a directory accessible to the client workstation from which Catalog Deployer is run. A shared network drive will allow central storage of libraries.
- Ensure that users responsible for deploying library contents have roles that include this capability.
- Import the library. Import the library into Catalog Deployer. See the "Importing a Library Package" section on page 5-47.
- Review the library contents. A person with permission to deploy the library package selects the received package on the target site and optionally previews the library contents (services). See the "Previewing Library Contents" section on page 5-49.

- Deploy chosen library contents. Choose the received package, choose the package contents to be deployed and run the deployment. See the "Deploying a Library" section on page 5-49.
- Review the deployment log. Deploying library contents generates a log detailing all deployment activities in regards to creating or updating entity definitions. See the "Reviewing the Deployment Log" section on page 5-49.

Assigning User Roles

A user must be granted a role that includes the "Import Deployments" and "Deploy Deployment Package" capabilities in order to import, preview or deploy content from Branded Content Libraries. Predefined roles that include these capabilities are:

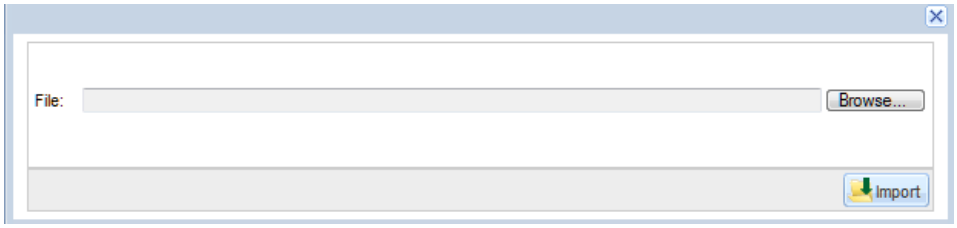
- Site Administrator
- Catalog Publisher

Importing a Library Package

Importing a library package is analogous to importing a deployment package:

Step 1 In the View and Search pane, choose **Action > Import**. (If that option is not available, the current user does not have a role that includes the "Deploy Branded Library Content" capability.)

An import dialog box appears, asking you to browse for the library file to be imported.



Step 2 Click **Browse**.

Step 3 Find and choose the library file.

Step 4 Click **Open**.

Step 5 Click **Import** (Import).

The library is imported, assigned a status of "Received for Deployment", and opened.

Step 6 Close the import dialog box.

A library package has attributes which identify the library and guide you in its deployment:

Attribute	Description
Package Name and Description	The name for the library and a brief description of its contents.
Library Version	The version of the library as released by the vendor.
Vendor Name	Cisco, or the name of the content provider.
Application Version	The version of the database for which deployment of the library is supported. Libraries are tailored to a specific version of Service Portal.
Image	An image icon associated with the library.
Package Type	Service Library or Custom Library.
Status	The current status of the library. Either “Received” or “Deployed”.
Created On/By Modified On/By	Catalog Deployer automatically tracks the person who created the library in this site (that is, who imported the library), and the date of the latest deployment activity and who performed it.
History	A brief log of all deployment activities performed against the library in this site with links to more detailed logs for deployment details.

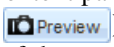
Creating a Branded Library Package

Licensed users with appropriate roles can create a new library by choosing **Action > New Library Package** in the View and Search pane.

The procedure is identical to the procedure for creating a deployment package (see the “Creating a Deployment Package” section on page 5-24) with the following exceptions:

- **Library Version** and **Vendor Name** are required. These are free-format text fields which identify the library.
- An orange ball icon ( ) is associated with a new library by default and is displayed in the first column of the View and Search pane (see the “Using Catalog Deployer” section on page 5-19). A custom image icon may be used by clicking **Upload Image** in the Action pane after the Library Package has been saved. Images can only be in JPG, PNG, or GIF format. They are displayed with a width of 16 pixels and a height of 16 pixels, so they should be created to maintain this aspect ratio. If the image is not sized correctly, it is resized by the browser and may appear blurry or pixilated.
- The Package Types available are:
  - **Service Library**, containing service definitions and their component entities
  - **Custom Library**, containing entities associated with a Service Library
- The library is also identified by the application version, the version of the Service Portal database under which it was created. Libraries can only be deployed into a Service Portal site running the same application version with which they were created.
- Library packages can only be exported, not transmitted. Just as for a deployment package, a library package must be assembled before export.

Previewing Library Contents

Service Library Packages allow you to preview the definition of a service before you deploy it. In the library's Content pane, choose the service to be previewed by checking its check box, and then click **Preview** (

). A service preview consists of a summary of the service definition, as well as a rendering of the service form.

Deploying a Library

Libraries are the vehicles for new content to be distributed from Cisco to customer sites. A library should be deployed into a development site only. Catalog Deployer is designed so that libraries can be deployed without disrupting or overwriting existing content that may have previously been customized to the customer's requirements. Therefore, the options for deploying a library package are configured so that library contents can supplement, but not replace, any previously created content with the same name on the target site.

- Primary entities in the package (a service or directory-related entity in a Custom package) will only be deployed if an entity by the same name is not found on the target site. Deploying from a library never replaces existing content.
- A service bundle always includes the parent service definition as well as all child Service Definitions. As is the case for all primary entities, the service definition is skipped (not deployed) if a service with the same name already exists in the target site.
- Component Entities are entities that are automatically deployed along with a primary entity. For example, deploying a service entails deploying the dictionaries, active form components, categories, and keywords used by that service. Such component entities are not deployed if an entity with the same name already exists in the target site.
- Associated Entities are those entities referred to by the primary entity. For example, queues, peoples, and organizations can be referred to by a service task plan, and are the service's associated entity types. If an associate entity exists in the target site at the time the primary entity is deployed, a relationship is established between the entities. If the associated entity does not exist, a default relationship (for example, to the “Default Service Queue”) is established.

The entire contents of a library containing supporting entities (such as queues and organizational units) must be deployed simultaneously. Conversely, Catalog Deployer allows you to choose the contents of a Service Library to be deployed. Each service is deployed in a separate transaction. If the deployment fails, the current transaction is rolled back.

Reviewing the Deployment Log

All import and deployment actions are logged for the library package. The **View Log** link in the History section displays the actions by Catalog Deployer in detail. See the “Known Errors and Omissions” section on page 5-38.

CHAPTER 6

Namespace

- Overview, page 6-1
- Expressions, page 6-3
- Email Templates, page 6-5
- Authorizations, Reviews and Delivery Tasks, page 6-12
- Namespace Reference, page 6-23

Overview

About Namespaces

The Business Engine provides facilities for service designers and system administrators to customize the contents of emails and to dynamically configure the progress of a requisition through the approval and delivery process by specifying conditional workflows.

In order to implement this capability, Service Portal provides a unified way to access the values of data elements, such as a request's initiator or the service form data related to a request. This unified way is referred as the Business Engine Namespace.

Namespace functionality increases the flexibility of the service design, by allowing the delivery plan to incorporate approvals or tasks that are executed conditionally, depending on the current value of a namespace variable, or to dynamically adjust the routing of a particular task. It also allows the content, subject, and addresses of emails to be dynamically adjusted.

Note The use of grid dictionary fields for Business Engine namespace is not supported.

This chapter describes Namespace in detail, its standards and implementation.

Namespace Definition

Namespace is a term used to describe a set of valid names that address the data objects used within Service Portal, exposing these objects to service designers. This allows designers to use these elements in these contexts:

- Within an email, to dynamically resolve the recipient, subject, or references within the email body.

- To conditionally execute reviews, authorizations, or tasks in a delivery plan.
- In expressions which determine the person or queue to which an authorization or delivery task is assigned.
- In task names.

Namespaces are hierarchical. Namespace names reflect the structure of the data which defines a service, and the data entered on the service form when that service is requested. The key to manipulating namespace variables is knowledge of the hierarchy which defines Service Portal data and the contexts in which particular Namespace variables can be used.

Each element in the hierarchy is case-insensitive. The elements are separated by periods (“.”). The last element may also be referred to as a “property”. All elements before the last one are “nodes” in the hierarchy.

References

When Namespaces are used in a textual context, as in an email template or an assignment expression, delimiters are required to differentiate the Namespace from surrounding text. This delimiter is the character “#” surrounding the namespace references.

EXAMPLES:

For emails:	Dear #Customer.Firstname#, We are pleased to inform you that the service you requested has been approved…
For expressions:	CN=#Service.Data.Initiator_Information.First_Name#
For conditions:	ActualCost > 2000.00

Nodes

Every node in the hierarchy has a node type. The type determines the subnodes and properties available for that element.

Node types are summarized in the table below.

Node Type	Description
Activity	Information about an activity (task), including its priority, status, and scheduled and actual start date, completion date, and duration
Service Form Data	Data in the fields in dictionaries used in a service
OrganizationalUnit	Information about an organizational unit
Person	Information about a person or queue, including name and, for person, company affiliation and contact information
Process	Information about a process (task), including its scheduled and actual dates and duration, and its current status and escalation level
Requisition	Information about a service request (shopping cart), including the start date and expected duration
Service	Information about the definition of the service requested, including form data
Message	Information about a failed Service Link task

The OrganizationalUnit (OU) and Person node types have multiple instances within the Namespace. For example, the Person node appears as the requisition's Customer and Initiator, as well as in many other contexts. In this case the “PersonType” is used to reference data for the appropriate person.

The Service node, and its children, relates to a particular service request. Elements in these nodes are available only in contexts when there is a current service request. In particular, emails and design configuration details pertaining to site- and organization-level authorizations, which apply to a complete shopping cart, rather than to an individual request, should not contain Service nodes. Similarly, the Order Confirmation Email Template, which pertains to the requisition rather than to individual service requests, should not contain Service nodes.

Details on PersonTypes, as well as the hierarchical structure of the namespace and the properties and subnodes available for each node are summarized in the “Person-Based Namespaces” section on page 6-27.

All dates are maintained in Greenwich Mean Time (GMT). Date data includes both the date and the time. All dates are presented in the format:

YYYY-MM-DD mm:HH:ss SSS

That is, a 4-digit year, 2-digit month, 2-digit (zero-filled) day, followed by minutes, hours, seconds, and fractions of seconds using a 24-hour clock. For example, “2007-06-27 23:19:39.197” corresponds to June 27, 2007 at 11:19 PM.

Expressions

Authorizations, reviews, and delivery tasks may be assigned to a person or queue or by using an expression. The expression provides a means to identify the person or queue, stored within Organization Designer, to whom a task should be dynamically assigned.

Configuring an Expression

The person/queue may be identified by using one of four assignment types:

Assignment Type	Meaning
CN	“Common name” – The full name of a user, which is expressed as Firstname Lastname
ID	User ID; the unique numeric identifier of a user
LOGINNAME	The login name of a user
QUEUE	The name of a queue defined in Organization Designer

Expression-based task assignments use the format:

```
AssignmentType=#Namespace_Expression#
```

The assignment statement should follow the rules below:

- The assignment operator (=) must immediately follow the AssignmentType and immediately precede the Namespace_Expression; no spaces are allowed.
- Namespaces used in the Namespace_Expression must be enclosed in hash marks (#).
- Expressions can consist of a combination of namespaces and constant data to create a person’s full name or the name of a queue. Separate expression elements by a single space.

- Do not include spaces in variable and functional position names, so that the expressions can evaluate properly.
- All lookups into the database using the result of an expression are case insensitive. For example, LOGINNAME assignment types in which the expression evaluated to “Ann” and “ann” would yield identical results; both would find a person whose login name was “ann”—or “Ann” or “ANN”.

CN (Common Name) Assignments

The CN assignment type attempts to find a person by matching the specified full name against the data stored in Organization Designer.

EXAMPLES:

```
CN=#Service.Data.Customer_Information.First_Name#
#Service.Data.Customer_Information.Last_Name#
CN=#Customer.FirstName# #Customer.LastName#
```

The values in the Namespace Expression must exactly match the name of the desired user as stored in Organization Designer. For example, a CN expression whose value is “Carroll Hastings” will not match a user whose name in Organization Designer is “Carol Hastings”. For this reason, you should not use dictionary fields into which users have been allowed to type data as free-format text. Instead, use dictionary fields that have been populated by using a Person data type, as this allows the user to choose people from a dialog box, eliminating typing errors. Alternatively, other Namespaces, such as the Customer, Initiator, or Performer, may be used if appropriate.

Use caution when using a CN assignment. It should not be used if two people could have identical first and last names. The assignment will always pick the first person (the one with the lowest PersonID) with the specified name.

When a field in a dictionary is defined as a Person data type, as is the “Select_Person” field in a person-based dictionary, its value is populated by choosing a person from the Select Person search box. The value is set to the unique identifier of the person in Organization Designer. This “ID” value may be referenced in a CN assignment.

EXAMPLE:

```
CN=#Service.Data.Approver.Select_Person#
```

ID (Identifier) Assignments

The PersonID for any of the Person nodes available via Namespace may be referenced in an ID assignment. Alternatively, a text field to which a PersonID has been copied or assigned may be used.

EXAMPLE:

```
ID=#Service.Data.TT_Contact.SupervisorID#
```

LOGINNAME Assignments

Login names uniquely identify a person in the application and can safely be used for assignments.

EXAMPLES:

```
LOGINNAME=#Service.Data.Customer_Information.Login_ID#
LOGINNAME=#Customer.HomeOU.Manager.LoginName#
```

QUEUE Assignments

A queue must sometimes be assigned based not only on a service team, but also on the location of the customer. Such location-based queues can be accommodated by using a QUEUE assignment.

EXAMPLE:

```
QUEUE=#Service.Data.RC_Queue.Location# Desktop Services
```

This assignment would direct the task or approval to a queue named "*Location* Desktop Services," where *Location* is the current value of the Location field in the RC_Queue dictionary on the service form. The resultant value for the QUEUE must *exactly* match the name assigned to a queue defined in Organization Designer. The match is case sensitive. If a match is not found, the task is directed to an Unassigned Work Queue.

Email Templates

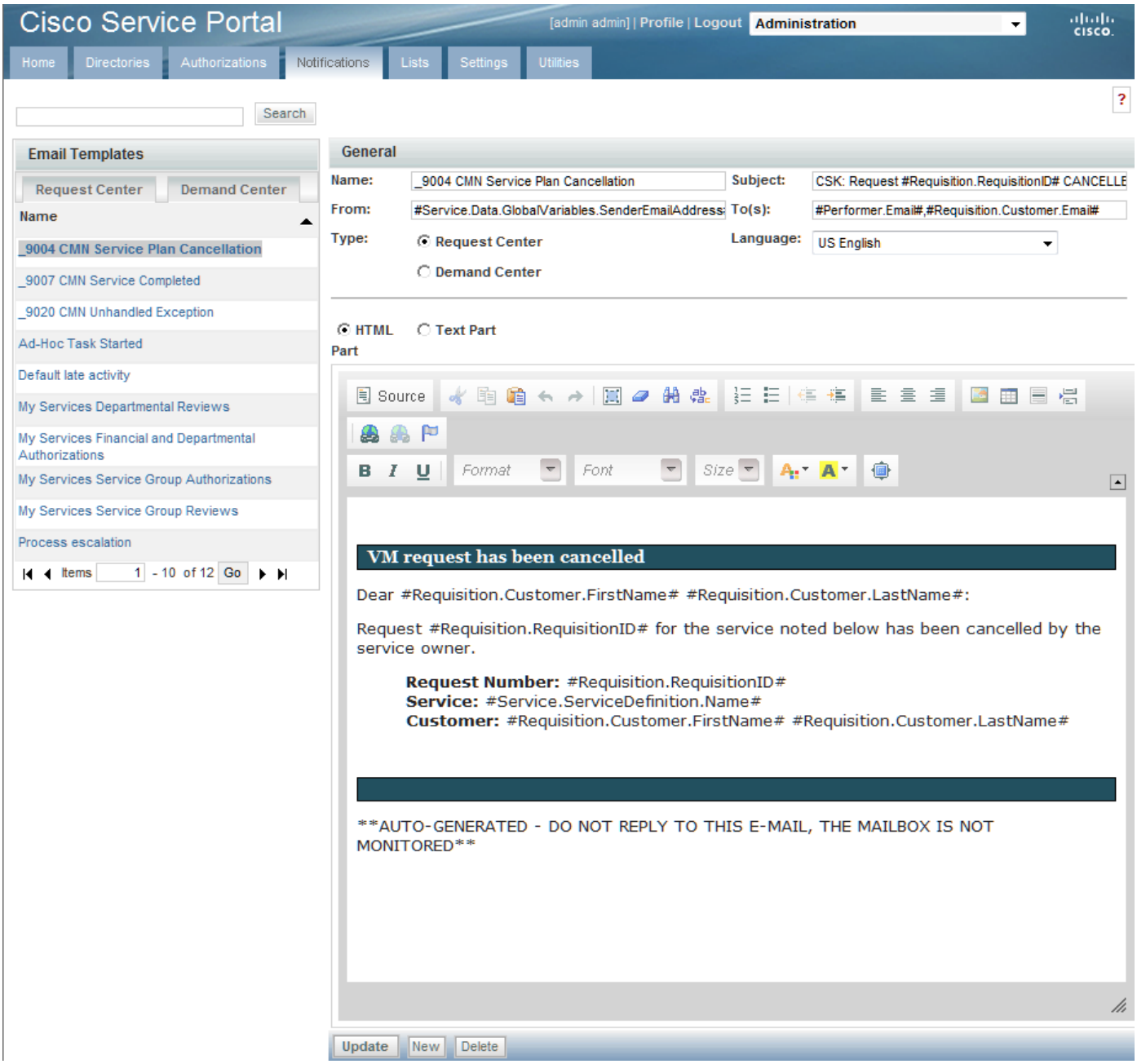
Namespaces can be used in the following sections of an email template:

- Subject – the subject of the email
- To(s) – the addressees/recipients of the email
- Body – the text of the email

Defining an Email Template

The Email Template definition page is accessed from the Administration menu:

Figure 6-1 Defining an Email Template



Previewing Email templates

To preview email templates:

Step 1 Start the Service Designer module.

Step 2 From the Services component, choose a service from the left side of the screen.

Step 3 Click the **Plan** tab.

Step 4 Click the **Email** subtab.

Step 5 Assign the email template you wish to preview to one of the moments listed in the drop-down lists in the Email subtab area.

Step 6 Click the corresponding **Preview** button to the right of the email template you just assigned.

Namespace Usage in Email Templates

The Namespace data available depends on the context in which the email is sent. For example, task-level data (such as Process or Activity details) will not be available for use in an email that is triggered by a Requisition-level event, such as a Financial Authorization. See the "Namespace Reference" section on page 6-23 for details as to which Namespaces are allowable in which emails.

Recipients

Namespaces can be used to send emails to people with an interest in the requisition.

Namespace Example	Resolves To
#Requisition.Customer.Email#	The customer's email, maintained on the To(s): field on the Notifications tab of the Administration module.
#Requisition.Customer.Supervisor.Email#	The email of the customer's supervisor.
#Alternate.Email#	The approver's authorization delegate. The delegate can be assigned via the user's Profile.
#Performer.Email#	The task performer's email, where the performer could be either a person or queue. The person's and queue's email can be found on the To(s): field on the Notifications tab of the Administration module.
#Position.EscalationManager.Email#	The escalation manager's email, where Escalation Manager is a user-defined functional position that has been associated, for example, with a Service Group, and to which a person on the Service Team has been assigned.
#Service.Data.**DictionaryName.FieldName#**	The current value of the specified field in the specified dictionary in the service form. The field should be validated to hold an email address.

Subject

In addition to hard coding data into the subject line of the email template, namespaces can be used to make the template more specific to the requisition (while still maintaining a consistent look throughout services).

Namespace Example	Resolves To
#Requisition.RequisitionID#	The requisition number
#Requisition.Customer.FirstName#	The customer's first name
#Service.ServiceDefinition.Name#	The service name

Body

As with the subject field, the body of an email can also be configured to include requisition data, task data, service data or service form data.

Namespace Example	Resolves To
#Requisition.RequisitionID#	The requisition number.
#Service.Data.**DictionaryName.FieldName#**	The current value of the specified field in the specified dictionary on the current service form.
#Site.URL#	The URL to access Service Portal; this will resolve to the user's default view, which is set under **Profiles > Preferences**.
#Site.URL#myservices/myservice.do?	A link to the My Services module.
#Site.URL#servicemanager/homepage.do	A link to the Service Manager module.
#URL#	A link to the Task Details (service form) page of the task in Service Manager.

Site URL Namespaces

The namespace #Site.URL# designates the URL through which Service Portal is accessed. The URL path may also include a specific module to run as well as parameters (following the question mark) to pass to that module. The examples above generate a hyperlink in the email body that will direct the user to the My Services module, the Service Manager module or to capabilities available within those modules.

It is possible to create a namespace reference that will link to just about any page in My Services or Service Manager. The key to determining the appropriate URL is to have a site administrator temporarily turn on site debugging (but please read the caveats in the *Cisco Service Portal Configuration Guide* carefully) and note the URL that appears when you navigate to the desired page. When embedding that URL in an email, you must use an encoded representation of characters such as ampersand (&).

Additional examples of such namespace references are given below.

Namespace Example	Resolves To
#Site.URL#myservices/navigate.do?query=requisitionentrystatus&reqid=#Service.Requisition.RequisitionID#&reqentryid=#Service.RequisitionEntryID#&formAction=displayEntryStatus&serviceid=#Service.ServiceID#&requisitionId=#Service.Requisition.RequisitionID#&waiting=0&**authTaskType=4**&activityId=#ActivityID#&buttonsPresent=true&	A link to the service authorization page within My Services with the current approval/review task selected. The Authorization Task Type of 4 refers to a Service Group authorization.
#Site.URL#myservices/navigate.do?query=authorizationtask&taskId=#ActivityID#&return_to_url=authorizationslist&	A link to the Service Group Authorization identified by ActivityID, which will take the user directly to the requisition containing the service requiring approval.
#Site.URL#myservices/navigate.do?query=authorizationlist&return_to_url=portal	A link to the authorizations list page within My Services where the user will see all the authorizations awaiting approval.
#Site.URL#myservices/navigate.do?query=requisition&requisitionId=#Requisition.RequisitionID#	A link the requisition confirmation page in My Services.

Site URL Configuration

As part of the installation process, administrators specify the Service Portal URL. This URL is stored in the configuration file be.properties. A sample entry is shown below.

```
!--------------------------------------------------------------------
!Define the URL for the application
!This is used in emails for constructing the various urls
!--------------------------------------------------------------------
ObjectCache.Application.URL=http://appsrv01.celosis.com/RequestCenter/
```

The entry for ObjectCache.Application.URL is the value of the namespace #Site.URL#. The URL typically ends with “/RequestCenter/”; however, it is possible to configure the application server to automatically reroute requests to /RequestCenter/, so this portion of the path may be absent from the properties file. In this case, the word **/RequestCenter/** must be added after #Site.URL#, such as #Site.URL#**/RequestCenter/**myservices/myservice.do?, to ensure that the namespace link resolves to the correct URL.

In order to determine if the /RequestCenter reference must be inserted, you may consult the be.properties file (or have the application server administrator report on its contents). An alternate method is:

- Enter #Site.URL# by itself in an email and send the email when, for example, a task starts.
- When receiving the email, note if the URL resolves to just http://www.mycompany.com or http://www.mycompany.com/RequestCenter/.
- If the resolved URL does not include the wording **/RequestCenter/**, then add **/RequestCenter/** after #Site.URL#.

Service Link Message Namespaces

The namespace node #Message# is available only within the body of email messages generated as a result of a Service Link failure to deliver an outbound message. The template to be used is specified as the “Failed Email” in the Service Link agent definition. Elements in this node may be useful in helping a administrator diagnose the cause of the failure.

Service Manager Task Details URL

The #URL# namespace takes the user directly to the form data of a task in Service Manager. This is very nice for Ad-hoc tasks, and for teams that rarely perform tasks in Service Portal. It can also be used for authorization tasks, but after the approver approves/denies the service, they find themselves in Service Manager, which can be confusing.

Some customers prefer this to the multistep process required to view the form data when approving via the #Site.URL#myservices/navigate.do?query=authorizationtask&taskId=#ActivityID# namespace. In fact, here, an approver can approve without seeing the form data which could be a slight audit issue depending if the data is important to the approval—they have to know to click on the name of the service to see the form data.

Namespace Processing and Alternate Values

When a Namespace element is referenced, Service Portal retrieves its value from the current context and replaces the reference to the Namespace in the email with the resolved value. Some Namespace elements are always guaranteed to exist and have a valid value, such as those relating to the customer or initiator of a request. Other Namespace elements, however, may be undefined at certain times. For example, a functional position may be vacant; no authorization delegate has been designated; or an employee is temporarily without a supervisor.

Any reference to a Namespace that is undefined is blank. Except for the email namespace, it is possible to provide an alternate value, by using a slash (/) immediately following the Namespace element name, and then assigning the alternate value.

EXAMPLE:

```
TO: #Alternate.email# #Performer.email#
```

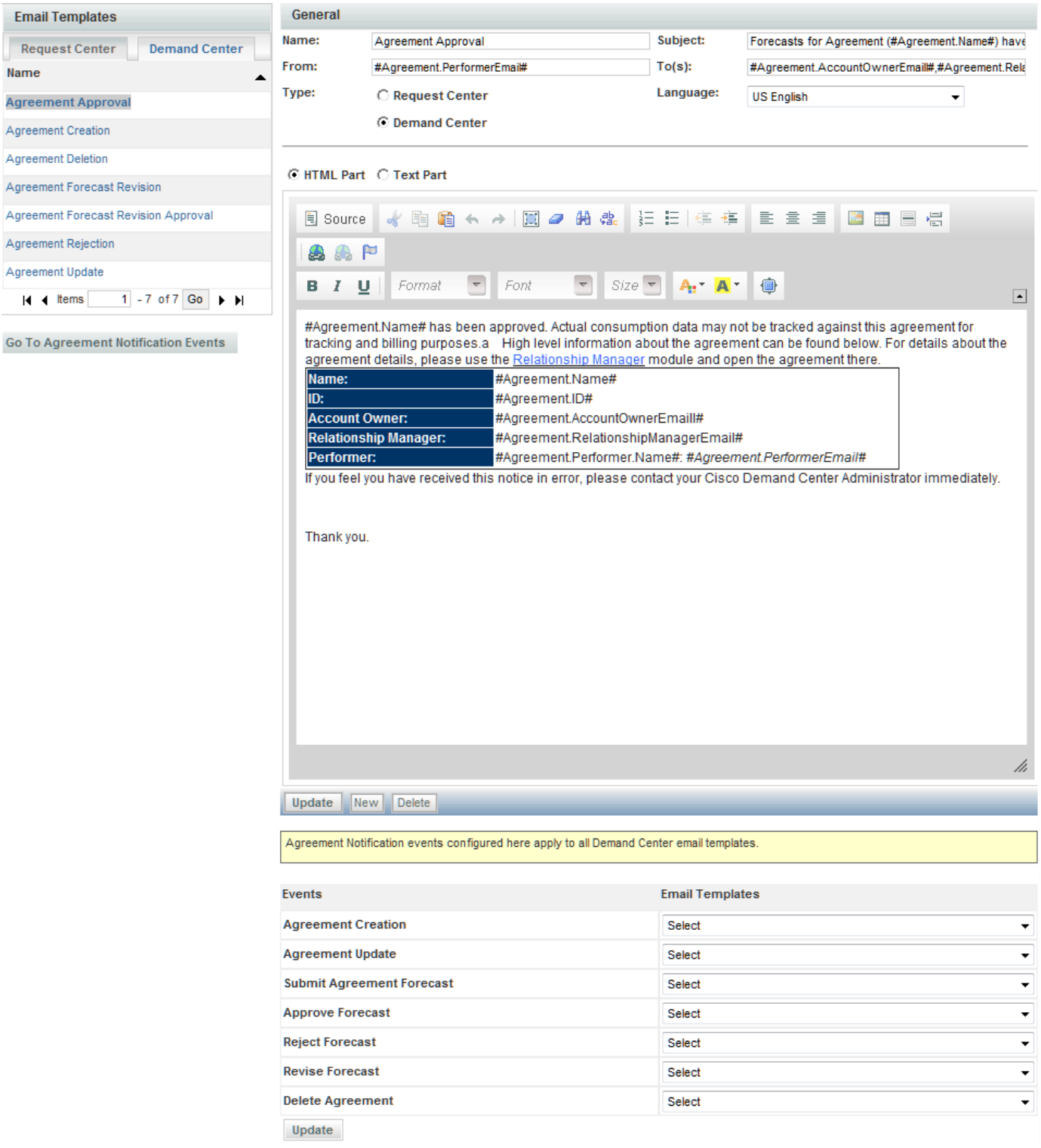
- Design emails to be sent to both the designated reviewer and a possible alternate (authorization delegate). If no delegate is currently designated, the #Alternate.email# is blank

  ```
  #Supervisor.LastName/Your current supervisor#
  ```

- If the person referenced currently has a supervisor, the supervisor's last name will appear in the email; otherwise, the string “Your current supervisor” will appear.

Demand Center Templates

Namespaces can be used in email notifications sent from Demand Center in response to an event which occurs while working with an agreement. Users can set up default Demand Center email templates to be sent in response to events listed in the Agreement Notification Events in the Administration module. Agreement Notification Events appear below a Demand Center email template, or click **Go To Agreement Notification Events** below the Email Templates list (you must click the **Demand Center** tab to show Demand Center email templates).



The mapping sets up default templates to use in response to these events. In addition to these default templates, templates can be customized for a particular service offering.

Authorizations, Reviews and Delivery Tasks

Authorizations and Reviews

Authorizations determine the review and authorization steps a requisition must flow through. Service Portal allows service designers and site administrators to establish authorizations at several levels:

- **Site**. Authorizations at the site-level establish the default authorization structure for all services for the site. These authorizations are maintained in the Administration > Authorizations tab.
- **Organizational Unit**. Authorizations at the organizational unit level establish the authorization structure for the departmental authorization and review by the current organization. The specified structure can be set to either override or supplement site-wide authorization. Organizational unit authorizations are maintained via the Organization Designer > Org Units > Authorization link.
- **Service Group**. Authorizations at the service group level establish the authorization structure for the service group. This structure can be set to either override or supplement the site authorization structure. Authorizations for service groups are maintained by choosing the Service Group in the Service Catalog of Service Designer and clicking on the Authorizations tab.
- **Service**. Authorizations at the individual service level establish the authorization structure for that service. Service-level authorizations are maintained via the Authorizations tab for the chosen service.

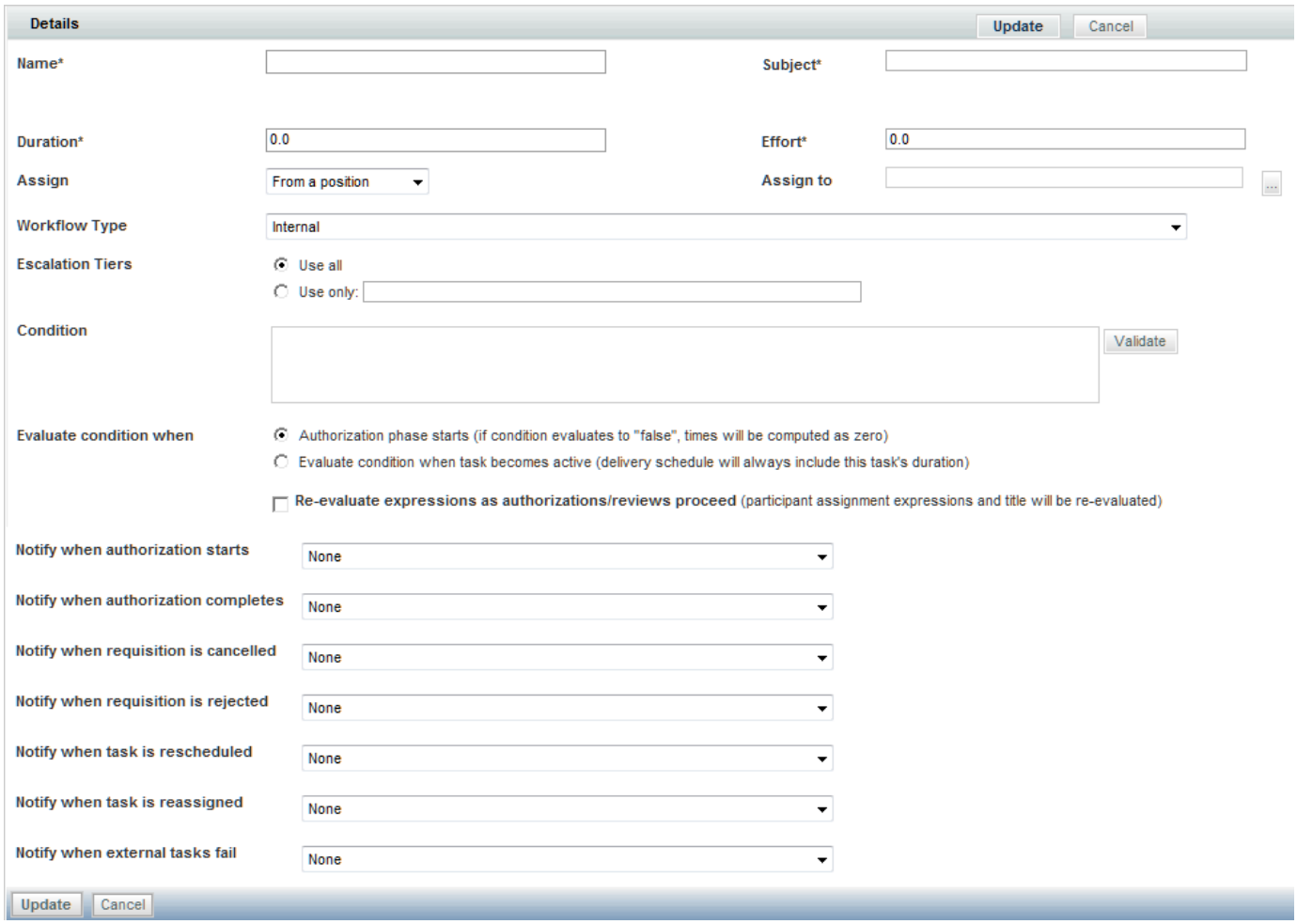
Namespace Usage for Authorizations and Reviews

Namespaces can be used in the following components of an authorization/review definition:

- Subject: Namespaces can be used to include form data in the task name
- Assign to: The authorization/review may be assigned from an expression.
- Condition: The authorization/review task may be conditionally performed

The subtab for specifying the details of an authorization or review looks like the figure below:

Figure 6-2 Service Definition Subtab for an Authorization Details



Subject

The subject of an authorization or review task can be configured to reflect requisition data through the use of namespaces, thus making the subject specific to the customer's order. Simply include the namespace in the Subject field.

The most frequently used namespace is the name of the service to which the current review/authorization applies. This is typically concatenated with other descriptive text; for example:

```
Group Review for #Name#
```

The namespace #Name# is available as a shortened form for the Service Name.

Only namespaces referring to a specific attribute of the service and **NOT** the content of a service form can be used for Organizational Unit Reviews or Authorizations or Financial Authorizations. For instance, when using a namespace to refer to the customer's first name:

```
#Requisition.Customer.FirstName# -- CORRECT
#Service.Data.Customer_Information.First_Name# -- INCORRECT
```

Service Group Reviews and Authorizations may use #Service.Data# namespaces. Such reviews/authorizations apply to an individual service, so the service (form) data is available. Financial Authorizations and Organizational Unit Reviews/Authorizations apply to an entire requisition, which may include multiple services (requisition entries); therefore, the service data for an individual service is not available.

Assigning from an Expression

Authorization and review tasks can be assigned to:

- the person currently filling a *functional position* defined in Organization Designer.
- a static person or queue.
- the person or queue identified by the value of an *expression*.

In many cases, assigning tasks to a predefined position, individual, or queue is not a viable option. For example, the same service may be handled by different queues, depending on the location of the requestor. In cases like these, you may need to route the tasks based on requisition data entered or otherwise supplied in the ordering moment.

Procedure:

Step 1 Choose **From an expression** in the Assign drop-down list.

Step 2 Type the desired expression in the “Assign to” field.

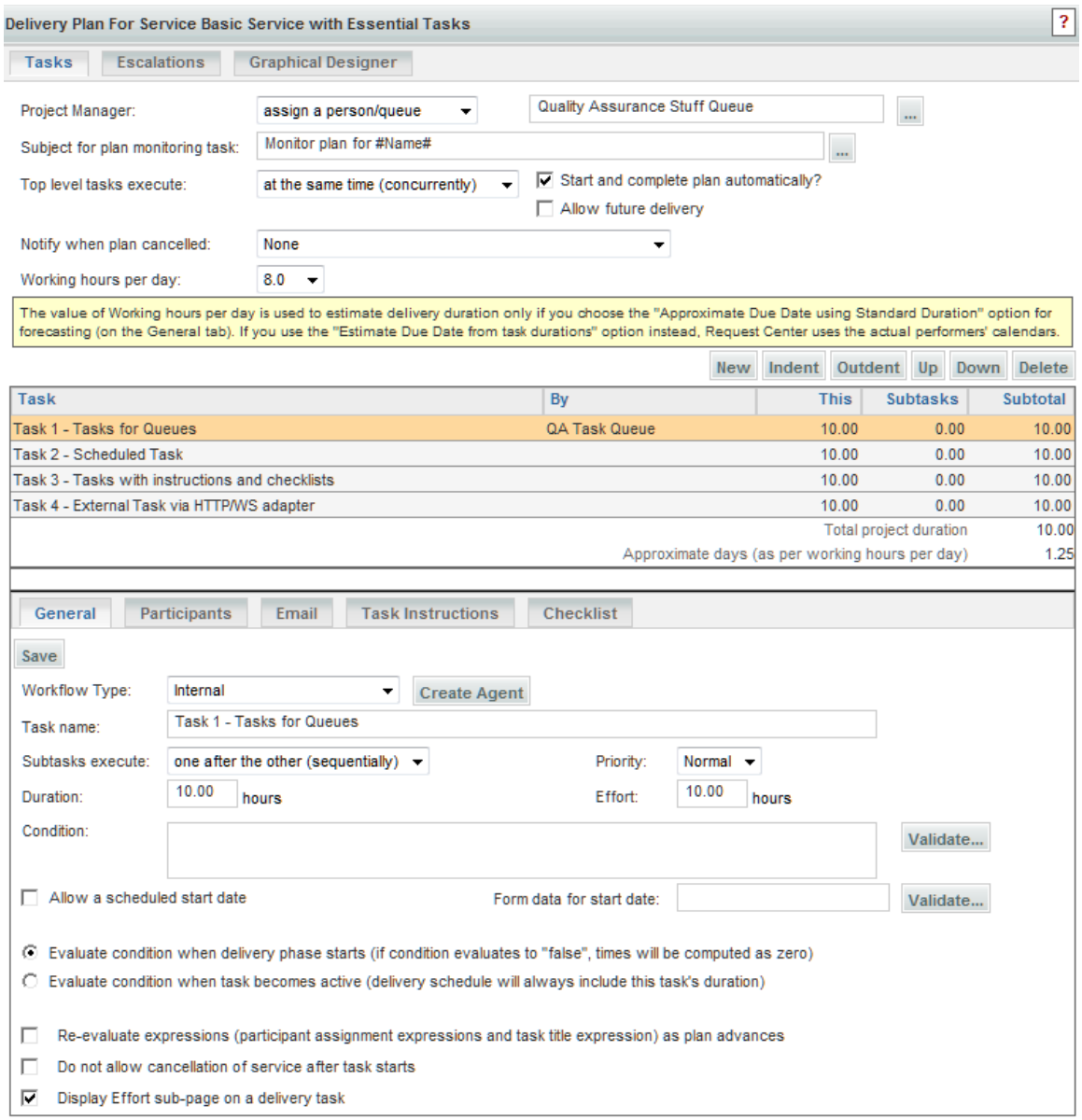
```
ID=#Service.Data.Customer_Information.SupervisorID#
```

Step 3 Click **Update** to save.

Delivery Plans and Tasks

The delivery plan is configured via the **Plan** tab of **Service Designer > Services**.

Figure 6-3 Monitor Task Definition



Each task is configured by choosing the task from the list of tasks (or clicking **New** to create a new task) and filling out the task-related subtabs.

Namespace Usage for Delivery Tasks

Configuring a service's delivery plan and component tasks provides multiple opportunities to use expressions and namespaces.

Namespaces can be used in the following components of an authorization/review definition:

- Project Manager: May be assigned from a namespace expression.
- Subject for Monitor Plan: May include namespaces as well as literal text.

- Task name: May include namespaces as well as literal text.
- Participants: Performer and supervisor may be assigned from a namespace expression.
- Condition: The task may be performed only if the specified condition, which may use namespaces, is true.

Project Manager for the Delivery Plan

The Project Manager may be assigned from a person, queue, or expression. The expression may use any of the Assignment Types to identify a person in Organization Designer.

EXAMPLES:

```
LOGINNAME=#Service.Data.Project.ManagerLogin#
ID=#Customer.Supervisor.ID#
```

Monitoring Task Subject

The subject for the monitoring task typically includes the #Name# namespace, to refer to the name of the service being monitored.

```
Monitor Plan for #Name#
```

Namespaces referring to performers, either people or queues, cannot be used.

Namespaces referring to the content of a service form can be used, but this may have side effects, as documented for the Delivery Task Name below.

Delivery Task Name

The delivery task name typically includes the service name, through the use of the #Name# Namespace. Service data (#Service.Data….#) can also be used. Since all tasks are created when the form is submitted, the value of the specified field must have been supplied in the ordering moment.

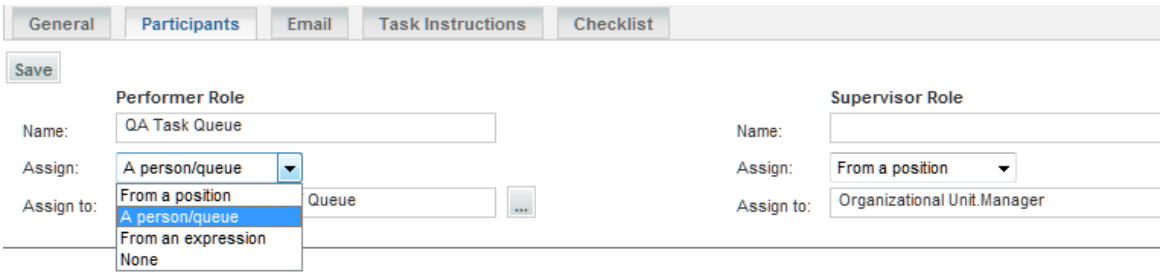
Note Using form data for a task name is not best practice, since the task would no longer be groupable by the task name in a reporting environment. Further, a Service Manager query would need to use the "contains" operator to retrieve the rest of the task name, which may adversely affect performance.

Assigning Roles (Performers and Supervisors) to the Task

As with authorizations, the service team or person assigned to tasks in the delivery plan may depend on data on each requisition, such as a customer's location. Expressions can be used to intelligently route tasks, rather than creating different forms or workflows for each possible scenario.

Plan tasks that can be dynamically routed include the Performer and Supervisor roles found on the Participants tab of each task:

Figure 6-4 Delivery Plan Task Participants



To assign “From an expression”, choose that option from the Assign drop-down list and click on the ellipsis next to the “Assign to” field. The Edit Expression window appears to allow you to enter and validate the expression.

Conditional Statements

A conditional statement allows tasks to be initiated or skipped based on the condition and can be configured to evaluate at various times within the authorization, review, and delivery plan.

Namespaces used in conditional statements are not enclosed in # signs:

```
Service.Data.DictionaryName.FieldName=”Yes” -- CORRECT
#Service.Data.DictionaryName.FieldName#=”Yes” -- INCORRECT
```

Conditional Authorization and Review Tasks

Authorizations and reviews may be needed only when certain conditions exist. For example, an authorization may be required only when a unit price exceeds a specified threshold. To define a conditional authorization or review task, follow the procedure below.

Step 1 Click on the authorization/review you are configuring. The Review/Authorization Details window appears, as shown in Figure 6-5.

Step 2 Enter the Condition under which the particular authorization or review task should be executed.

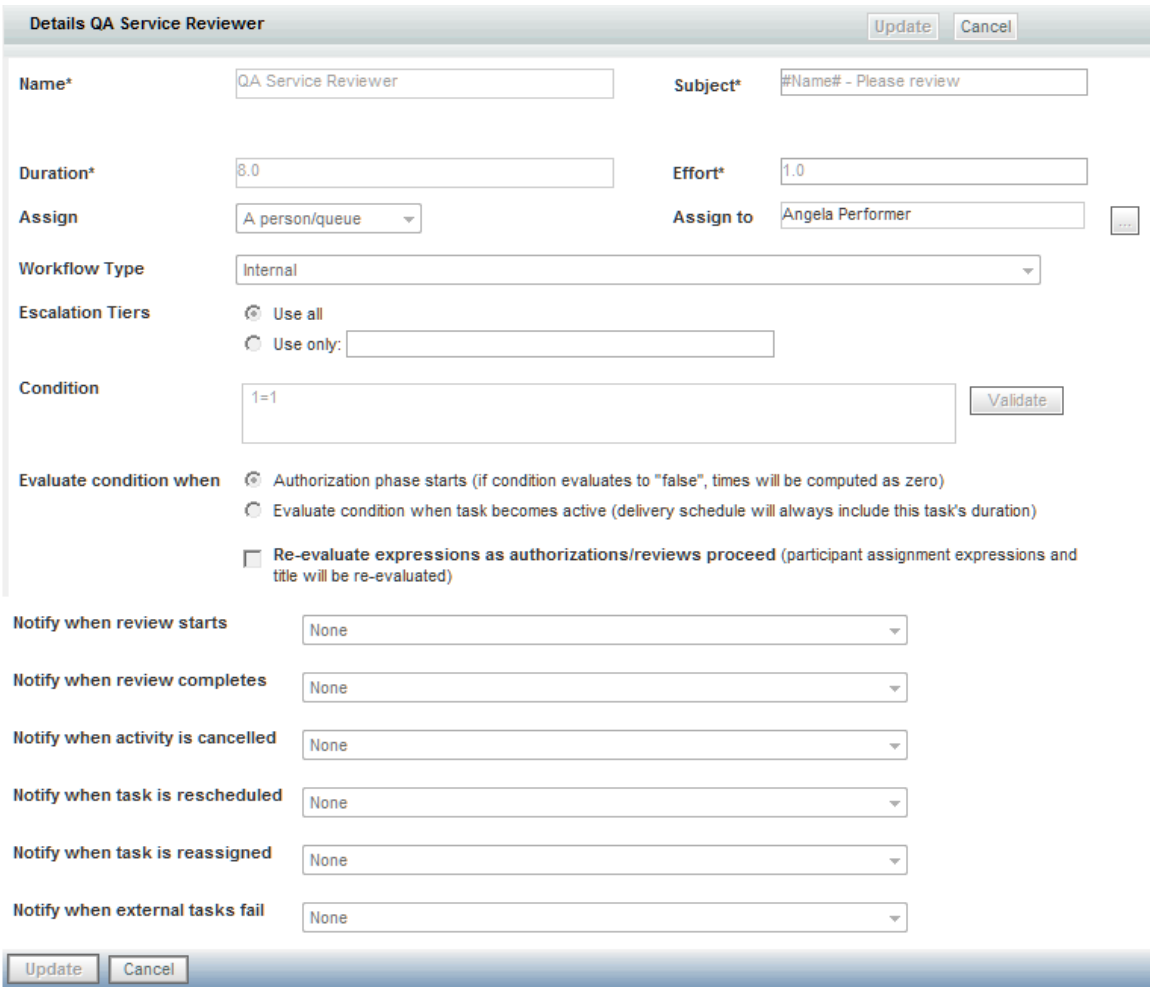
Step 3 Validate the condition by clicking **Validate**.

A Validate window appears. The message “unexpected token” indicates that the namespace used is not valid in this context or, perhaps, that you have forgotten to enclose an alphanumeric literal in quotes.

Step 4 Click **OK** to dismiss the Validate window.

Step 5 Click **Update** to save the Review/Authorization Details.

Figure 6-5 Review Details Window



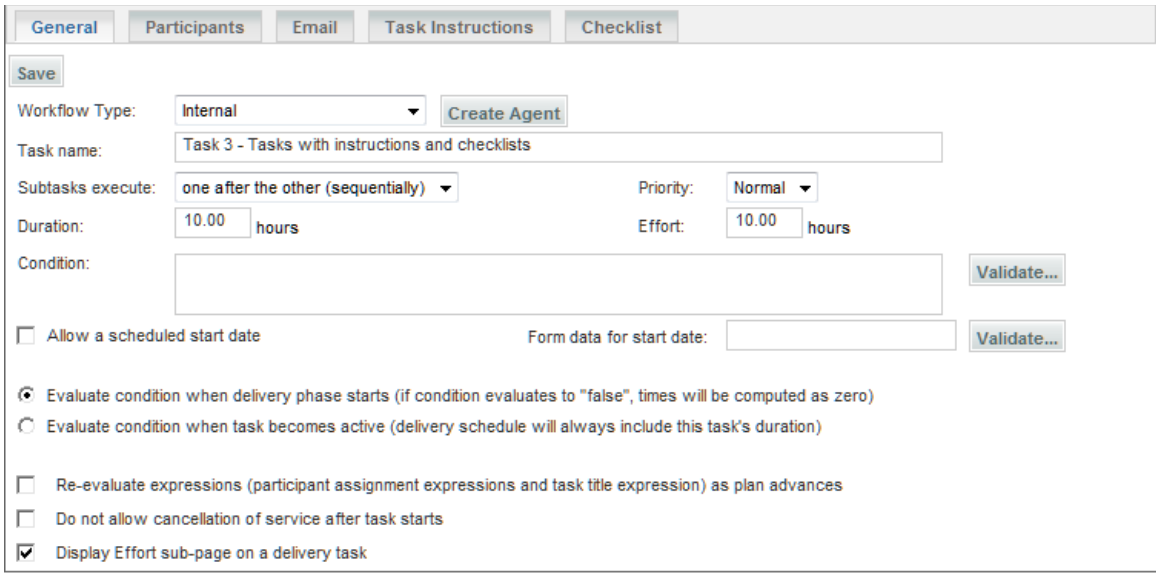
Validation checks that any namespace specified is valid for the current scope (the specific level of review/authorization or task), with the exception of dictionary fields (Data.**DictionaryName.FieldName**). This is perfectly legitimate, since the Review/Authorization may be defined at a site- or organization- level, independent of the service with which it is integrated. However, it may cause a runtime error: if the specified namespace, for example, Data.EUIT_ACCESS.Access_Type, does not exist.

Validation also checks that correct relational and arithmetic operators are used.

Conditional Delivery Plan Tasks

As with authorization and review tasks, a plan task in the service delivery moment may also be conditional.

Figure 6-6 Task Definition General Tab



Evaluating the Condition

Once the condition has been entered, you must decide when that statement will be evaluated. Each task (approval, review, or delivery) with a condition can be evaluated either

- At the beginning of a phase (Authorization or Delivery moment), or
- When the activity becomes active.

Evaluate condition when authorization/delivery phase starts

The designer can choose to evaluate a conditional statement when a **phase** starts by choosing the "*Evaluate condition when delivery phase starts (if condition evaluates to "false", times will be computed as zero)*" option within a specific task, as shown above for delivery tasks. A "phase" corresponds to any of the system moments defined for processing a requisition. Each authorization or review has its own moment; all delivery tasks are performed within the Service Delivery moment.

Authorization tasks are always serial. You could put one conditional on one task saying if field= "somevalue" and a conditional on another saying field<> "somevalue". That way, you know one authorization task will always be executed, and if you choose "*when authorization phase starts*", only one authorization task will appear in the process view. If you choose "when task becomes active", both tasks appear, but one would be skipped.

The "*if conditions evaluate to "false", times will be computed as zero"* statement means that Service Portal will evaluate the conditions at the beginning of the phase. If these conditions are false, then the corresponding tasks will not be executed, and the **Due Date** for the service will be calculated without including the duration of these tasks.

Evaluate condition when activity becomes active

Alternatively, the designer can choose to evaluate a conditional statement when the task starts by choosing the "*Evaluate condition when activity becomes active (times will not be affected, scheduling will be done by using these efforts)*" option.

Service Portal will evaluate the condition at the beginning of each task. If the condition is false, the corresponding task will not be executed. Durations for any task configured with this option will be used to calculate the due date upon submission.

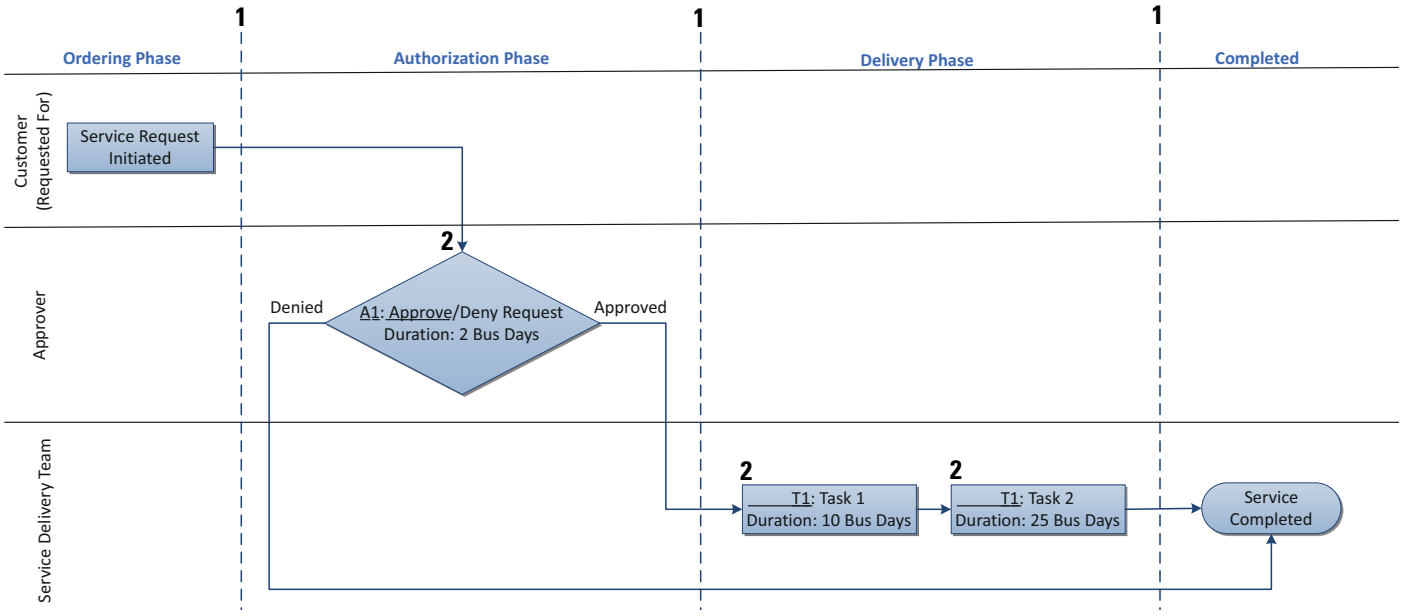
Re-evaluate Expressions as plan advances

The Re-evaluate Expressions feature is useful for designs in which there are multiple authorizations. It enables the person performing an authorization task to enter information in the service form which is then used to re-evaluate the expression used to assign the performer for a *subsequent* authorization task. If this option is not checked, all information used in expressions in the authorization task must be present during the Ordering moment.

This feature allows dynamic assignment of an approval or review task to a user (person or queue) and dynamic adjustment of the task title during the authorization or review moment. Once the authorization/review becomes active, the expression will be evaluated and the task will be assigned appropriately. This capability is available only for authorization and review tasks, not for delivery tasks.

Start of a Phase and Start of a Task

The following figure pinpoints and differentiates the start of a phase and the start of a task.



1	Start of Phase	2	Start of Task

Syntax for Conditional Statements

Conditional statements may include arithmetic operators as well as relational (logic) operators.

The type of logical operators you can use depend on the HTML representation of the field being tested. Most fields have s (input elements) that allow only one value to be assigned; these include text, text area, radio button, single-select (drop-down) list, and radio buttons. The following logical operators can be used in conditional statements applied to such fields:

Operator	Usage
<	Less than.
>	Greater than.
<=	Less than or equal to.
>=	Greater than or equal to.
<>	Not equal to.
=	Equal to.
OR	Performs the logical OR of two or more statements. The result is true if all of the statements are true and false otherwise; that is, if any of the statements are false.
AND	Performs the logical AND of two or more statements. The result is true if any of the statements is true and false otherwise.

Operators for conditionals have the standard order of operation:

```
-  (negative)
*  (multiplication), /  (division)
+  (addition),   -  (subtraction)
<, >, >=, <=, <> (not equal), =
NOT
OR
AND
```

Parentheses can be used to change the operator precedence or clarify the condition.

EXAMPLES:

```
ActivityID >= 50
Customer.FirstName = "Ann"
Requisition.ActualCost >= 2000
Service.Quantity * Service.PricePerUnit <= 1000
Data.Approver.Custom_2 = "VP" OR Data.Approver.Custom_2 = “Director”
```

The INCLUDES operator can only be applied to fields that can hold multiple values. These include multi-select (drop-down boxes) and checkboxes. The following logical operators can be used in conditional statements applied to such fields:

Operator	Usage
OR	Performs the logical OR of two or more statements. The result is true if all of the statements are true and false otherwise; that is, if any of the statements are false.
AND	Performs the logical AND of two or more statements. The result is true if any of the statements are true and false otherwise.
NOT	Negates the value of the condition.
INCLUDES	True if the specified value is a currently chosen option for the Namespace variable; applicable only to fields designated as “Multi-value” in the dictionary, typically fields with HTML representations of multi-select and checkbox.

EXAMPLE:

```
Data.EUIT_ACCESS.AccessType INCLUDES “DSL” AND NOT
Data.EUIT_ACCESS.AccessType INCLUDES “Dial-Up”
```

Notes:

- No “#” signs are used to enclose the Namespace.
- An alphanumeric value included in the condition must be enclosed within double quotation marks (“”).
- Namespace names are not case sensitive. The recommended standard is to use Title case (capitalizing the first letter of all words).
- All alphanumeric comparisons are case sensitive. For example, the condition “Data.MoveIndividual.FirstName=”Matt”” would be true only if the value of the FirstName field in the MoveIndividual dictionary were “Matt”, with an initial capital letter and the rest lower case letters.
- Any Boolean Namespaces (those whose names start with “Is”) have different values, depending on the underlying database. In SQLServer, the value of a Boolean is either true or false. In Oracle the corresponding values are 1 (for true) and 0 (for false).
- Less than and greater than operators for numeric value comparisons are not supported for dictionary fields. They are treated as text during comparisons. For example, the condition “Service.Data.VM.MemoryGB > 4” is evaluated to false if the dictionary field VM.MemoryGB has a value of 16.
- Multibyte characters are not supported for text string comparisons.
- An ampersand (“&”) included in a literal must be encoded as “&”. For example, the value “two & three” would appear in an expression as “two & three”.

Tips and Techniques

Use a condition that always evaluates to false (for example, “1=2”) in a conditional statement to specify a task that will automatically be skipped.

The most common use cases for this are:

- When a service needs to “autocomplete” without having any tasks completed. The skipped task will mark the end of the delivery plan, and the requisition is marked as complete.
- When an email needs to be sent without having to complete a task or when multiple emails are needed during a given moment in the following ways:
  - Create a parent task. On the email tab, choose an email to be sent out at completion (you may also configure one to go out when the activity becomes active).
  - Create a child task with the condition 1=2. Set the condition to evaluate when the activity becomes active.

Namespace Reference

This reference section lists all Namespaces and the contexts in which each can be used.

Namespace Objects and their Relationships

The following diagram illustrates the nodes in the Namespace and the relationships between the nodes. The type of a node determines not only its own properties, but also the subcontexts (other nodes) to which it provides access.

The labeled boxes in this figure represent the types of Namespace nodes. Labeled arrows show the Namespace elements that allow the properties of one node to be accessed from another node. This figure can serve as a guide for service designers and administrators who need to navigate the Namespace to get access to Service Portal data. For example, tracing the arrow labeled “Customer” from the Process node, we can see that the “Customer” element gives the Process node access to the properties of the Person node. So, for example, to access the Customer’s login name in a conditional statement for a task in a delivery plan, the condition would reference the namespace:

```
Requisition.Customer.LoginName
```

To access the same namespace from an email for a delivery plan, the namespace reference would be:

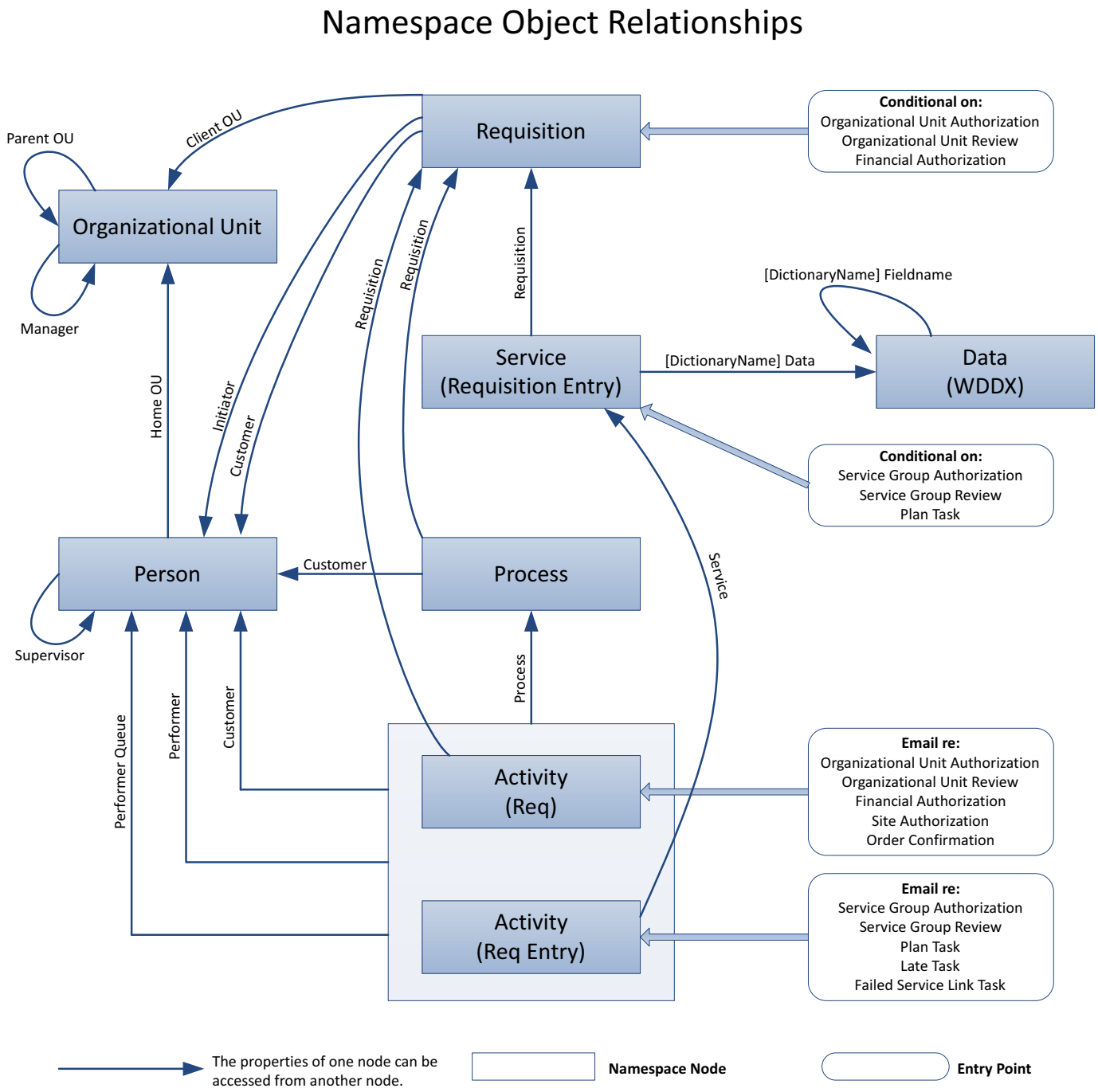
```
#Service.Requisition.Customer.LoginName#
```

Node relationships are not bidirectional. The fact that the Process node has access to the properties of the Person node does not imply that the properties of the Process node are also available to the Person node.

From the perspective of the service designer or administrator who wishes to use Service Portal data in formatted emails or conditional statements, there are multiple “entry points” to the paths between nodes:

- Activities associated with Requisition Entries (services), such as delivery plan tasks, Service Group Authorizations, Service group reviews, and emails generated by any of these activities.
- Activities associated with Requisitions, such as Departmental Authorizations, Departmental Reviews, and Financial Authorizations.

These entry points determine the Namespace context that applies when a service designer or administrator is configuring email to be sent from an activity, or defining a conditional execution statement. Think of them as the starting point for navigating the paths between nodes. Where you enter the Namespace determines what nodes, and ultimately what properties and data values, will be available to you from that particular activity.



Email Namespace Elements

The # character must be used to delimit Namespace elements in email notifications.

OU-based Authorizations and Reviews

The authorizations and reviews at an organizational level may trigger an email:

- Organizational Unit Authorization
- Organizational Unit Review
- Financial Authorization

These email entry points support the Namespace elements listed below:

```
#ActivityID#
#Priority# - priority assigned to the task: 1=high, 2=normal, 3=low
#DueOn#
#Subject# - the name of the task
#Waiting#
#ScheduledStart#
#StartedOn#
#CompletedOn#
#ExpectedDuration# - typical task duration, in hours
#ExpectedDurationUnits# - units in which task duration is displayed
#ActualDuration# - actual task duration, in hours
#CurrentDate#
#StateName# - the status of the task
#URL#
#Customer.*# - where * denotes any Customer element
#Performer.*# - where * denotes any Performer element
#PerformerQueue.*# where * denotes any PerformerQueue element
#Process.*# -- where * denotes any Process (task) element
#Requisition.*# - where * denotes any Requisition element
```

Tasks and Service Group Authorizations/Reviews

Tasks which are part of the delivery plan, as well as service group authorizations and reviews may trigger an email:

- Service Group Authorization
- Service Group Review
- Plan Task
- Late Task
- Ad-hoc Task
- Failed Service Link Task

These email entry points support the Namespace elements listed below:

```
#ActivityID#
#Priority#
#DueOn#
#Subject#
#Waiting#
#ScheduledStart#
#StartedOn#
#CompletedOn#
#ExpectedDuration#
#ActualDuration#
#Instructions# -- for Adhoc tasks only
#CurrentDate# -- the current date and time in GMT
#StateName# -- the current status (state) of the activity
#URL# -- the URL for directly accessing Task Details for this activity
```

```
#Customer.*# -- all Customer elements; for an adhoc task only, this
   refers to the service performer who created the task
#Performer.*# -- all Performer elements
#PerformerQueue.*# -- all Queue elements
#Process.*# - all Process elements
#Service.ServiceID#
#Service.ProcessTrackingID#
#Service.Quantity#
#Service.PricePerUnit#
#Service.HasPrice# -- false (0) if the service has no price, 1 (true)otherwise
#Service.Bundled# -- true (1)if the service is a child service in a
  bundle, false (0) otherwise
#Service.isBundle# -- true (1) if the service is itself a bundle, false
  (0) otherwise
#Service.BundledServices# -- the number of child services bundled with
  the current service
#Service.ServiceDefinition.Name#
#Service.ServiceDefinition.IsEntitlement#
#Service.ServiceDefinition.DescriptionURL#
#Service.ServiceDefinition.ExpectedDuration#
#Service.ServiceDefinition.FunctionalPosition.PersonTypeElement#
#Service.ServiceDefinition.ServiceGroup.Name#
#Service.ServiceDefinition.ServiceGroup.FunctionalPosition.PersonElement#
#Service.RequisitionEntryID#
#Service.Requisition.URL#
#Service.Requisition.ProcessTrackingID#
#Service.Requisition.ExpectedDuration#
#Service.Requisition.StartedDate#
#Service.Requisition.ActualCost#
#Service.Requisition.ExpectedCost#
#Service.Requisition.RequisitionID#
#Service.Requisition.Name#
#Service.Requisition.Customer.*# --all Customer elements
#Service.Requisition.Initiator.*# -- all Initiator element
#Service.Requisition.ClientOU.*# -- all ClientOU elements
#Service.Data.DictionaryName.FieldName#
```

The service data namespaces generally have the format

```
#Service.Data.DictionaryName.FieldName#
```

Conditional Namespace Elements

The # character is NOT used to access variables in conditionals. Some characters such as #, “, &, +, and – on the left side of the condition will cause database problems.

OU-based Authorizations and Reviews

The authorizations and reviews at an organizational level may be conditionally executed.

- Organizational Unit Authorization
- Organizational Unit Review
- Financial Authorization

These conditional entry points support the following Namespaces:

```
Site.URL
Customer.*
Initiator.*
ClientOU.*
```

Tasks and Service Group Authorizations/Reviews

Service group authorizations and reviews, as well as tasks which are part of the delivery plan allow conditions to determine whether the task/review is executed:

- Service Group Authorization
- Service Group Review
- Plan Task
- Late Task

These conditional entry points support the following Namespaces:

```
Requisition.*
Data.DictionaryName.FieldName
```

Organizational Unit-Based Namespaces

Information is available via Namespaces about organizational units. Organizational units may occur in many contexts, denoted by these namespace entity types:

Entity Type	Description	Example
ClientOU	Organizational unit of the client for the requisition	#Requisition.HomeOU. **OrgElementType#**
HomeOU	Home organizational unit of the performer, performer queue, or customer or of the manager of any of these people	#Performer.HomeOU. **OrgElementType#**
ParentOU	The parent organizational unit of the current OU	#Customer.HomeOU.ParentOU. **OrgElementType#**

Organizational Unit Element Types (OrgElementType) are listed below, in the context of the ClientOU.

```
#ClientOU.Name#
#ClientOU.Description#
#ClientOU.OrganizationalUnitID#
#ClientOU.OrganizationalUnitTypeID#
#ClientOU.CostCenterCode#
#ClientOU.FunctionalPosition.PersonElementType#
#ClientOU.Manager.PersonTypeElement#
#ClientOU.ParentOU.Name#
#ClientOU.ParentOU.Description#
#ClientOU.ParentOU.OrganizationalUnitID#
#ClientOU.ParentOU.OrganizationalUnitTypeID#
#ClientOU.ParentOU.CostCenterCode#
#ClientOU.ParentOU.FunctionalPosition.PersonElementType#
```

Person-Based Namespaces

The Customer, Initiator, Alternate, and Performer Namespace elements expose information about a person stored in Organization Designer. Therefore, all entity types support the same variables; only the element denoting the entity type (“Customer”, “Initiator”, “Alternate”, or “Person”) will vary.

A list of person-based Namespace nodes is given below. Namespaces are available for all basic and extended person attributes. To form a valid Namespace variable, use these as the last part of the Namespace name, where the first part is the entity type. As always, the Namespace must be enclosed in hash marks (#) when used in an email.

PersonType is one of:	
Alternate	The designated delegate for the current authorization task performer.
Customer	Generally, the person for whom the current requisition was ordered; for task-based emails and escalations, the Supervisor of the task performer; for ad-hoc tasks, the task performer who created the ad-hoc task.
Customer.Supervisor	The supervisor of the customer for the current order.
Customer.HomeOU.Manager	The manager of the home organizational unit of the customer for the current order.
Initiator	The person who ordered the current requisition.
Initiator.Supervisor	The supervisor of the person who ordered the current requisition.
Initiator.HomeOU.Manager	The manager of the home organizational unit of the person who ordered the current requisition.
Performer	The person responsible for performing the current task.
Performer.Supervisor	The supervisor of the task performer.
Performer.HomeOU.Manager	The manager of the home organizational unit of the task performer.
ClientOU.Manager	The manager of the organizational unit for which an organizational unit review or authorization is being performed.
FunctionalPosition	A Service Portal-defined functional position within an organization, service, or service group.
Position.**FunctionalPosition**	A user-defined functional position within an organization, service, or service group.

Elements of the Person-type namespace node are listed below. If no element is specified, the expression returns the person's unique identifier in the database.

```
#PersonType.FirstName#
#PersonType.LastName#
#PersonType.Title#
#PersonType.Birthdate#
#PersonType.Hiredate#
#PersonType.SSN#
#PersonType.EmployeeCode#
#PersonType.IsOffice#
#PersonType.TimeZoneID#
#PersonType.Email#
#PersonType.CompanyAddress# -- a concatenation of all lines of the company address,
including formatting into multiple lines
#PersonType.PersonalAddress# -- a concatenation of all lines of the personal address,
including formatting into multiple lines
#PersonType.SimpleCompanyAddress# -- a concatenation of all lines of the company
(business) address
#PersonType.SimplePersonalAddress# -- a concatenation of all lines of the personal address
#PersonType.DetailedCompanyAddress.Street1#
#PersonType.DetailedCompanyAddress.Street2#
#PersonType.DetailedCompanyAddress.City#
#PersonType.DetailedCompanyAddress.StateProvince#
```

#PersonType.DetailedCompanyAddress.Zip#
#PersonType.DetailedCompanyAddress.Country#
#PersonType.DetailedPersonalAddress.Street1#
#PersonType.DetailedPersonalAddress.Street2#
#PersonType.DetailedPersonalAddress.City#
#PersonType.DetailedPersonalAddress.StateProvince#
#PersonType.DetailedPersonalAddress.Zip#
#PersonType.DetailedPersonalAddress.Country#
#PersonType.Location# - a concatenation of all aspects of the location, with formatting to place elements on separate lines
#PersonType.DetailedLocation.Building#
#PersonType.DetailedLocation.BuildingLevel#
#PersonType.DetailedLocation.Office#
#PersonType.DetailedLocation.Cubicle#
#PersonType.SimpleLocation# -- a concatenation of all aspects of the location, with spaces separating the elements
#PersonType.WorkPhone#
#PersonType.HomePhone#
#PersonType.Fax#
#PersonType.Mobile#
#PersonType.Pager#
#PersonType.LoginName#
#PersonType.TimeZone#
#PersonType.ExtManager#
#PersonType.CompanyCode#
#PersonType.Division#
#PersonType.BusinessUnit#
#PersonType.DepartmentNumber#
#PersonType.CostCenter#
#PersonType.ManagementLevel#
#PersonType.Region#
#PersonType.EmployeeType#
#PersonType.LocationCode#
#PersonType.Custom1#
#PersonType.Custom2#
#PersonType.Custom3#
#PersonType.Custom4#
#PersonType.Custom5#
#PersonType.Custom6#
#PersonType.Custom7#
#PersonType.Custom8#
#PersonType.Custom9#
#PersonType.Custom10#

EXAMPLES:

```
#Person.FirstName# #Person.LastName#
#Customer.Supervisor.Fax#
```

Customer Namespaces

In delivery tasks, including escalations, the customer is the Task Supervisor. For ad-hoc tasks, the customer is the task performer who initiated the ad-hoc task. For all other tasks, and for a requisition, the customer is the person for whom the service has been requested. Customer namespaces allow access to customer information, including:

- Customer (person) information as defined in the person's profile and accessible via Organization Designer > People
- All person/profile information defined for the customer’s supervisor
- Information about the home Organizational Unit of the customer

- All person/profile information defined for the manager of the customer's home OU

The namespaces below must be enclosed in hash marks (#) when used in emails, but entered without the hash marks when used in expressions. A complete list of person-type namespaces are listed in the "Person-Based Namespaces" section on page 6-27.

Customer Namespaces.

```
#Customer.PersonTypeElement#
#Customer.Supervisor.PersonTypeElement#
#Customer.HomeOU.Name#
#Customer.HomeOU.OrganizationalUnitID#
#Customer.HomeOU.OrganizationalUnitTypeID# -- the type of unit, where 1=service team, and
2=business unit
#Customer.HomeOU.CostCenterCode#
#Customer.HomeOU.Manager.PersonTypeElement#
#Customer.HomeOU.ParentOU.Name#
#Customer.HomeOU.ParentOU.OrganizationalUnitID#
#Customer.HomeOU.ParentOU.OrganizationalUnitTypeID# -- the type of unit, where 1=service
team, and 2=business unit
#Customer.HomeOU.ParentOU.CostCenterCode#
```

Performer Namespaces

The performer is the person responsible for performing a task in the service's delivery plan. Performer namespaces are not available in context in which no task is current—these include the email for Organization Unit reviews and authorizations; and financial authorizations.

Performer namespaces allow access to performer information, including:

- Performer (person) information as defined in the person's profile and accessible via Organization Designer > People
- All person/profile information defined for the performer's supervisor
- Information about the home Organizational Unit of the performer
- All person/profile information defined for the manager of the performer's home OU

The namespaces below must be enclosed in hash marks (#) when used in emails, but entered without the hash marks when used in expressions. A complete list of person-type namespaces are listed in the "Person-Based Namespaces" section on page 6-27.

```
#Performer.PersonTypeElement#
#Performer.Supervisor.PersonTypeElement#
#Performer.HomeOU.Name#
#Performer.HomeOU.OrganizationalUnitID#
#Performer.HomeOU.OrganizationalUnitTypeID#
#Performer.HomeOU.CostCenterCode#
#Performer.HomeOU.Manager.PersonTypeElement#
```

PeformerQueue Namespaces

Performer Queue namespaces provide information about the queue to which a review, authorization, or task was assigned. A subset of the Person-based namespace elements and properties are meaningful—those which are exposed in the user interface for maintaining queues in Organization Designer.

```
#PerformerQueue.FirstName#
#PerformerQueue.LastName#
#PerformerQueue.TimeZoneID#
#PerformerQueue.Email#
```

```
#PerformerQueue.WorkPhone#
#PerformerQueue.HomePhone#
#PerformerQueue.Fax#
#PerformerQueue.Mobile#
#PerformerQueue.Pager#
#PerformerQueue.TimeZone#
#PerformerQueue.HomeOU.Name#
#PerformerQueue.HomeOU.OrganizationalUnitID#
#PerformerQueue.HomeOU.OrganizationalUnitTypeID#
#PerformerQueue.HomeOU.CostCenterCode#
#PerformerQueue.HomeOU.ParentOU.Name#
#PerformerQueue.HomeOU.ParentOU.OrganizationalUnitID#
#PerformerQueue.HomeOU.ParentOU.OrganizationalUnitTypeID#
#PerformerQueue.HomeOU.ParentOU.CostCenterCode#
#PerformerQueue.HomeOU.Manager.PersonTypeElement#
```

Initiator Namespaces

The initiator is the person who orders a service.

Initiator namespaces allow access to initiator information, including:

- Initiator (person) information as defined in the person's profile and accessible via Organization Designer > People
- All person/profile information defined for the initiator's supervisor
- Information about the home Organizational Unit of the initiator
- All person/profile information defined for the manager of the initiator's home OU

The namespaces below must be enclosed in hash marks (#) when used in emails, but entered without the hash marks when used in expressions. A complete list of person-type namespaces are listed in the "Person-Based Namespaces" section on page 6-27.

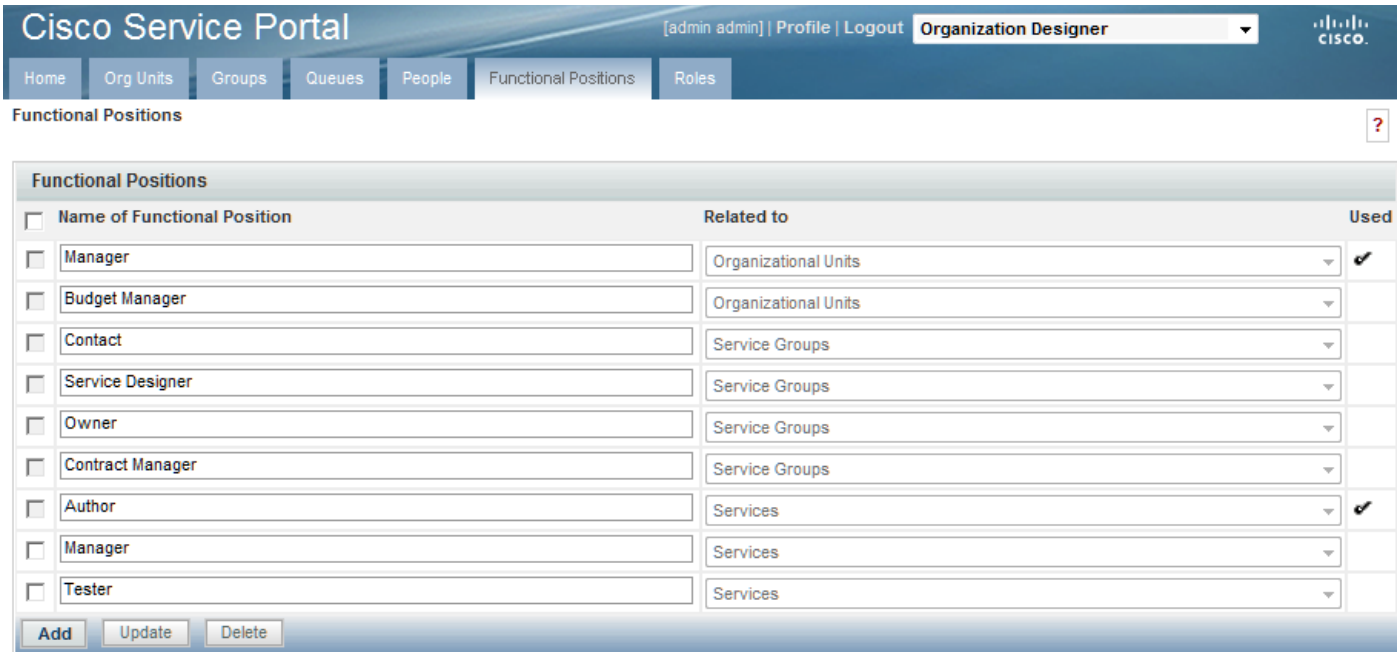
```
#Initiator.PersonTypeElement#
#Initiator.Supervisor.PersonTypeElement#
#Initiator.HomeOU.Name#
#Initiator.HomeOU.OrganizationalUnitID#
#Initiator.HomeOU.OrganizationalUnitTypeID#
#Initiator.HomeOU.CostCenterCode#
#Initiator.HomeOU.Manager.PersonTypeElement#
#Initiator.HomeOU.ParentOU.Name#
#Initiator.HomeOU.ParentOU.OrganizationalUnitID#
#Initiator.HomeOU.ParentOU.OrganizationalUnitTypeID#
#Initiator.HomeOU.ParentOU.CostCenterCode#
```

Functional Positions

Functional positions allow you to access the Person information for individuals who have been assigned to these positions. You can access this information both for the default functional positions, and for those that have been configured for a particular implementation.

Functional positions are defined through the module selection Organization Designer > Functional Positions. Each functional position is associated with a particular entity—this may be a service, a service group, or an organizational unit.

Figure 6-7 Functional Positions



Once the position has been defined, you may assign a person to the position through the Positions page for the organizational unit, or on the General tab for the service or service group. Some sample usages include:

Use **Requisition.Customer.HomeOU.BudgetManager.Emai**l to access the email address of the Budget Manager of the customer's home organizational unit.

Use **Performer.HomeOU.ParentOU.Manager.FirstName** to access the first name of the Manager of the parent organizational unit of the task performer's home organizational unit.

Default functional positions may have spaces in the position name, for example, "Budget Manager". The space is omitted with the functional position is used within a namespace reference.

User-specified functional positions may not include spaces. The namespace reference must prefix the functional position with the keyword "Position" as in the following example:

```
#Service.ServiceDefinition.Position.EscalationManager.Email#
```

For all of the functional positions, you have access to all the variables defined in the Person table above. If no person variable is included, the expression returns the person's database ID.

You can access information about Home OU, Parent OU, or Client OU functional positions from any person role that you can access in either the Requisition or the Service context. Information about Service and Service Group functional positions is only available in the Service context.

When using these expressions to include dynamic data in emails and task names, you must add the pound separator ('#') at the beginning and end of the expression, for example:

```
#Customer.HomeOU.Manager.Email#
```

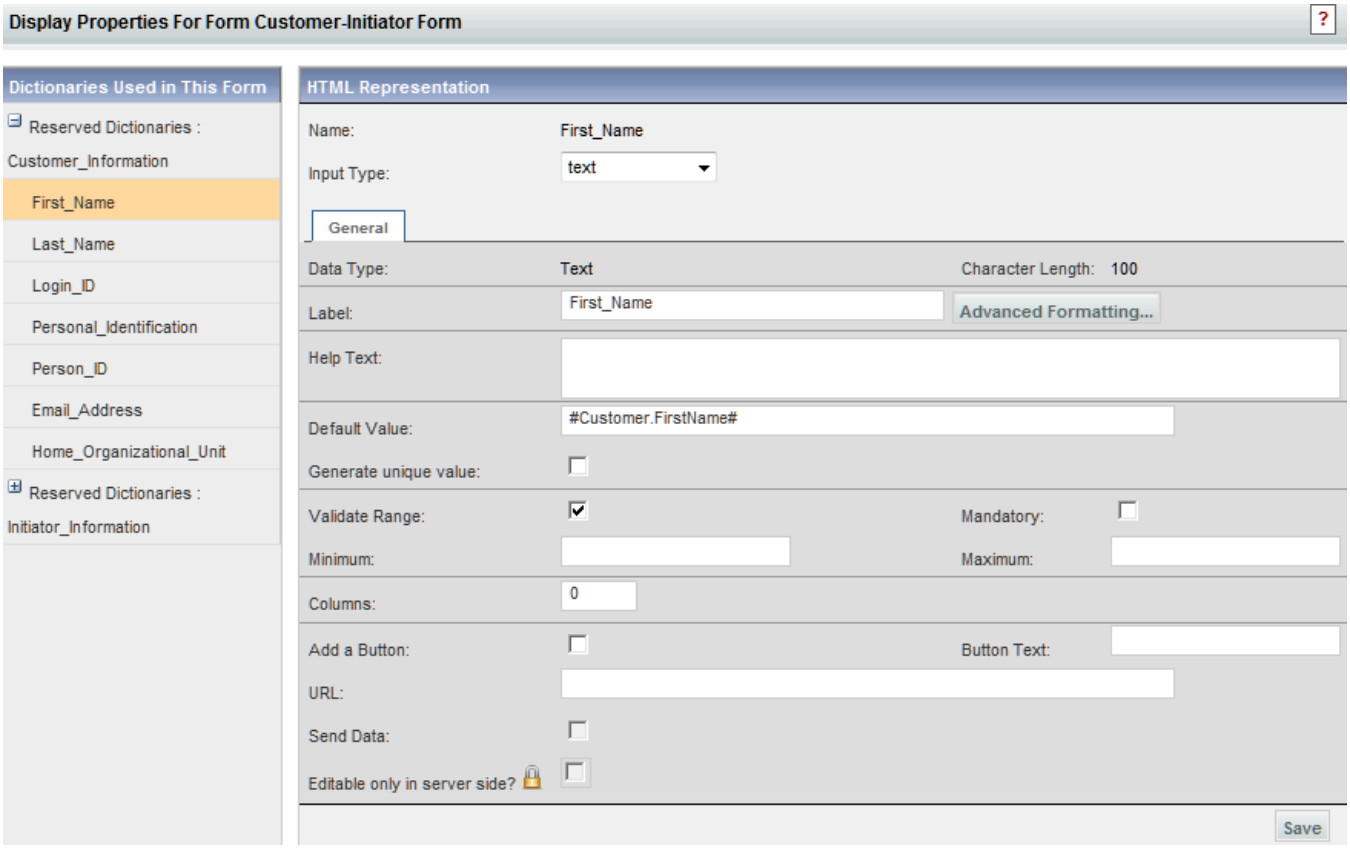
Lightweight Namespaces

Active form rules need the equivalent of namespaces in order to dynamically access field values to be used or evaluated. For example, the service may need to display an additional dictionary or field if the user entered “Other” in a previous field; the current customer's organization may need to be used as the criteria for building a drop-down list to display valid locations for a service delivery; default values need to be provided for customer and initiator data.

Lightweight namespaces provide these capabilities. They are “lightweight” since only the information accessible to the form (not, for example, details about the service's delivery plan or task performers) can be used within the rules.

Any forms that include person-based dictionaries use lightweight namespaces to provide the values to the form fields, based on corresponding values in fields stored in the profile for the selected person. This includes both the Customer-Initiator form and any user-defined forms. Lightweight namespaces have the format #Customer.**FieldName**#, or #Initiator.**FieldName**#.

Note The use of grid dictionary fields for lightweight namespaces is not supported.



The namespace is automatically supplied as the Default Value for the field. If desired, the initial assignments can be replaced.

Lightweight namespaces referring to customer or initiator data can also be used as default values for fields in dictionaries that are not person-based. In this case the service designer will, of course, be responsible for mapping from the dictionary field to the appropriate person attribute. This capability allows you to define dictionaries that contain both person-based and other data.

Customer-Based Namespaces

Customer-based namespaces used for the Customer Information reserved dictionary are summarized below.

Dictionary Field Name	Field Type	Lightweight Namespace
First_Name	Text	#Customer.FirstName#
Last_Name	Text	#Customer.LastName#
Login_ID	Text	#Customer.LoginID#
Person_ID	Number	#Customer.PersonID#
Personal_Identification	Text	#Customer.PersonIdentification#
Email_Address	Text	#Customer.Email#
Home_Organizational_Unit	Text	#Customer.HomeOU.Name#
Title	Text	#Customer.Title#
Social_Security_Number	Text	#Customer.SSN#
Birthdate	Date	#Customer.Birthdate#
Hiredate	Date	#Customer.Hiredate#
Timezone	Text	#Customer.TimeZoneID#
Locale	Text	#Customer.LocaleID#
Supervisor	Text	#Customer.Supervisor.Name#
Employee_Code	Text	#Customer.EmployeeCode#
Supervisor_Email	Text	#Customer.Supervisor.Email#
Supervisor_ID	Number	#Customer.Supervisor.PersonID#
Supervisor_Phone	Text	#Customer.Supervisor.Phone#
Notes	Text	#Customer.Notes#
Company_Street_1	Text	#Customer.DetailedCompanyAddress.Street1#
Company_Street_2	Text	#Customer.DetailedCompanyAddress.Street2#
Company_City	Text	#Customer.DetailedCompanyAddress.City#
Company_State	Text	#Customer.DetailedCompanyAddress.StateProvince#
Company_Country	Text	#Customer.DetailedCompanyAddress.Country#
Company_Postal_Code	Text	#Customer.DetailedCompanyAddress.Zip#
Building	Text	#Customer.DetailedLocation.Building#
Level	Text	#Customer.DetailedLocation.BuildingLevel#
Office	Text	#Customer.DetailedLocation.Office#
Cubicle	Text	#Customer.DetailedLocation.Cubicle#

Dictionary Field Name	Field Type	Lightweight Namespace
Personal_Street_1	Text	#Customer.DetailedPersonalAddress.Street1#
Personal_Street2	Text	#Customer.DetailedPersonalAddress.Street2#
Personal_City	Text	#Customer.DetailedPersonalAddress.City#
Personal_State	Text	#Customer.DetailedPersonalAddress.StateProvince#
Personal_Country	Text	#Customer.DetailedPersonalAddress.Country#
Personal_Postal_Code	Text	#Customer.DetailedPersonalAddress.Zip#
Work_Phone	Text	#Customer.WorkPhone#
Home_Phone	Text	#Customer.HomePhone#
Fax	Text	#Customer.Fax#
Mobile_Phone	Text	#Customer.Mobile#
Pager	Text	#Customer.Pager#
Other	Text	#Customer.OtherPhone#
Main_Phone	Text	#Customer.MainPhone#
Primary_Phone	Text	#Customer.PrimaryPhone#
Primary_Fax	Text	#Customer.PrimaryFax#
Sales_Phone	Text	#Customer.SalesPhone#
Support_Phone	Text	#Customer.SupportPhone#
Billing_Phone	Text	#Customer.BillingPhone#
Other_Contact_Information	Text	#Customer.OtherContactInfo#
Company_Code	Text	#Customer.CompanyCode#
Division	Text	#Customer.Division#
Business_Unit	Text	#Customer.BusinessUnit#
Department_Number	Text	#Customer.DepartmentNumber#
Cost_Center	Text	#Customer.CostCenter#
Management_Level	Text	#Customer.ManagementLevel#
Region	Text	#Customer.Region#
Employee_Type	Text	#Customer.EmployeeType#
Custom_1	Text	#Customer.Custom1#
Location_Code	Text	#Customer.LocationCode#
Custom_2	Text	#Customer.Custom2#
Custom_3	Text	#Customer.Custom3#
Custom_4	Text	#Customer.Custom4#
Custom_5	Text	#Customer.Custom5#
Custom_6	Text	#Customer.Custom6#
Custom_7	Text	#Customer.Custom7#
Custom_8	Text	#Customer.Custom8#

Dictionary Field Name	Field Type	Lightweight Namespace
Custom_9	Text	#Customer.Custom9#
Custom_10	Text	#Customer.Custom10#

Initiator-Based Namespaces

Initiator-based namespaces used for the Initiator Information reserved dictionary are summarized below.

Dictionary Field Name	Field Type	Lightweight Namespace
First_Name	Text	#Initiator.FirstName#
Last_Name	Text	#Initiator.LastName#
Login_ID	Text	#Initiator.LoginID#
Person_ID	Number	#Initiator.PersonID#
Email_Address	Text	#Initiator.Email#
Personal_Identification	Text	#Initiator.PersonIdentification#
Home_Organizational_Unit	Text	#Initiator.HomeOU.Name#
Title	Text	#Initiator.Title#
Social_Security_Number	Text	#Initiator.SSN#
Birthdate	Date	#Initiator.Birthdate#
Hiredate	Date	#Initiator.Hiredate#
Timezone	Text	#Initiator.TimeZoneID#
Locale	Text	#Initiator.LocaleID#
Employee_Code	Text	#Initiator.EmployeeCode#
Supervisor	Text	#Initiator.Supervisor.Name#
Supervisor_ID	Number	#Initiator.Supervisor.ID#
Supervisor_Phone	Text	#Initiator.Supervisor.Phone#
Supervisor_Email	Text	#Initiator.Supervisor.Email#
Notes	Text	#Initiator.Notes#
Company_Street_1	Text	#Initiator.DetailedCompanyAddress.Street1#
Company_Street_2	Text	#Initiator.DetailedCompanyAddress.Street2#
Company_City	Text	#Initiator.DetailedCompanyAddress.City#
Company_State	Text	#Initiator.DetailedCompanyAddress.StateProvince#
Company_Country	Text	#Initiator.DetailedCompanyAddress.Country#
Company_Postal_Code	Text	#Initiator.DetailedCompanyAddress.Zip#
Building	Text	#Initiator.DetailedLocation.Building#
Level	Text	#Initiator.DetailedLocation.BuildingLevel#
Office	Text	#Initiator.DetailedLocation.Office#
Cubicle	Text	#Initiator.DetailedLocation.Cubicle#
Personal_Street_1	Text	#Initiator.DetailedPersonalAddress.Street1#

Dictionary Field Name	Field Type	Lightweight Namespace
Personal_Street2	Text	#Initiator.DetailedPersonalAddress.Street2#
Personal_City	Text	#Initiator.DetailedPersonalAddress.City#
Personal_State	Text	#Initiator.DetailedPersonalAddress.StateProvince#
Personal_Country	Text	#Initiator.DetailedPersonalAddress.Country#
Personal_Postal_Code	Text	#Initiator.DetailedPersonalAddress.Zip#
Work_Phone	Text	#Initiator.WorkPhone#
Home_Phone	Text	#Initiator.HomePhone#
Fax	Text	#Initiator.Fax#
Mobile_Phone	Text	#Initiator.Mobile#
Pager	Text	#Initiator.Pager#
Other	Text	#Initiator.OtherPhone#
Main_Phone	Text	#Initiator.MainPhone#
Primary_Phone	Text	#Initiator.PrimaryPhone#
Primary_Fax	Text	#Initiator.PrimaryFax#
Sales_Phone	Text	#Initiator.SalesPhone#
Support_Phone	Text	#Initiator.SupportPhone#
Billing_Phone	Text	#Initiator.BillingPhone#
Other_Contact_Information	Text	#Initiator.OtherContactInfo#
Company_Code	Text	#Initiator.CompanyCode#
Division	Text	#Initiator.Division#
Business_Unit	Text	#Initiator.BusinessUnit#
Department_Number	Text	#Initiator.DepartmentNumber#
Cost_Center	Text	#Initiator.CostCenter#
Management_Level	Text	#Initiator.ManagementLevel#
Region	Text	#Initiator.Region#
Employee_Type	Text	#Initiator.EmployeeType#
Location_Code	Text	#Initiator.LocationCode#
Custom_1	Text	#Initiator.Custom1#
Custom_2	Text	#Initiator.Custom2#
Custom_3	Text	#Initiator.Custom3#
Custom_4	Text	#Initiator.Custom4#
Custom_5	Text	#Initiator.Custom5#
Custom_6	Text	#Initiator.Custom6#
Custom_7	Text	#Initiator.Custom7#
Custom_8	Text	#Initiator.Custom8#
Custom_9	Text	#Initiator.Custom9#
Custom_10	Text	#Initiator.Custom10#

Process Namespaces

Process namespaces are available only for use in emails. They cannot be used in conditions. The Process object includes all namespaces regarding the Customer and Requisition. The Process refers to the current task.

```
#Process.Name#
#Process.Status#
#Process.StatusID#
#Process.StartedOn#
#Process.CompletedOn#
#Process.ExpectedDuration#
#Process.ActualDuration#
#Process.CostCenterID#
#Process.DueOn#
#Process.EscalationLevel#
#Process.TicketID#
#Process.TicketObjectID#
#Process.DueOnTZ#
#Process.StartedOnTZ#
#Process.DateNow# -- the current date and time in GMT
#Process.Customer.*# -- any Customer element
#Process.Requisition.*# -- any Requisition element
```

Requisition Namespaces

```
#Requisition.URL#
#Requisition.ProcessTrackingID#
#Requisition.ExpectedDuration#
#Requisition.StartedDate#
#Requisition.ActualCost#
#Requisition.ExpectedCost#
#Requisition.RequisitionID#
#Requisition.Name#
#Requisition.Services# -- The number of requisition entries in the requisition
#Requisition.Customer.*# -- Any Customer element
#Requisition.ClientOU.*# -- Any ClientOU element
#Requisition.Initiator.*# -- Any Initiator element
```

Message Namespaces

Message namespaces are available only for use in emails generated as a result of a failed Service Link task. They cannot be used in any other emails or in any conditions. The Message elements may be helpful in diagnosing the Service Link failure, and may eliminate having to consult the log files for diagnostics.

```
#Message.Error.Text# -- The error text as it would appear in the Service Link adapter or
application server log file
#Message.Error.Stack#
#Message.Error.StackHtml#
#Message.ChannelID#
#Message.Agent.Name#
#Message.Agent.Action#
#Message.Agent.Description#
#Message.NewscaleContent# -- Complete newScale XML message
#Message.ExternalContent# -- Complete external message, after any transformation has been
applied
```

Demand Center Namespaces

These namespaces should be enclosed in hash marks (#) when used in an email template.

BusinessService Namespaces

```
Agreement.BusinessService.Name
Agreement.BusinessService.Description
Agreement.BusinessService.PriceDescription
Agreement.BusinessService.ServiceLevelDescription
Agreement.BusinessService.IncludedServicesDesc
Agreement.BusinessService.FiscalYear
Agreement.BusinessService.CreationDate
Agreement.BusinessService.expirationdate
Agreement.BusinessService.topHTML
Agreement.BusinessService.bottomHTML
```

Agreement Namespaces

```
Agreement.Name
Agreement.Id – the unique identifier for the agreement **
Agreement.ownerName – the creator of the agreement **
Agreement.startDate
Agreement.expirationDate
Agreement.Performer.Name – the name of the person who in ii tat es or proposes the
revision
Agreement.Performer.Email - the email address of the person who in ii tat es or proposes
the revision
Agreement.Stakeholders - a list of email addresses representing anyone interested in a
specific agreement
```

** - Available in versions 2008.2 and above

Account Namespaces

```
Agreement.Account.Name
Agreement.Account.DateCreated
Agreement.Account.Description
```

Other Demand Center Namespaces

```
RelationshipManager.Email - the email address of the person who created the agreement
AccountOwner.Email - the email address of the person responsible for the account for which
the agreement is associated
```

INDEX

A

Access Control 2-27 to 2-30
Accounts 4-45
Actions 2-70
  Supported in Grids 2-24
Active Form Behavior 2-75 to 2-77
Active Form Components 1-1
  Best Practices 2-111 to 2-115
  Common Uses 2-2
  Configuring 3-27 to 3-29
  Defined 1-4
  Forms 2-12
  Using in a Service 1-71
Active Form Rules 2-31
  Active Form Behavior 2-75 to 2-77
  Conditional Rules 2-66 to 2-74
  Data Retrieval Rules 2-33 to 2-43
  Server-Side Events 2-31, 2-32, 2-35, 2-74, 2-76
Additional URL 1-15
Advanced Formatting Button 2-16, 2-21
Agents 4-40
Agreements 4-41
API. See Application Programming Interface (API).
Application Programming Interface (API) 2-2
  ISF 2-82
Approvals Portlet 4-14
Arguments, Adding to JavaScript Functions 2-94
Assignment Types 6-3
Associated Controls 2-103
  Server-Side 2-125
Associated Entity 5-3
Associated Services Subtab 3-18
Asynchronous Submission 1-16
Attribute Type 3-9
Authentication
  Automatic 4-34
  Settings 4-32
Authorizations 1-10, 4-42, 6-12
  Defined 1-4
  Namespace Usage 6-12
  Types 1-56
Authorizations Tab 1-9, 1-54 to 1-59
Automatic Retrieval, Service Item Instance Data 2-57, 3-28

B

Branded Content Libraries 5-46
  Deploying ?? to 5-44, 5-46 to 5-49
Bundle 1-19, 1-62
  Creating 1-63
  Discounting the Price 1-66
  Importing and Exporting 1-67
  Included Tasks 1-64
  Namespace Varibles 1-68
  Preventing 1-64
  Pricing 1-66
Business Engine 6-1
Buttons 2-103

C

Capacity Management 3-68
Case Analysis 2-116
Catalog Deployer
- Adding Content to a Deployment Package 5-25
- Advanced Services Deployment Type 5-35
- Assembling a Deployment Package 5-27
- Basic Services Deployment Type 5-34
- Closing/Reopening Deployment Packages 5-33
- Configuring 5-9
- Copying Deployment Packages 5-33
- Creating a Branded Libraryt Package 5-48
- Creating a Deployment Package 5-24
- Custom Deployment Type 5-36
- Data Source Configuration 5-14
- Defined 5-1, 5-3
- Deleting Deployment Packages 5-33
- Deploying a Library 5-49
- Deploying Deployment Packages 5-30
- Deployment Types 5-19
- Exporting Deployment Packages 5-28
- Hiding/Adjusting the View and Search Pane 5-20
- Importing a Library Package 5-47
- Importing Deployment Packages 5-29
- Operation 5-1
- Panes 5-20
- Performance Considerations 5-17
- Previewing Deployment Packages 5-25
- Sample Deployment Scenarios 5-39
- Search Packages 5-22
- Supported Entities 5-8
- Terminology 5-3
- Transmit and Deploy Mulitiple Packages 5-31
- Transmitting a Deployment Package 5-27
- Using 5-19
- View Log Files 5-37

Categories 4-39
- Configuring 1-77
- Defined 1-75
- Defining 1-76
- Display Options 1-78
- Removing 1-79

Checklist Subtab 1-54
Child Service 1-45, 1-62 to 1-68
CMDBs. See Configuration Management Databases.
Coding Standards 2-106
Common Settings 4-31
Component Entity, Defined 5-3
Conditional Authorization and Review Tasks 6-17
Conditional Delivery Plan Tasks 6-18
Conditional Namespace Elements 6-26
Conditional Rules 2-66 to 2-74
- Actions 2-70
- Adding New 2-62
- Applied to Grids 2-24
- Conditions 2-67
- Defined 2-1

Conditional Statements 6-17 to 6-22
- Operators 6-21

Conditions 1-41
Configuration Management 5-5
Configuration Management Databases (CMDBs) 3-1
Configuration Management Tools 5-16
Content Definition 4-37
Content Portlet 4-6 to 4-12
Core Entities 4-7, 4-36, 4-37
Cost Details 1-21
Custom Content 4-15 to 4-17
Customer
- Defined 1-5
- Dictionary 1-70

Customer-Based Namespaces 6-34
Customer-Initiator Form 2-26
Customer Namespaces 6-29

D

Database Administration 3-71
Database Table Lookup 2-36
Data Retrieval Rules 2-33 to 2-43
  Adding New 2-33, 2-49, 2-60
  Defined 2-1
  Distributing Rule 2-34, 2-80
  Distributing Vs. Validating 2-80
  Example 2-43, 2-58
  Performance 2-81
  Validating Rule 2-34, 2-39
Data Security 2-121
Datasource 2-33
Data Type 2-7
Date and Time 2-7
  Grid 2-24
Defining Standards 3-19 to 3-21
Delivery Plan 6-14
  Configuring 1-28 to 1-62, 3-29 to 3-43
Delivery Tasks 1-31
  Namespace Usage 6-15 to 6-17
Demand Center
  Namespaces 6-39
  Templates 6-10
Deployment Packages
  Adding Content 5-25
  Advanced Services Type 5-35
  Assembling 5-27
  Basic Services Type 5-34
  Closing/Reopening 5-33
  Copying 5-33
  Creating 5-24
  Cutom Type 5-36
  Defined 5-3
  Deleting 5-33
  Deploying 5-30
  Deploying Multiple Packages 5-32
  Exporting 5-28
  Importing 5-29
  Previewing 5-25
  Searching 5-22
  Transmitting 5-27
  Transmitting Multiple Packages 5-31
Deployment Types 5-19
Design Guidelines 2-110
Design Service Items Tab 3-4, 3-6
Design Standards Tab 3-19, 3-20
Dictionary 1-68
  Adding New 2-45
  Configuring Service Items 3-22 to 3-27
  Create Internal 2-5
  Customer and Initiator 1-70, 2-9
  Data Type 2-7
  Defined 1-1, 1-5, 2-1, 2-4
  Display as Grid 2-19, 2-21, 2-47
  Free-Form 1-69, 2-5
  Hiding 2-121
  Integration 2-11
  Person-Based 1-69, 1-70, 2-7
  Reserved 1-68, 1-70, 2-9
  Selecting Fields to Use 2-11
  Service Item-Based 1-71
  Setting Fields to Display in a Grid 2-18
  Type 2-5
Dictionary-Level Functions 2-85
  Usage in Grids 2-25
Directory Task 1-37
  Configuring 1-38
Display as Grid 2-19, 2-21, 2-47, 2-59
  Grid Options 2-21
Display Properties 2-13
Distributing Rule 2-34, 2-80
Document Type Definition (DTD) 3-65
DTD. See Document Type Definition (DTD).
Due Dates, Forecasting 1-15
Dynamic Pricing 1-23

E

Editable on Server-Side Only 2-15, 2-79
Email Namespace Elements 6-24
Email Subtab 1-52
Email Template
  Defined 1-5
  Defining 6-5
  Namespace Usage 6-7
  Previewing 6-7
Encryption 2-121
Entitlement 1-14
Entity, Defined 5-3
Entity Homes 5-12
  Defined 5-3
Error Messages, VMware Adapter 3-73
Escalation Manager 1-11
Escalations 1-10
  Defined 1-5
  Subtab 1-29
Export Portal Pages 4-54
Export Portlets 4-53
Expressions 6-3
  Configuring 6-3
  Validating 1-41
Extended Person Field 1-70
External Sites, Edit Passwords 4-48
External Tasks 1-36

F

Field-Level Functions 2-86 to 2-90
  Specialized 2-90 to 2-92
  Usage in Grids 2-25
Fields
  Generate Unique ID Value 2-15
  Hiding 2-121
  Make Read-Only 2-119
  Person-Based 2-91
Filter, Portlet 4-10
Forecasting Due Dates 1-14, 1-15
Form
  Access Control 2-27 to 2-30
  Active Form Rules 2-31
  Adding to a Service 1-71
  Appearance 2-15
  Components 2-12
  Customer-Initiator 2-26
  Defined 2-2
  Display Properties 2-13
  Form Content 2-12
  HTML Representation 2-13
  Tab 1-71
Form Content Tab 2-12
  Advanced Formatting Button 2-16, 2-21
Free-Form Dictionary 1-69, 2-5
Functional Position 1-15
  Adding to a Service Group 1-9
  Defined 1-8
Functions
  Adding Arguments 2-94
  Adding to a Form 2-97
  Dictionary-Level 2-85
  Field-Level 2-86 to 2-90
  JavaScript 2-84
  Supported in Grids 2-25

G

General Subtab 1-33 to 1-42
  Defined 1-33
Generate Unique Value 2-15
Graphical Workflow Designer 1-29, 1-32, 1-42 to 1-49
  Creating a Delivery Plan 1-45 to 1-49
  Defined 1-29, 1-42
  Toolbar 1-44

Grid
Adding Rows 2-58
Date and Time 2-24
Designing 2-18 to 2-22
Example 2-43, 2-58
Height 2-21, 2-23
Mandatory Field 2-23
Maximum Number of Rows 2-21
Options 2-21
Person Search 2-23
Properties 2-22
Supported Actions 2-24
Supported ISF 2-25
URL 2-24
Using on a Form 2-22
Width 2-23
Grid Fields 2-18
HTML Representation 2-21
Groups 4-45

H

HTML Editor Tools 1-84
HTML Portlets 4-17
HTML Representation 2-13
Grid Fields 2-21
Input Type 2-13

I

Implementation 5-3
Configuring 5-12
Import File Format 3-51
Import from File Subtab 3-48
Importing
Service Items 3-49
Standards 3-50
Included Participants 1-65
Included Tasks 1-64
Initiator
Defined 1-5
Dictionary 1-70
Initiator-Based Namespaces 6-36
Initiator Namespaces 6-31
Input Type 2-13
Select (Single) 2-57
Integration Dictionary 2-11
Integration Folder 1-71
Integration Wizard 1-71
Integration Dictionary 2-11
Interactive Service Forms. See ISF.
ISF 2-2
Application Programming Interface (API) 2-82
Coding and Best Practices 2-104
Components 2-83
Defined 1-5, 2-82
Function Names 2-106
Global Identifiers 2-83
Integrating ISF Code into Service Forms 2-92
Supported in Grids 2-25
Testing Code 2-110
Using Server-Side Events 2-79

J

JavaScript 2-82
JavaScript Functions 2-84
Adding Arguments 2-94
Associating Libraries 2-95
Creating and Maintaining 2-93
JavaScript Portlets 4-18
JSR Portlets 4-21 to 4-25
Adding to the Portal 4-24
Migrating Between Portals 4-25
Removing from the Portal 4-25

K

Key Terms 5-3
- Service Designer 1-4

Keywords 1-15, 4-32
- Adding New 1-80
- Associating 1-81
- Defined 1-80
- Deleting 1-82
- Removing Associations 1-82

L

Libraries
- Adding New 2-96
- Creating 2-107
- Defined 2-96
- Structuring and Using 2-106

Library Packages
- Creating 5-48

Lifecycle, Service Item 3-1

Lifecycle Center Module 3-1

Lightweight Namespaces 2-2, 2-26, 2-115, 6-33

Links 2-103

Lookup Conditions 2-37

M

Manage Service Items Tab 3-5, 3-15

Manage Standards Tab 3-21

Mandatory Grid Field 2-23

MDR. See Metadata Repository (MDR).

Message Namespaces 6-38

Metadata Repository (MDR) 3-71

Moments, Defined 1-5

My Items Portlet 3-44
- Enabling 3-46

My Service Items Portlet, Enabling 3-70

My Services
- Preferences 3-46
- Profile 3-46
- Service Items Tab 3-44

My Services 360-Degree Consumer Role 3-4, 3-70

My Services 360-Degree Professional Role 3-70

My Services Consumer Role 3-4

My Workspace Module 4-26, 4-47

N

Namespaces
- Conditional Namespace Elements 6-26
- Customer 6-29
- Customer-Based 6-34
- Defined 1-6, 6-1
- Demand Center 6-39
- Email Namespace Elements 6-24
- Initiator 6-31
- Initiator-Based 6-36
- Lightweight 2-2, 2-26, 2-115, 6-33
- Message 6-38
- Namespace Reference 6-23 to 6-39
- Nodes 6-2
- Node Types 6-2
- Organizational Unit-Based Namespaces 6-27
- Peforrmer 6-30
- PerformerQueue 6-30
- Person-Based 6-27
- References 6-2
- Requisition 6-38
- Usage in Email Templates 6-7

Naming Standards 2-106

Nodes 6-2

O

Offer Tab 1-19
Ongoing Status 1-61
Order Status Portlet 4-12
Organizational Unit 4-43
  Homepage 4-47
  Settings 4-32
Organizational Unit-Based Namespaces 6-27
Organization Designer 3-3

P

Parent Service 1-62 to 1-68
Participants, Included 1-65
Participants Subtab 1-49
PeformerQueue Namespaces 6-30
People, Assigning to a Functional Position 1-9
Performer Namespaces 6-30
Permissions
  Portal Page 4-30
  Portlet 4-11
  Service Group 1-11
Permissions Tab 1-59
Person-Based Dictionary 1-69, 1-70, 2-7
Person-Based Fields 2-91
Person-Based Namespaces 6-27
Persons 4-44
Person Search
  Grid Field 2-23
Plan Tab 1-28 to 1-32
Portal
  Accessing External Site 4-34
  Common Settings 4-31
  Defined 4-3
  Home Pages 4-47
Portal Access Control 4-56 to 4-57
Portal Content
  Exporting 4-52
  Importing 4-54
Portal Designer 4-2
  Defined 4-3
  Screens 4-3
Portal Manager 4-1
  Reference Data 4-36 to 4-46
Portal Modules 4-46
Portal Page 4-25 to 4-30
  Adding Portlets 4-28
  Creating 4-25, 4-51
  Defined 4-3
  Edit Mode 4-49
  Export 4-54
  General Information 4-26
  Page Settings 4-50
  Permissions 4-30
  Subscribed Users 4-30
  View Mode 4-47
Portal Page Content 4-28
Portal Page Group 4-25
  My Workspace 4-26
  System 4-26
Portal Settings 4-31 to 4-35
Portlet
  Adding to Page 4-49
  Core Entities 4-7
  Creating 4-6
  Defined 4-3
  Export 4-53
  Filter 4-10
  HTML 4-17
  JavaScript 4-18
  JSR 4-21 to 4-25
  Maximum Number of Portlets on a Page 4-31
  My Items 3-44
  Permissions 4-11
  Refresh 4-48

Reserved 4-3, 4-12
User-Defined 4-3
View 4-7
Portlet Taxonomy 4-3
Power Operations 3-38
Pricing
Dynamic 1-23
Options 1-23
Pricing a Service 1-21 to 1-23
Pricing Summary 1-22
Process Namespaces 6-38
Project Manager, Defined 1-30

Q

Query Type 2-35
Database Table Lookup 2-36
Enter Your Own SQL Query 2-41

R

Read-Only Fields 2-119
Related Services 3-45
Reportable 1-14
Requisition API (RAPI) 2-15, 2-115
Requisition Namespaces 6-38
Requisitions 4-42
Reserved Dictionaries 1-68, 1-70, 2-9
Reserved Folder 1-68
Reserved Portlets 4-3, 4-12
REST API, Invoking 4-19
Reviews 1-10, 6-12
Namespace Usage 6-12
RIA. See Rich Internet Applicatoin (RIA).
Rich Internet Application (RIA) 2-2
Role-Based Access Control (RBAC) 3-3, 4-1, 4-2
Roles
My Services 360-Degree Consumer 3-4
My Services 360-Degree Consumer Role 3-70
My Services 360-Degree Professional 3-70
My Services Consumer 3-4
Portal End Users 4-57
Service Item Manager 3-69
Rule Type 2-34

S

Scheduled Start Task
Creating 1-60 to 1-62
System Behavior 1-61
Scheduled Status 1-61
Scripts, JavaScripts 2-93
Search Portlet 4-12
Securing Sensitive Data 2-15, 2-77, 2-121
Security 2-15, 2-77, 2-121
Select_Person Attribute 1-69, 2-8 to 2-9, 2-14
Server-Side Events 2-31, 2-32, 2-35, 2-74, 2-76, 2-77
Service
Adding Forms 1-71
Bundles 1-19, 1-62, 2-77
Copying 1-17
Creating New 1-12
Defined 1-5
Deleting 1-19
Edit the Exported XML File 1-18
Exporting 1-18
Formatting Appearance 1-24 to 1-28
General Information 1-12 to 1-16
Importing 1-18
Permission to Order 1-59
Preview 1-27
Pricing 1-21 to 1-23
Removing Forms 1-75
Searching and Viewing 1-16
Status 1-13

Worklfow Design 1-28
Service Designer
Components 1-2
Defined 1-1, 3-3
Key Terms 1-4
Service Form
Appearance 2-15
Conditional Rule 2-1
Data Retrieval Rule 2-1
Defined 1-1, 2-1, 2-4
Integrating ISF 2-92
Performance 2-77
Service Group 1-5
Authorizations Tab 1-9
Authorization Structure 1-10
Authorization Types 1-10
Defined 1-6
Deleting 1-12
General Information 1-7 to 1-9
Permissions 1-11
Reserved 2-9
Searching and Viewing 1-7
Service Item 3-3
Adding New 3-16
Associating with Services 3-18
Configuring Access 3-69
Defined 1-5
Designing 3-6
Fields 3-24
Filter and Search 3-17
History 3-9
History Field 3-24
Importing 3-49
Lifecycle 3-1
Managing 3-15
Subscription 3-9
Subscription Field 3-24
Task Operations 3-31
Tracking 1-6
Virtual Machine 3-10
Service Item-Based Dictionary (SIBD) 1-37, 1-71, 3-25, 3-27
Display Properties 3-27
Pre-Fill Data 3-27
Service Item Definition 3-7
Service Item Details 3-45
Service Item Group 3-4
Creating 3-6
Service Item Instance 3-5
Enable Automatic Retrieval 3-28
Service Item Manager
Defined 1-37, 3-3, 3-4
Roles 3-69
Taxonomy 3-4
Service Items Page 3-45
Related Services 3-45
Service Item Details 3-45
Service Item Task 3-3
Configuring 1-37
Service Item Type 3-4
Service Link 3-3
Defined 1-5
Importing Service Items and Standards 3-64
Service Offerings 4-40
Services 4-39
Service Team, Defined 1-5, 1-8
Settings
Authentication 4-32
Organizational Unit 4-32
Portal 4-31 to 4-35
Show in Grid 2-18, 2-45
SIBD. See Service Item-Based Dictionary (SIBD).
Single Sign-On 4-32
Site Authorization Scheme 1-59
Site Homepage 4-30, 4-47
Site Protection Level 5-14
Guidelines 5-14
Snapshot Management 3-38
Specialized Field-Level Functions 2-90 to 2-92

SQL Query 2-41
Standard Duration 1-14
Standards
Coding and Naming 2-106
Defining 3-19 to 3-21
Importing 3-50
Managing 3-21
Virtual Data Center 3-20
Standards Tables 3-19
Status
Ongoing 1-61
Scheduled 1-61
Service 1-13
Subscribed Users 4-30
System Module 4-47
System Moment 2-27

T

Task 4-43
Participants 1-49
Performer 1-50
Supervisor 1-50
Task Instructions Subtab 1-53
Task Priority 1-35
Tasks Subtab 1-29
Defined 1-29
Fields 1-31
General Subtab 1-31
Monitor Task 1-30
Testing, ISF Code 2-110
Toolbar, Graphical Workflow Designer 1-44
Tracking, Service Items 1-6
Triggering Events 2-31, 2-35, 2-76

U

Unexpected Token Message 1-41
Universally Unique Identifier (UUID) 2-15
URL, In Grid 2-24
User-Defined Portlets 4-3
User Homepage 4-47
UUID. See Universally Unique Identifier (UUID).

V

Validating Expressions 1-41
Validating Rule 2-34, 2-39, 2-80
Virtual Data Center Standards 3-20
Virtual Machine 3-3, 3-10
Cloning 3-37
Creating 3-36
Deleting 3-37
Details 3-42
Importing 3-47
Operation Fields 3-26
Reconfiguration 3-39, 3-43
VMware 3-3
Adapter Error Messages 3-73
Configuring Operations 3-33

W

Web Portal. See Portal.
Workflow
Directory Tasks 1-37
External Tasks 1-36
Service Design 1-28
Type 1-33