



Cisco Service Portal Integration 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 Integration Guide

© 2012 Cisco Systems, Inc. All rights reserved.



CONTENTS

About this Guide xi

CHAPTER 1

Directory Integration and API 1-1

Overview 1-1

Introduction 1-1

Prerequisites 1-2

Purpose and Scope 1-2

Intended Audience 1-2

Gathering Directory Integration Requirements 1-2

Overview 1-2

Defining Datasources 1-3

Defining Mappings 1-4

Defining Integration Events, Operations and Steps 1-9

Configuring Directory Integration 1-20

Enabling Directory Integration 1-20

Configuring Directory Integration 1-22

Configuring Datasource Information 1-22

Configuring Mappings 1-26

Testing Mappings 1-31

Configuring Directory Integration Events 1-34

Using Custom Code in Directory Integration 1-36

Custom Code Operation Interfaces 1-38

Custom Java Class Mapping Interface 1-44

Directory Server API 1-45

Import/Refresh Person API 1-47

Best Practices 1-48

Compiling Custom Code Java Files 1-48

Coding Guidelines 1-49

Configuring Custom Code in the Administration Module 1-49

Deploying Custom Code 1-51

Sample View/Usage of the API 1-51

SQL Datasource 1-51

Datasource Definition 1-52

Sample Mapping 1-53

Sample Event Configuration 1-54
 Sample Code for SQL-Based Person Lookup 1-55
 Supported Time Zones 1-63
 Sample build.xml File 1-65

CHAPTER 2

Service Link 2-1
 Overview 2-1
 Introduction 2-1
 Service Link Prerequisites 2-2
 Service Link Design Methodology and Components 2-2
 Business Engine and nsXML 2-3
 Service Link Design and Development 2-4
 Overview 2-4
 Accessing Service Link 2-5
 Designing the Communication Protocol 2-8
 Adapters 2-8
 Agents 2-10
 Transformations 2-20
 Reviewing Agent Definitions and Property Sheets 2-22
 Configuring a Task to use a Service Link Agent 2-23
 nsXML Messages 2-25
 Creating and Deploying a Service Link Agent 2-33
 Monitoring Service Link Transactions 2-34
 Viewing Messages from the Service Link Home Page 2-34
 Viewing Messages 2-35
 Viewing External Tasks 2-39
 Republishing Service Link Messages 2-42
 Service Link Adapters 2-42
 Auto-Complete Adapter 2-43
 Dummy Adapter 2-43
 Database Adapter 2-43
 File Adapter 2-48
 HTTP/WS Adapter 2-49
 JMS Adapter 2-54
 MQ Adapter 2-55
 Service Item Listener Adapter 2-56
 VMware Adapter 2-57
 Web Service Listener Adapter 2-58
 Integration Wizard 2-59

Overview	2-59
Using the Integration Wizard	2-60
Outbound Request Parameter Mappings	2-63
Outbound Response Parameter Mappings	2-64
Integration Summary	2-64
Service Link Troubleshooting and Administration	2-66
Checking Service Link Status	2-66
Starting and Stopping Agents	2-66
Logging	2-66
Message Purging	2-67
Application Server Configuration Files	2-68
Online Error Log	2-68
Prebuilt Functions	2-69
Overview	2-69
Function Usage	2-69
Function Synopsis	2-70

CHAPTER 3**Service Link Adapter Development Kit 3-1**

Overview	3-1
Intended Audience	3-1
Getting Started	3-1
Installing the JDK	3-2
Installing the ADK	3-2
ADK Structure	3-2
Creating Adapter Source Structures	3-3
Compiling Adapters	3-3
Deploying Adapters	3-4
What is an Adapter?	3-4
Concepts	3-4
Types of Adapters	3-5
Properties	3-6
Example Adapter	3-6
Directory Structure	3-6
Outbound Adapter Class	3-6
Poller Inbound Adapter Class	3-8
Listener Inbound Adapter	3-9
Exception Handler	3-9
Transaction Support	3-9
Understanding the adapter.xml Descriptor	3-10

- nsXML Format 3-13
 - Message 3-13
 - Task Started or Task Cancelled 3-14
 - Task 3-15
 - Requisition 3-17
 - Requisition Entry 3-18
 - Data Values 3-19
 - Service 3-20
 - Dictionary 3-21
 - Form 3-22
 - Agent Parameter 3-23
- Sample Inbound and Outbound Documents 3-24
 - task-started or task-cancelled (outgoing) 3-24
 - take-action (incoming) 3-30
 - send-parameters (incoming) 3-30
 - update-data (incoming) 3-30
 - add-comments (incoming) 3-31

CHAPTER 4

- Remedy Service Adapter 4-1**
 - Overview 4-1
 - Integration Scenarios 4-2
 - Prerequisites 4-4
 - Service Portal Requirements 4-4
 - BMC Requirements 4-4
 - BMC Remedy Configuration Steps (Sample) 4-4
 - Obtaining the Adapter 4-4
 - Installing the Adapter 4-5
 - Viewing the Adapter 4-5
 - Configuring the Agent 4-5
 - Outbound Properties 4-7
 - Inbound Properties 4-7
 - Configuring the Transformation 4-7
 - Inbound Transformation Details 4-8
 - Outbound Transformation Details 4-9
 - Outbound and Inbound Date Format Transformations 4-10
 - Designing a Service for a Request Handled by the Remedy Adapter 4-11
 - Test Scenario 4-12
 - Log Messages 4-12

CHAPTER 5**Web Services 5-1**

- Overview 5-1
 - Audience 5-1
 - Web Services 5-1
- Prerequisites for Web Services 5-2
 - Service Portal Installation and Configuration 5-2
 - Testing and Development Environment 5-4
 - Generating Code 5-4
- Web Services for Request Management 5-5
 - Overview 5-5
 - Sample Service Definition 5-5
 - Authentication 5-6
 - Getting the Service Definition 5-8
 - Submitting a Requisition 5-10
 - Getting a List of Requisitions 5-12
 - Getting the Requisition Status 5-13
 - Adding Comments to a Requisition 5-14
 - Cancelling a Requisition 5-15
- Web Services for Task Management 5-16
 - Overview 5-16
 - Getting a List of Authorizations 5-16
 - Approving or Rejecting an Authorization 5-18
- Web Services for Portfolio Management 5-19
 - Overview 5-19
 - Exporting Offering Cost Data 5-19
 - Retrieving Service Offerings and their Status 5-19
- Sample Requests and Responses 5-21
 - getServiceDefinition Response 5-21
 - Sample submitRequisition Request 5-31
 - Sample exportOfferingCostData Response 5-35
- Web Services Error Messages 5-40

CHAPTER 6**REST API 6-1**

- Overview 6-1
 - Supported Entities 6-1
 - Operations 6-2
 - Conventions and Syntax 6-2
 - Filters 6-3

- Sorting 6-6
- Paging 6-7
- Nested Entities 6-8
- Invoking REST API 6-9
 - Using nsAPI with HTTP Clients 6-9
 - Using nsAPI with JavaScript Portlets 6-10
 - Using nsAPI with JSR Portlets 6-12
- Detailed API Reference 6-16
 - Definitional Data 6-16
 - Directory Data 6-24
 - Transactional Data 6-31
 - Lifecycle Center Data 6-39
 - Service Portal Data 6-47
- Error Messages 6-48
- Quick Reference 6-49

CHAPTER 7

- JSR Portlets 7-1**
 - Overview 7-1
 - Portlet Structure and Packaging 7-1
 - JBoss Application Server 7-1
 - Weblogic Application Server 7-3
 - WebSphere Application Server 7-4
 - Dependent Libraries 7-5
 - Portlet Development 7-6
 - MyJSR.css 7-8
 - MyJSRCreatePersonView.js 7-8
 - MyJSREdit.js 7-10
 - MyJSRHelp.js 7-10
 - MyJSRView.js 7-10
 - portlet.xml 7-13
 - web.xml 7-14
 - MyJSREdit.jsp 7-15
 - MyJSRHelp.jsp 7-16
 - MyJSRView_listperson.jsp 7-17
 - MyJSRView_updateperson.jsp 7-19
 - MyJSRController.java 7-20
 - MyJSRApplicationContext.xml 7-26
 - jsrportlet.properties 7-26
 - Log4j.properties 7-26

jboss-deployment-structure.xml	7-27
Compiling JSR Portlet Controller	7-27
Portlet Deployment	7-28

INDEX



About this Guide

Objectives

The *Cisco Service Portal Integration Guide* explains all of the features you can use to integrate the Cisco Service Portal (Service Portal) application to other applications and systems for a complete solution at your site.

Service Portal is built to integrate with your corporate directory so that it can consume data from that directory to determine each end-user's department membership, role, and place in the corporate reporting structure. The data in that directory may need to be augmented and transformed at the same time as it is loaded into the Portal database, however, to fulfill the specific requirements of your service catalog. This guide explains how you do just that.

Service Portal also provides a number of APIs for solving complex service delivery use-cases, including automation, and the submission of service requests and instantiation of service items through a mechanism other than the Portal user interface. Those APIs are fully documented here.

Finally, this guide explains the Service Link module, through which you can configure Agents that orchestrate workflows in external systems.

Audience

This guide is aimed at the individual or team responsible for implementing the Service Portal application and integrating it with all other corporate systems, including the corporate directory or directories.

Document Organization

The *Cisco Service Portal Integration Guide* is divided into the following seven chapters:

- [Chapter 1, “Directory Integration and API”](#): This chapter describes how to configure directory integration for Service Portal using the Administration module. It also describes the set of public APIs and interfaces available for customizing the integration options available; best practices for compiling and deploying custom code; and steps to configure the custom code using the Administration module.
- [Chapter 2, “Service Link”](#): This chapter describes the Service Link module that provides integration with external systems.
- [Chapter 3, “Service Link Adapter Development Kit”](#): This chapter contains instructions for using the Service Link Adapter Development Kit (ADK) to develop Service Link adapters.

- [Chapter 4, “Remedy Service Adapter”](#): This chapter describes the Cisco Service Adapter for BMC® Remedy® IT Service Management.
- [Chapter 5, “Web Services”](#): This chapter describes the use of web services for Service Portal.
- [Chapter 6, “REST API”](#): This chapter describes the Cisco standard REST (Representational State Transfer) APIs and Java stubs for accessing entities defined in Service Portal.
- [Chapter 7, “JSR Portlets”](#): This chapter covers some guidelines on the development and deployment of JSR portlets for the Portal Manager solution.

Conventions

This document uses the following conventions:

Convention	Indication
bold font	Commands and keywords and user-entered text appear in bold font .
<i>italic font</i>	Document titles, new or emphasized terms, and arguments for which you supply values are in <i>italic font</i> .
[]	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

Directory Integration and API

- [Overview, page 1-1](#)
- [Gathering Directory Integration Requirements , page 1-2](#)
- [Configuring Directory Integration, page 1-20](#)
- [Using Custom Code in Directory Integration, page 1-36](#)
- [Best Practices, page 1-48](#)
- [Sample View/Usage of the API , page 1-51](#)
- [Supported Time Zones, page 1-63](#)
- [Sample build.xml File, page 1-65](#)

Overview

Introduction

Service Portal Directory Integration simplifies security administration and enhances user convenience and productivity by implementing centralized user authentication and synchronization with an enterprise directory.

Service Portal enables customers to integrate with an external directory (typically using the LDAP protocol) for user information synchronization. This synchronization is invoked whenever a user is selected for Order-on-behalf (OOB) or during Person Lookup.

Single Sign-On (SSO) integration enables centralized user authentication, eliminating the need for a separate login mechanism. When the SSO event is enabled, users who are already logged into an enterprise portal with which Service Portal has been integrated do not have to login again. User authorization data is stored within the application database. Service Portal relies on the SSO tool to protect all Demand Center and Request Center URLs and for the SSO tool to perform authentication. Service Portal expects the SSO tool to provide person identification information for each successful authentication to a Service Portal URL via the HTTP header. Once a person has been authenticated, their information can be synchronized to the application database.

If SSO is not enabled, then the Service Portal login screen is presented to all users so they can provide a valid username and password combination. By default, these credentials are authenticated against the internal database. Alternatively, Directory Integration could be configured to authenticate to an external system (generally an LDAP directory). Any users who wish to access Service Portal must be present in this source for successful authentication.

The Directory Integration Framework provides the above capabilities for many frequently deployed SSO and directory server products through configuration options available in the Administration module. The framework also includes an application programming interface (API) which can supplement predefined configuration capabilities. The API allows programmers to access additional SSO portals and directory servers, as well as to alter or supplement default behavior for synchronizing user information between Service Portal and the external directory.

Prerequisites

Configuring directory integration requires the following:

- A working Service Portal installation.
- Directory server installed and directories populated with corporate data. Directory entries for all potential users must contain non-null values for all attributes that are mapped to fields required for integration operation, as explained in the [“Defining Mappings” section on page 1-4](#).
- If Single Sign-On (SSO) is to be used, an SSO system that is responsible for the authenticating and authorizing access to Service Portal.
- A user login with a role that includes the capability to “Manage Global Settings”. This capability is automatically included in the “Site Administrator” role and assigned to the “admin” user, but may be assigned to other roles or users as appropriate, using the Roles option in the Administration module.

Access to an LDAP browser is strongly recommended.

Purpose and Scope

This chapter describes how to configure directory integration for Service Portal using the Administration module. It also describes the set of public APIs and interfaces available for customizing the integration options available; best practices for compiling and deploying custom code; and steps to configure the custom code using the Administration module.

Intended Audience

This chapter is intended for software customization and integration engineers.

Gathering Directory Integration Requirements

Overview

To configure directory integration, you need to have handy information about the current implementation of SSO (if used) and directory servers at your company, and to document the requirements for integrating these systems with Service Portal. This section provides a set of worksheets for collecting this information.

These worksheets should help you collect the information required to configure directory/SSO integration, and to identify issues which need to be resolved before the integration can be implemented. This, in turn, can help in estimating the amount of development and testing time required for the directory integration.

Defining Datasources

Service Portal defines a “datasource” for each directory which stores personnel and organization data to be accessed. The datasource definition includes all information required to connect to the external directory and extracting information from that directory.

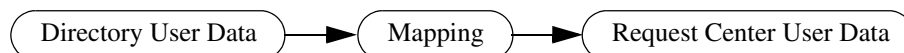
You will need to define one datasource for each external directory. For example, different development and production directories may be used. In addition, Service Portal supports LDAP directory referrals—a datasource needs to be defined for each directory in the referral chain.

Setting	Value	Description
Datasource Name		The name of the datasource. Do not use spaces or special characters.
Datasource Description		Optional description of the datasource.
Protocol	<ul style="list-style-type: none"> LDAP 	LDAP is the only supported protocol at this time. If directory information is stored using another protocol, you need to create custom code to access this information.
Server Product	<ul style="list-style-type: none"> Sun™ ONE Directory Microsoft® Active Directory® IBM® Tivoli® 	Choose the directory server product you are using. If the server is not currently supported, you will need to create custom code to access the server and extract directory information.
Authentication Method	<ul style="list-style-type: none"> Simple Anonymous SASL 	Simple means plain text user/password. SASL (Simple Authentication and Security Layer) is also available, but SASL only works with Sun ONE Directory Server.
Connection Mechanism	<ul style="list-style-type: none"> SSL Non SSL 	Only needed if you choose Simple or SASL as the authentication method. Choose SSL to send encrypted information.
BindDN		Bind distinguished name field. BindDN is used to connect to the LDAP server when Service Portal performs a directory operation. You may want to create a service account for this purpose. When this datasource is used in an External Authentication step, you will provide an EUA Bind DN in the Options area to override this value. For more details, see the “External User Authentication (EUA) Operation” section on page 1-12.
Host		Fully qualified hostname or IP address of the LDAP directory server.

Setting	Value	Description
Port Number		Port number to connect to the directory server. Port Number 389 is typically used for non-SSL access.
Password		Required if you choose Simple or SASL authentication; the password for the user specified as the Bind DN. If the account uses password aging, you will need to update this password periodically.
User BaseDN		The directory from which to start searching for persons in the directory; since corporate directories may include many branches, specifying a base DN for the user data will optimize directory searches.
AuthzID		Required if you choose SASL authentication.
Optional Filter		This filter are added to other search filters you use, and it can be used to effectively change the search results. The filter expression must be enclosed in parentheses; for example, the filter: (&(!(msExchHide=true)(ISC-GID=*)) will return only those entries for which the msExchHide attribute is true and for which an ISC-GID attribute is defined.
Security Certificate Name		Required if you choose SSL as the connection mechanism. Do not use spaces or special characters in the certificate alias name. Ensure that you have the certificate data ready to enter.
Referral Datasource		You can add one or more datasources as a referral. When a datasource search does not return results, the system searches the referral datasources as well. Referrals are supported for searches only, not binding. <i>You cannot set up cyclic referrals. Cyclic referrals are those where one datasource has another datasource as a referral, while that datasource has the original as a referral. For example, datasource A has datasource B as a referral, while datasource B has datasource A as a referral.</i>

Defining Mappings

A “mapping” is a set of rules that give instructions for how data is to be transferred from the external directory to Service Portal. It maps between source attributes in the directory and target fields in the Service Portal database. The rules are used to transfer data from the directory to the designated target field when the Service Portal database is synchronized with the directory.



The same mapping can be applied to multiple directories (datasources).

A mapping includes the user/person's profile along with all related entities: addresses, contacts, locations, one or more group associations, one or more organizational unit (OU) associations, and one or more RBAC (role-based access control) role associations.

A person profile includes seven mandatory fields, listed in the "Mandatory" section of the Mapping Worksheet below. Directory records which do not provide a value for any of these fields cannot be imported. Other fields which are part of the person profile can also be mapped. For an overview of these fields, consult the screens available in Organization Designer for maintaining People information.

Most of the fields on the person profile are used by application processes, and the mapping should ensure that mapped attributes provide a source value appropriate for the field; that is, do not try to overload these fields with more information than would be suggested by the field name, or with information that does not match the field name.

Service Portal also includes fields which provide an extension to the standard personnel data. These fields are denoted as "Extension" on the following table and appear on the Extensions page of the Person information in Organization Designer. 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 Request Center, map the attributes containing that information to one of the personnel extended fields.

The "Directory Attribute" column in the worksheet below should be filled in for all Person profile fields for which the directory must supply data. The Attribute should be one of the following:

- The directory attribute name or names, if two or more attributes can be concatenated (with optional literals) to form the value for the field.
- "Custom mapping", following by a number or description. All custom mappings should be explained in detail in the ["Custom Mappings" section on page 1-8](#) or noted briefly in the "Comments" column. Custom mappings may assign the result of a regular expression to the attribute, or may be implemented via a module of custom Java code. Details for implementing these mappings are given in the ["Configuring Mappings" section on page 1-26](#).

Mandatory Mappings

Field	Comments
First Name	
Last Name	
Login ID	Unique identifier to be used as the person's login name for Service Portal.
Person Identification	The Person Identification should map to an attribute that provides a unique value for each person. For example, specify an attribute that contains the employee id or social security number. Ideally, the same attribute should map to both the Login ID and the Person Identification; at a minimum, the two should be tightly coupled.
Email Address	
Home Organizational Unit	The Home OU is always a business unit, not a service team.
Password	Directory servers will typically not return a password. However you can use this field to create, for example, default passwords for new users.

Optional Mappings

Field	Comments
Title	
Social Security Number	
Birthdate	The return type of the LDAP attribute being mapped must return a long. Service Portal does not support other formats.
Hire Date	The return type of the LDAP attribute being mapped must return a long. Service Portal does not support other formats.
Timezone ID	<p>The mapped attribute must return a value in one of the following formats:</p> <ul style="list-style-type: none"> • GMT+- Offset • Country/Language <p>As of the March 2008, the familiar three-letter time zone designations (for example, “EST” for Eastern Standard Time) should not be used. For a list of supported values for the above formats, see the “Supported Time Zones” section on page 1-63. If the return value does not match one of the valid formats, Service Portal uses PST as the default time zone.</p>
Locale ID	<p>The mapped attribute must return a value in the form:</p> <p>language_COUNTRY</p> <p>where language is a two-letter language code and the country is a two-letter country code.</p> <p>Directory integration supports the following locales:</p> <ul style="list-style-type: none"> • en_US (United State English) • de_DE (German) • es_ES (Spanish) • fr_FR (French) • ja_JP (Japanese) • zh_CN (Mainland Chinese) • zh_TW (Taiwanese Chinese)
Employee Code	
Supervisor	This field represents the identification of manager. For more details see the “Import Manager Operation” section on page 1-16 .
Notes	
Company Street 1	
Company Street 2	
Company City	
Company State	
Company Postal Code	

Field	Comments
Company	
Country	
Building	
Level	
Office	
Cubicle	
Personal Street 1	
Personal Street 2	
Personal City	
Personal State	
Personal Postal Code	
Personal Country	
Work Phone	
Home phone	
Fax	
Mobile Phone	
Pager	
Other	
Main Phone	
Primary Phone	
Primary Fax	
Sales Phone	
Support Phone	
Billing Phone	
Other Contact Information	
Company Code	Extension
Division	Extension
Business Unit	Extension
Department Number	Extension
Cost Center	Extension
Management Level	This should return a number. When used with the Import Manager event, Management Level is expected to be in increasing order according to the hierarchy. For example, if there are two designations, Junior Engineer and Senior Engineer, Management Level returned for Junior Engineer should be less than the Management Level of Senior Engineer.
Region	Extension

Field	Comments
Employee Type	Extension
Location Code	Extension
Custom 1	Extension
Custom 2	Extension
Custom 3	Extension
Custom 4	Extension
Custom 5	Extension
Custom 6	Extension
Custom 7	Extension
Custom 8	Extension
Custom 9	Extension
Custom 10	Extension
Organizational Unit List	<p>Use this mapping to associate the person with one or more Organizational Units. The mapping may return multiple values. For this field Service Portal uses all values returned by multivalued LDAP attributes. Input for this field should be in one of the following formats:</p> <ul style="list-style-type: none"> Name of the Java class that returns the multiple values as defined in Directory Integration API documentation. One or more simple mappings separated by “:”. For example, ou::departmentNumber. One or more expression mappings separated by “:”, as in: expr:#memberOf#=(cn=(.*),cn=Users,dc=celosis,dc=com)?(\$1):Default:: expr:#memberOf#=(cn=(.*),ou=Users,dc=celosis,dc=com)?(\$1):Default
Group List	Similar to Organizational Unit List.
Role List	<p>Similar to Organizational Unit List. The returned roles may be either system- or user-defined.</p> <p>For system-defined roles, the names must be exactly as they appear in a browser with language US English. Other languages are not supported. For example, “My Services Executive” should be returned to associate a user with this role.</p> <p>For user-defined roles, the name must exactly match the user input in user language while creating the role.</p>

Custom Mappings

You may use the worksheet below to document requirements for custom mappings.

Field	Type	Requirements
	Expression Java	
	Expression Java	

Defining Integration Events, Operations and Steps

Integration events are the interfaces between Service Portal and an external directory or SSO program—the only times in the use of Service Portal that the external program or directory is accessed. These events consist of a series of operations which are executed in sequence.

Events

Service Portal supports four directory integration events:

- The “**Login**” event occurs when a user's credentials are validated and the user connects to Service Portal. This event occurs when a user initially starts a Service Portal session. It also occurs if a session times out (the administrator-specified time-out period expires) and the user must reconnect.
- A “**Person Lookup**” event occurs every time user information must be retrieved. There are actually three types of Person Lookup events:
 - **Person Lookup for Order on Behalf:** A user requests a service on behalf of another person, and must choose the person who is the customer for the service.
 - **Person Lookup for Service Form:** A service form includes a Person field, which allows the user to designate another person as part of the service data.
 - **Person Lookup for Authorization Delegate:** A user responsible for reviewing or authorizing service requests modifies his/her profile to designate another person as a temporary authorization delegate.

Operations

You can configure events to perform various types of operations. The operations are specified for each event in a series of steps, which determines the sequence in which each operation is invoked.

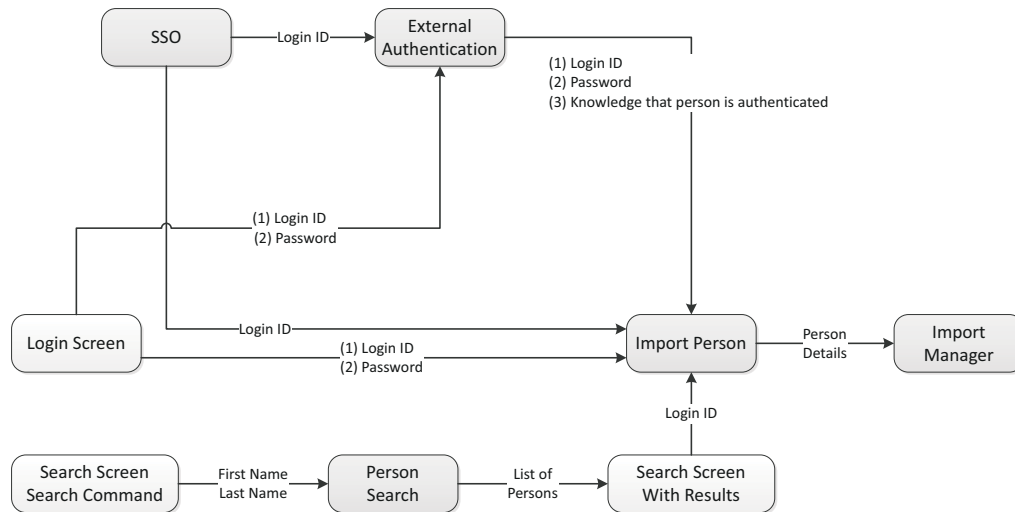
The directory framework includes the following operations:

- **Single Sign-On (SSO)** is always the first step in the Login event. The SSO operation identifies the login name of the user.
- **External Authentication** can occur after the SSO operation or, if an SSO operation is not used, after the default Login screen. External Authentication uses the login name and password of the user and authenticates them against an external datasource.
- **Person Search** is triggered when the user invokes a search on a datasource. Person Search uses the First Name and Last Name of the user to provide a list of matched items.
- **Import Person** can occur after External Authentication or after SSO, or after a person is chosen in the Person Search dialog box. Import Person uses the login name of the person searched or logging in to query a datasource and import the person into the database.
- **Import Manager** can only occur after Import Person. The Import Manager operation will use the imported person information to import the managers of this person.

Each operation can be customized via implementation of custom code interfaces.

Figure 1-1 below shows the sequence in which operations are triggered.

Figure 1-1 Trigger Order



Login Event

If a directory integration login event is not configured, the default behavior is to present the login screen and validate the credentials entered (user name and password) against the contents of the application database.

If the directory integration event is enabled, the Login event may be configured with either one of the following operations as its first step:

- **Single Sign-On:** In a corporate environment where all users are preauthenticated using SSO vendors, automatically extract the login id of the user from request headers or CGI headers and allow transparent login, bypassing the application login screen.
- **External User Authentication:** Present the application login screen and validate the credentials entered against the specified external directory. External User Authentication may also follow an SSO operation.

Once the user credentials have been validated, the Login event may include additional operations to synchronize user data between the external datasource and Service Portal:

- The “Import Person” operation may be the next step. This operation imports the profile of the authenticated person selected to Service Portal, synchronizing the data.
- The “Import Manager” operation may follow the “Import Person” step. This operation retrieves information on the managers of the selected person from the external directory and updates the Service Portal database with that information.

Single Sign-On Operation

Integration with Single Sign-On (SSO) solutions can use one of the following two mechanisms/protocols:

1. Active Directory Services (ADS)/NT LAN Manager (NTLM)-based authenticated user
 - The third-party IM/AM/SSO product is not needed to log into Service Portal.
 - The logged in user credentials from any POSIX-compliant OS are returned by the browser to Service Portal.

- This is also called integration through CGI Headers for SSO.
2. HTTP Request Headers
- This is for non-ADS/NTLM integration.
 - It requires the third-party IM/AM/SSO product to log into Service Portal using RequestHeaders in the http protocol.

For customers who plan to use SSO for both Portlet and Directory Integration, only HTTP Header SSO is supported. Custom SSO plug-ins within the Directory Integration framework are not supported.

Setting	Value	Description
Single Sign-On Type	HTTP Header Remote User	Specify the type corresponding to your SSO solution. Be sure to verify that login ID information is accessible. Check HTTP Header to use http Request Header protocol. Check Remote User to use ADS/NTLM protocol.
Login ID Mapping		Login ID mapping for HTTP Sign-Ons should be the exact name of the Http Request Header that contains the login name of user signing in. Login ID mapping for ADS/NTLM Sign-Ons should be of the following format: #AnyDomain#\#LoginId# For example, celosis\#LoginId# limits users to the “celosis” domain, while #AnyDomain#\#LoginId# allows logins across multiple domains. If multiple domains are in use, the LoginId must be unique across domains.
Redirect URL		The URL of the corporate portal from which users typically access Service Portal products. Users are redirected to this URL if authentication fails, or when the application user session times out.

Administrative Bypass of SSO

It is sometimes necessary to allow some users to bypass the Single Sign-On and login directly to Service Portal. This capability is typically required for:

- System administrators who need to investigate problems with Single Sign-On
- Testers who need to emulate the performance of multiple users in order to validate a service design and task plan

Service Portal provides a mechanism for allowing users to access the login screen and enter a user name and password. The `newscale.properties` file (located within the `RequestCenter.war`) specifies a value for the “BackDoorURLParam”; for example:

```
BackDoorURLParam=AdminAccess
```

The URL used to access Service Portal via the login screen must include a parameter. For the above value of the `backDoorURLParam`; for example, a sample URL might be:

```
http://prod.RequestCenter.com/RequestCenter?AdminAccess=true
```

It is the responsibility of the administrator to establish policies for aging out the value of the BackDoorURLParam according to corporate guidelines and for controlling administrative access to Service Portal. Access via the administrative URL can be restricted to only those users who have the “Site Administrator” role via the corresponding Administration Setting:

On	Off	Setting	Description
Common			
<input type="radio"/>	<input checked="" type="radio"/>	Enable Custom Header Footer	Site will add content from the custom header and footer HTML. Default is off.
<input type="radio"/>	<input checked="" type="radio"/>	Enable Custom Style Sheets	Site will utilize the custom stylesheet allowing for the changing of logos, color schemes, fonts and others. Default is off.
<input type="radio"/>	<input checked="" type="radio"/>	Directory Integration	Enable the Directories feature that searches for and imports users into the site from an external datasource (e.g. LDAP). Default is off.
<input type="radio"/>	<input checked="" type="radio"/>	Restrict Site Administrator URL	Allow only those users with the Site Administrator Role to log in using the administrator URL (i.e., bypassing Single Sign-On). Default is off.

The administrator must also ensure that the URL is directly accessible to users—access to the Service Portal application may have previously been restricted to the SSO software via web server or network configuration parameters.

The Request Center service must be restarted for a change to this parameter to take effect.

External User Authentication (EUA) Operation

Use External Authentication to authenticate all Service Portal users with a corporate directory. This way you do not have to worry about synchronizing user passwords.

External User Authentication must follow a login attempt—either via a configured Single Sign-On operation or through the application login screen. The LoginId retrieved from the previous operation is available to the EUA operation. However, validating this user in the external directory requires additional information, so that the BindDN can be located.

The EUABindDN setting allows the application to automatically extrapolate the bind DN of the user trying to sign on.

Setting	Description
External Authentication EUABindDN	<p>EUABindDN is of the format: Prefix#LoginId#Suffix.</p> <p>Service Portal will replace #LoginId# with the loginId of the user signing in from EUABindDN and use it as BindDN for authentication.</p> <p>For example, you can provide the EUABindDN like this: uid=#LoginId#,OU=People,dc=example,dc=com</p> <p>In such case if the user provides scarter as the login id in the logic screen during sign up, Service Portal will use uid=scarter,OU=People,dc=example,dc=com</p> <p>to bind the user with external datasource.</p>

Person Lookup Events

All Person Lookup events (Order on Behalf, Service Form, and Authorization Delegate) share the same behavior and configuration options.

If the directory integration event is not enabled, the Person Search window searches personnel information in the Service Portal database. If a person is selected, their information is used. Personnel information is not updated.

If the directory integration event is enabled, the Person Lookup event may be configured with the following operations:

- The “Person Search” operation must be the first step. This operation retrieves personnel information from the external directory and displays it in the Person Search window. If the user selects a person, additional information on that person is retrieved, according to the mapping specified for the event, and supplied to the calling context.
- The “Import Person” operation may be the next step. This operation imports the profile of the person selected from the external directory to Service Portal, synchronizing the data.
- The “Import Manager” operation may follow the “Import Person” step. This operation retrieves information on the managers of the selected person from the external directory and updates the Service Portal database with that information.

Person Search Operation

Settings for the Person Search operation determine the appearance and behavior of the window that displays people meeting the search criteria.

In order for a person to be imported into Service Portal, all mandatory fields must have a valid attribute mapping, which returns in a nonblank value. If any required values are missing, the default behavior is to exclude that person from the Search Results. The alternative is to include such people in the Search Results, but to flag them as having incomplete information, as shown below.

Figure 1-2 Person Search

Select Person

* Search For: First Name : Search
Last Name :

Search Results

Name	Organizational Unit	Email Address	User Name
Angela Fong	newScale, Inc.	angfong@cisco.com	angfong
Andrew Tahvidary	newScale		andrew

Items 1 - 2 of 2

* = Person cannot be selected because some mandatory information is undefined in the Directory

Cancel OK

People with incomplete information cannot be chosen.

When configuring a Person Search operation

Setting	Value	Comments
Search Selectivity	<ul style="list-style-type: none"> Show People with Incomplete Information 	Default is to exclude people with incomplete information from the Search Results window.
Sort By	<ul style="list-style-type: none"> First Name Last Name First Name First Name Last Name Last Name No Sort 	Default is to sort by Last Name.
Max Results		Default for the number of rows to display in the Search Results is 1000.

The * (Asterisk) Wildcard Character and Person Search

When configuring and testing a Person Search, you need to be aware of the use of the asterisk (*) as a wildcard character.

Transparent to the user, the system always appends an * to the end of the search string. Therefore, if a user enters john in the Last Name field, and clicks **Search**, the system returns all persons in the directory whose last name begins with the word john, such as “John”, “Johnson”, and “Johnston”.

A user may also explicitly enter the * character in the search string of the Search Person dialog box. Some examples of the usage for wildcard search are:

- Enter * in the **Last Name** field, and click **Search**. The system returns all persons in the directory.
- Enter **john*** in the **Last Name** field, and click **Search**. This is essentially the same as typing just **john** in the **Last Name** field. The system returns all persons in the directory whose last name begins with the word “john”.
- Enter ***john** in the **Last Name** field, and click **Search**. The system returns all persons whose last name contains the word “john,” including “John”, “McJohn”, and “Johnson”.
- Enter ***john*son** in the **Last Name** field, and click **Search**. The system returns all persons whose last name contains the word “john,” followed (not necessarily immediately) by the word “son.” These include “Johnson”, “Mcjohnson”, and “Upjohnningson”.



Note

The * is always treated as a wildcard character in the search string. Therefore, the user is NOT able to search for a value in the directory that contains the character *. Any other special characters may be used in the search string.

Configuring the Search Results Window

By default, the Search Results window in the Select Person Popup displays the person's first name followed by the last name. Additional fields can be added to the display by changing the Setting for the Person Popup available in the Administration module.

Figure 1-3 Configuring Search Results Window

The screenshot shows the Cisco Service Portal Administration interface. The top navigation bar includes "Home", "Directories", "Authorizations", "Notifications", "Lists", "Settings", and "Utilities". The "Administration" dropdown menu is open, showing "Customizations" selected. The "Person Popup" configuration page is displayed, featuring a table with two columns: "Column Heading to Display" and "Request Center Person Data to Use for this Heading". The first row shows "Name" in the first column and "First Name Last Name" in the second column. There are three empty rows below, each with a dropdown arrow in the second column. An "Update" button is located at the bottom left of the configuration area. On the right side, a "Customizations" sidebar is visible, listing "Person Popup", "Entity Homes", "Debugging", "Custom Styles", and "Data Source Registry".

Import Person Operation

Import Person settings govern whether person information in the application is refreshed from current information about the selected person (when Import Person is used in a Person Search event) or the person who has logged in (when Import Person is used in a Login event).

Setting	Value	Comments
Refresh	<ul style="list-style-type: none"> Refresh Person Profile Refresh Period (Hours) 	Leave the refresh period blank or zero to refresh on every import—this will ensure that the Service Portal database always reflects recent changes in the external directories. Alternatively, you can designate that a user's profile should be refreshed only after the designated period has passed since this last refresh.
Create Associations	<ul style="list-style-type: none"> Do Not Create Organizational Unit Do Not Create Group 	Default is to create organizational units and groups if they do not exist. Roles cannot be created via directory integration and must exist before the person is imported.
Remove Existing Associations	<ul style="list-style-type: none"> Organizational Unit Group Role 	Default is not to remove existing organizational unit, group, or role associations.

Import Manager Operation

Service Portal allows authorizations and reviews to be dynamically assigned. For example, a request with a dollar value greater than a specified threshold might need approval by the director of a particular department. Another request might need to be reviewed by the requestor's immediate superior.

To implement business rules like these, the managers of an employee who can request a service must also be present in the Service Portal database. The Import Manager operation supports this requirement, importing manager (supervisor) data in conjunction with the employee's data.

To govern the behavior of the Import Manager operation:

- Identify the attribute in the employee's directory entry that is to designate his/her manager.
- For all employees, ensure that the designated attribute is populated with a value that uniquely identifies their manager. This is typically the login id or email address.
- In the mapping for the Supervisor field (listed in the Optional Person Data Mappings) specify the attribute in the employee data that holds the manager information. In the sample below, the managerEmail attribute is used.

Person Data	Mapped Attributes
* First Name	givenName
* Last Name	sn
* Login ID	sAMAccountName
* Person Identification	sAMAccountName
* Email Address	mail
* Home Organizational Unit	department
* Password	sn
<input type="checkbox"/> Optional Person Data Mappings	
Person Data	Mapped Attributes
Title	
Social Security Number	
Birthdate	
Hire Date	
Timezone ID	
Locale ID	
Employee Code	
Supervisor	managerEmail

- In the Import Manager settings, specify as the “Key Field for Manager ID” the field in the manager’s directory record whose value corresponds to the Supervisor attribute specified for the original person.

In one possible scenario, a single attribute exists in each person's directory record which uniquely identifies the person's supervisor. Assume, for example, that the person's directory record contains the manager’s email ID within the attribute **manager_email**. No other manager information is present

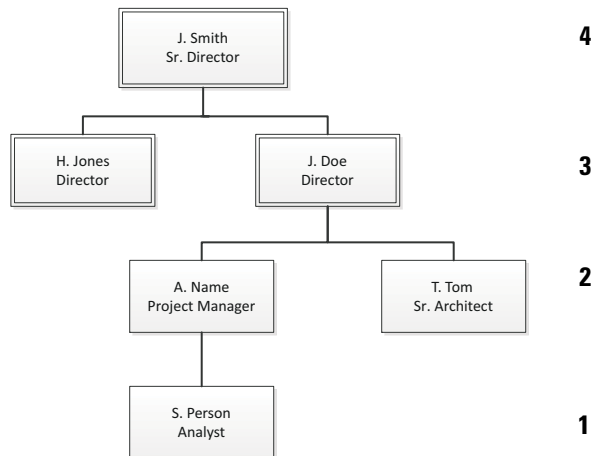
Solution	Supervisor	manager_email
	Key Field for Manager ID	email (the email attribute in the manager's directory record)

An alternative scenario may be that the directory record contains an attribute that is exactly the DN of the person's supervisor. Assume the name of this attribute is **manager**.

Solution	Supervisor	manager
	Key Field for Manager ID	dn (DN is a special attribute and is not prefixed before the search string)

Supervisory hierarchies may also need to be accommodated.

For example, consider this organizational chart:



1	Level 1	3	Level 3
2	Level 2	4	Level 4

If requests were subject to an immediate supervisor’s approval, a “relative” search is needed, going up the tree one level.

Alternatively, if certain requests were subject to, for example, a Director’s approval, an “absolute” search is needed. People (managers) would be imported until the position of the current person was “Director”. In the example above, in the case of S. Person, two additional people would be needed—her immediate manager, A. Name, and his manager, J. Doe, who is their Director. For T. Tom, only one import would be required.

If you are using an absolute search (import all managers with successively higher levels of authority until you find one with the specified position), you must assign numeric equivalents to the positions:

- Analyze the corporate hierarchy, assigning numeric equivalents to all management positions.
- Identify the attribute in the employee’s directory entry that is to designate his/her management level. For example, perhaps an attribute named “paygrade” could be used.
- For all employees, ensure that the designated attribute is populated.
- In the mapping for the Management Level field (listed in the Optional Person Data Mappings) specify the attribute that holds this information.
- Enter the highest level of manager to be imported as the “Maximum Level” in the Import Manager settings.

You may configure a search terminator if you do not want to synchronize supervisors beyond a known value. You can specify multiple values in the format: #value1#, #value2# and so on.

For example, you may not want to import any supervisors who rank above a person with uid as “scarter.” His Supervisor attribute is mapped to his email (scarter@email.com). In this case set the Search Terminator to #scarter@email.com#. The directory integration will stop supervisor synchronization as soon as a record is found with scarter@email.com as the supervisor.

Supervisor synchronization stops as soon as either limiting condition is met—Maximum Level or Search Terminator.

Setting	Value	Comments
Key Field for Manager ID		The directory attribute that uniquely identifies the employee's manager (supervisor).
Maximum Level		For absolute searches, indicates the number of managers above the current employee that need to be imported; for relative searches, indicates the highest management level for a manager to be imported.
Search Mode	<ul style="list-style-type: none"> Absolute Relative 	
Search Terminator		The value or values that match the key field for managers that stop the search.
Refresh options	<ul style="list-style-type: none"> Refresh Person Profile Refresh Period (Hours) 	Check the Refresh Person Profile check box to indicate that the manager's profile within Request Center is to be refreshed. If the Refresh Period is left blank, the profile is refreshed every time the Import Manager event takes place for the same person. If a number is provided, the manager's profile is refreshed only once within the specified period.
Associations	<ul style="list-style-type: none"> Do Not Create Organizational Unit Do Not Create Group 	Identical to settings for Import Person.
Remove Existing Association	<ul style="list-style-type: none"> Organizational Unit Group Role 	Identical to settings for Import Person.

Custom Code Operations

Use a custom code operation to invoke routines not supported by the application. A custom code operation may replace or supplement Service Portal operations.

Setting	Value	Comments
Custom Code Operation Type	<ul style="list-style-type: none"> Single Sign-On External Authentication Import Person Import Manager Custom Code Person Search 	Use a Java class to provide the name of your mapping. For more details about the Java class see the Javadocs.

Configuring Directory Integration

Configuring directory integration involves using the Administration module's Directories options. The basic process is to:

- **Enable directory integration.** Click the **Directory Integration** radio button on the Administration module's Settings tab to enable directory integration.
- **Configure datasource information.** Use the Datasources area of the Administration module's Directories tab to configure datasources that connect to directory servers. Information such as the datasource name, description, protocol, server product, and authentication method is required.
- **Configure mapping.** Use the Mappings area of the Administration module's Directories tab to map application data to the directory server data. Mappings update the entire user/person's profile along with all related entities: addresses, contacts, locations, one or more group associations, one or more organizational unit (OU) associations, and one or more role associations.
- **Configure events.** Use the Events area of the Administration module's Directories tab to configure directory integration behavior. The Login and Person Lookup events can be configured to include operations such as Single Sign-On (SSO), End User Authentication (EUA), Import Person, Import Manager, and Person Search.
- If required, **configure custom code interfaces** for client customizations, including directory java class attribute mapping, directory server API, and Import Person, with its related entities.

Enabling Directory Integration

To enable directory integration:

-
- Step 1** Log in using an account with administrative privileges and choose the **Administration** module.
 - Step 2** Click the **Settings** tab.
 - Step 3** Next to Directory Integration, click the **On** radio button.
 - Step 4** On the bottom of the Customizations screen, click **Update**.
- You have now enabled directory integration. (See [Figure 1-4](#).)
-

Figure 1-4 Enabling Directory Integration

The screenshot shows the Cisco Service Portal Administration interface. The top navigation bar includes 'Home', 'Directories', 'Authorizations', 'Notifications', 'Lists', 'Settings', and 'Utilities'. The 'Settings' tab is selected and highlighted with a red box and the number 2. The main content area is titled 'Customizations' and contains a table of settings. The 'Directory Integration' setting is highlighted with a red box and the number 3. A red box with a question mark points to the 'Customizations' section header.

Setting	Setting Value	Description
KpiSourceOfData:	Datamart	This setting controls where the KPI charts retrieve data.
SessionTimeout:	20	Set the session timeout.
Fiscal Year End:	Month: Dec Day: 31	Sets the month and day of fiscal year end for fiscal calendar related calculations.
Attachment Maximum Size:	0 KB	Sets the maximum size of the file that can be uploaded as an attachment (0 indicates no maximum size).
Attachment File Type Restrictions:	None	Defines the file types that are allowed/prevented. Specify these as a list of file extensions separated by comma; for example: .exe,.bmp,.zip
Order Confirmation Email Template:	None	An email will be sent when a customer submits a requisition.
Order Failure Email Template:	None	Email to be sent if the order submission process fails unexpectedly. This entry takes effect only if the <i>Submit, Approve and Review Tasks Asynchronously</i> setting is on.
Approval Failure Email Template:	None	Email to be sent if an approval or review task performed by the user fails unexpectedly. This entry takes effect only if <i>Submit, Approve and Review Tasks Asynchronously</i> setting is on.
Maximum number of results returned by non-Directory-enabled Person Popup:	1000	Maximum number of people returned when end-users attempt 'select (*)' type queries in non-Directory-enabled Person Popup dialogs by entering only wildcard characters (default is 1000 people; 0 indicates all people)
Browser Cache:	Enabled Disabled	The Browser Cache setting enables the browser-side caching of images, JavaScripts, css, etc., which may improve performance. When the Version setting value is incremented, the login process is interrupted until the browser's cache is deleted. Default is Disabled.
Version:	0	

Update

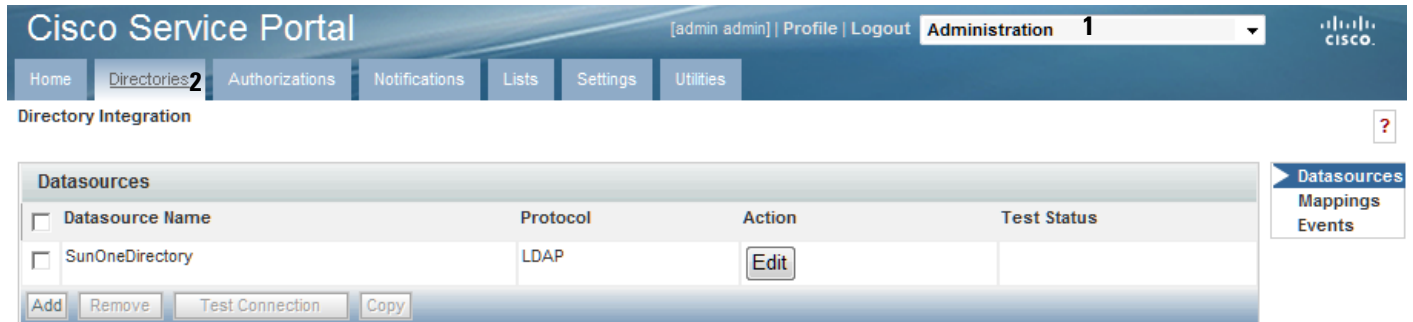
On	Off	Setting	Description
<input type="radio"/>	<input checked="" type="radio"/>	Enable Custom Header Footer	Site will add content from the custom header and footer HTML. Default is off.
<input type="radio"/>	<input checked="" type="radio"/>	Enable Custom Style Sheets	Site will utilize the custom stylesheet allowing for the changing of logos, color schemes, fonts and others. Default is off.
<input type="radio"/>	<input checked="" type="radio"/>	Directory Integration 3	Enable the Directories feature that searches for and imports users into the site from an external datasource (e.g., LDAP). Default is off.

1	Administration module	3	Directory Integration setting
2	Settings tab		

Configuring Directory Integration

You use the Directories tab of the Administration module to configure many of the directory integration settings.

Figure 1-5 The Directory Integration Area



1	Administration module
2	Directories tab

-
- Step 1** Log in using an account with administrative privileges.
- Step 2** From the drop-down menu, choose **Administration**.
- Step 3** Click the **Directories** tab.

The Directory Integration page appears. These settings will be in effect once directory integration has been enabled.

Configuring Datasource Information

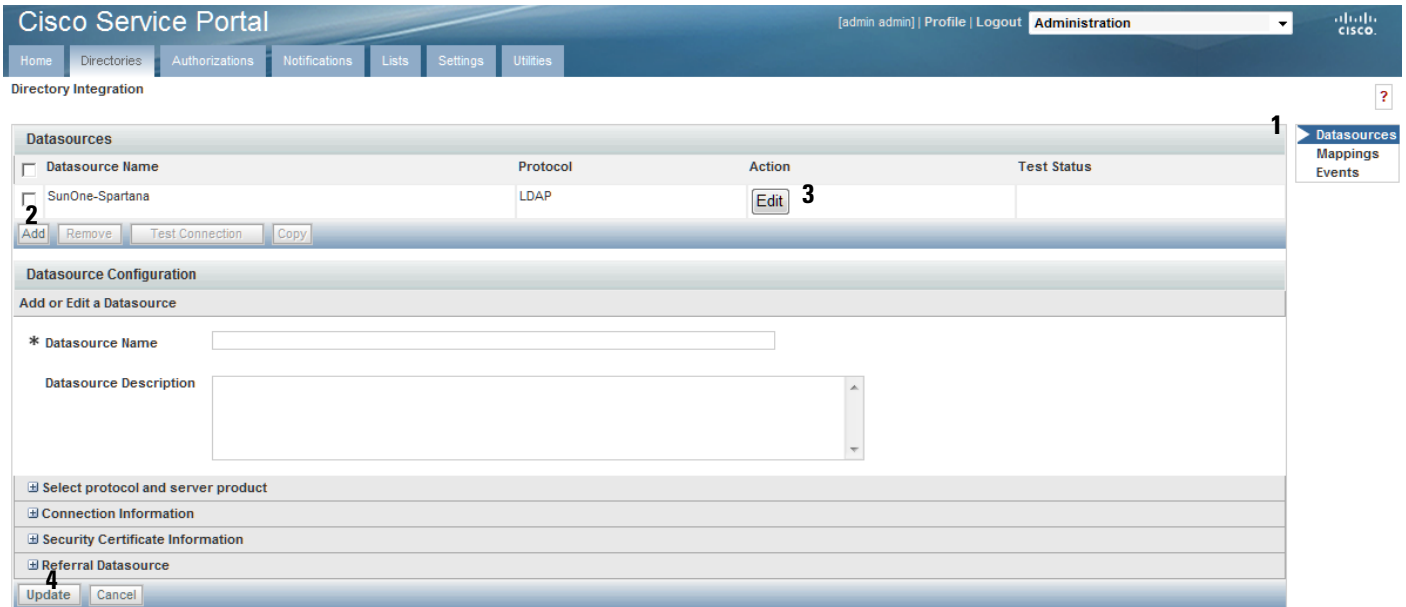
The following sections guide you through configuring datasource specific information. The tasks include:

- **Adding or editing a datasource** – You need to add a datasource to a new installation that does not yet have any datasources. If datasources exist, you may edit them.
- **Adding a server certificate for SSL connections** – You only need to do this if you choose SSL as the connection mechanism.
- **Adding referral datasources** – Only if desired.
- **Testing the connection** – You should always test the connection to prove connectivity.

Adding or Editing a Datasource

At least one datasource must be defined. To add a new datasource:

Figure 1-6 Adding or Editing a Datasource



1	Datasources option	3	Edit Datasource button
2	Add Datasource button	4	Update button

-
- Step 1** Navigate to the Directory Integration page by choosing the **Administration** module and then clicking the **Directories** tab.
- Step 2** In the page navigator, click the **Datasources** option, if not already selected.
- Step 3** Click **Add**. To edit an existing datasource instead of adding a new datasource, click **Edit** next to the desired datasource in the list.
- The Datasource Configuration area expands.
- Step 4** Enter the Datasource Name, Datasource Description, and the desired settings. Click the **+** buttons to access all of the settings in the adjacent area. See the Datasource Worksheet for more information about these settings, or see the following sections.
- Step 5** Click **Update**.
-

Configuring Connection Information

Specify the connection protocol and user credentials used to connect to the datasource.

Figure 1-7 Configuring Connection Information

The screenshot shows a configuration window titled "Connection Information". It contains several fields and dropdown menus:

- * Authentication Method:** A dropdown menu set to "Simple".
- * Mechanism:** A dropdown menu set to "Non SSL".
- * BindDN:** An empty text input field.
- * Host:** An empty text input field.
- * Port Number:** A text input field containing the value "0".
- * Password:** An empty text input field.
- * User BaseDN:** An empty text input field.
- Optional Filter:** An empty text input field.

Configuring Certificates

If you chose SSL as the connection mechanism, you need to specify the certificates for the directory integration system.

Figure 1-8 Configuring Security Certificates

The screenshot shows a configuration window titled "Security Certificate Information". It features a table with the following columns: "Certificate Name", "Certificate Type", and "Action".

Certificate Name	Certificate Type	Action
cert001 2	Server 3	Hide Certificate Value

Below the table, the following information is displayed:

Issuer DN: CN=ceberus1.oakqas.celosis.com
Subject DN: CN=ceberus1.oakqas.celosis.com

-----BEGIN CERTIFICATE-----
MID4jCCAsqgAwIBAgIQQNHYIDz0pYRDCOssJZrRQjANBgkqhkiG9w0BAQUFADAmMSQwlgYDVQQDD
ExtjZWJlcnVzMS5vYWxYXMuY2Vsb3Npcy5jb20wHhcNMTEwOTAyMTg1MzA5WmcNMjEwOTAyMTg1
ODEzWjAmMSQwlgYDVQQDEExtjZWJlcnVzMS5vYWxYXMuY2Vsb3Npcy5jb20wggEIIA0GCsqGSib3
DQEBAQUAA4IBDwAwggEKAoIBAQCb23MTF9y6zwMzqtii69Lj+knIZf7OzJkYlm3Hfw0OVl6YV5J

4

At the bottom, there are two buttons: "Add certificate **1**" and "Remove Certificate".

1	Add certificate button	3	Certificate Type drop-down menu
2	Certificate Name field	4	Certificate Value field

- Step 1** Navigate to the Directory Integration page by choosing the **Administration** module and then clicking the **Directories** tab.
- Step 2** In the page navigator, click the **Datasources** option, if not already selected.
- Step 3** Next to the datasource to which you wish to add a certificate, click **Edit**.
- Step 4** Click **Add Certificate**.
- Step 5** Name the certificate. Do not use spaces or special characters in the certificate alias name.
- Step 6** From the Certificate Type drop-down menu, choose the certificate type.
- Step 7** Paste the certificate value (obtained from a vendor like VeriSign) into the certificate field.
- Step 8** Click **Update**.

Configuring Referral Datasources

If you have multiple datasources configured, you can designate datasources as referral systems to a selected datasource. This way, whenever the system performs a search against the selected datasources, it will also search all referral datasources.

The referral datasources are searched in the order in which they are specified until a match is found. A match is said to be found when the search criteria returns one or more records.

Referral datasources are typically used when directory information is divided among multiple directories. For example, different company divisions may each maintain their own directory.

Figure 1-9 Configuring Referral Datasources

Referral Datasource			
<input type="checkbox"/>	Sequence	Datasource Name	Mapping Name
<input type="checkbox"/>	1	2	3

Add Referral **1** Remove Referral

1	Add Referral button	3	Mapping Name drop-down menu
2	Datasource Name drop-down menu		

-
- Step 1** Navigate to the Directory Integration page of the Administration module.
 - Step 2** In the page navigator, click the **Datasources** option, if not already selected.
 - Step 3** Next to the datasource for which to configure a referral datasource, click **Edit**.
 - Step 4** Click **Add Referral**.
 - Step 5** The Referral Datasource area appears. From the Datasource Name drop-down menu choose a datasource name, and then from the Mapping Name drop-down menu choose a mapping name.
 - Step 6** Click **Update**.
-

Testing the Connection

If you have completed all the necessary configuration steps, then you are ready to test the directory integration connection.

Figure 1-10 Testing the Connection


The screenshot shows the Cisco Service Portal Administration interface. The top navigation bar includes 'Home', 'Directories', 'Authorizations', 'Notifications', 'Lists', 'Settings', and 'Utilities'. The 'Administration' module is selected. The 'Directory Integration' section is active, showing a table of Datasources. The table has the following structure:

Datasource Name	Protocol	Action	Test Status
<input type="checkbox"/> SunOne-Spartana	LDAP	Edit	2

Below the table are buttons: Add, Remove, Test Connection (1), and Copy. A legend below the screenshot identifies '1' as the Test Connection button and '2' as the Test Status column.

1	Test Connection button	2	Test Status column
---	------------------------	---	--------------------

- Step 1** Navigate to the Directory Integration page in the Administration module.
- Step 2** In the page navigator, click the **Datasources** option, if not already selected.
- Step 3** Choose the datasource to test by checking the check box to the left of the datasource name.
- Step 4** Click **Test Connection**.

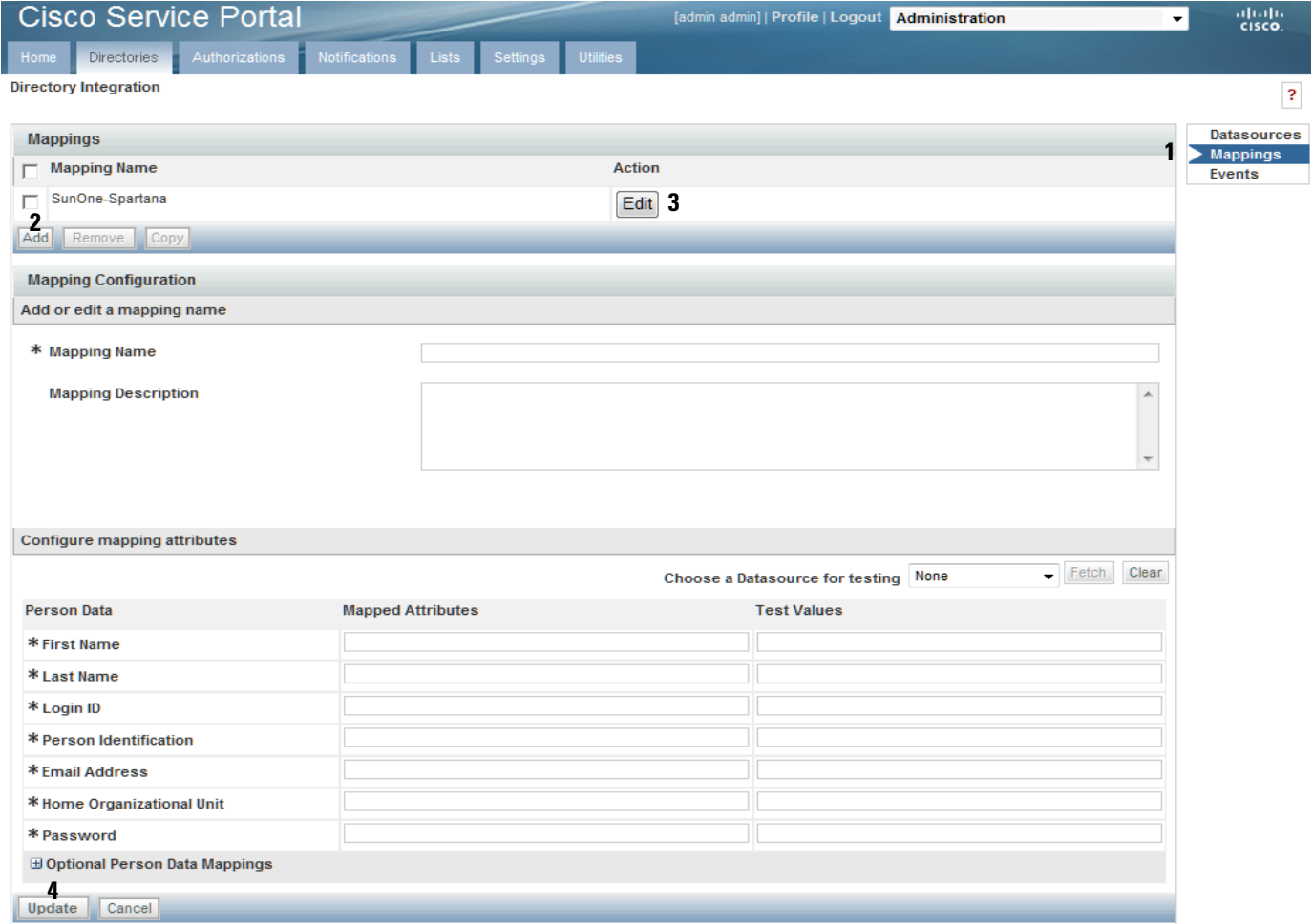
The Test Status column displays OK if the connection is successful, and  if it is unsuccessful.

Configuring Mappings


You use the Mappings area of the Administration module's Directories tab to map Service Portal data to directory server data.

To configure mapping, see [Figure 1-11](#) and follow the procedure below.

Figure 1-11 Configuring Mapping



1	Mappings options	3	Edit Mapping button
2	Add Mapping button	4	Update button

- Step 1** Navigate to the Directory Integration page of the Administration module.
- Step 2** In the page navigator, click the **Mappings** option.
- Step 3** Click **Add** to add a new mapping, or click **Edit** next to the desired mapping in the list to edit an existing mapping.
The Mapping Configuration area expands.
- Step 4** Configure the mapping name, description, and attributes, based on the requirements documented in the Mapping Worksheet. The mappings prefixed with an asterisk (*), shown in the Person Data section, are mandatory. You may also configure optional mappings by clicking the  button, to expand the Optional Person Data Mappings section.
- Step 5** Click **Update**.

The mapping fields accept simple, composite, expression, and Java mapping types, as described below.

Mapping Types

This section describes accepted mapping types, illustrates a valid sample mapping, and explains with examples expression mapping. The following table describes the supported mapping types.

Table 1-1 Mapping Types

Mapping Type	Description
Simple	One directory attribute maps to the field. This is simple one-to-one mapping. For example: Person Field: First Name Directory Attribute: givenName
Composite	A combination of attributes maps to the field. # delimits each attribute name. The mapping may include literals. For example: Person Field: Email Directory Attributes: #givenName#_#sn#@#domain#.com
Expression	An expression uses regular expressions and pattern matching to derive the mapping. For more details see the “Expression Mapping” section on page 1-28 .
Java Class	Use Java mapping when simple, composite, or expression mapping does not offer the desired functionality. This involves writing a Java class and placing the compiled class file on the appropriate directory on the application server. For more details see the “Java Class Mapping” section on page 1-31 .

Simple and Composite Mappings

The following table illustrates sample simple and composite mappings for the mandatory fields.

Table 1-2 Sample Mapping

Person Data	Directory Value
First Name	givenName
Last Name	sn
Login ID	uid
Person Identification	uid
Email Address	#givenName#_#sn#@#company#.com
Home Organizational Unit	Ou
Password	Uid

Expression Mapping

Expression mapping allows you to conditionally assign a value to an attribute, based on which pattern (regular expression) the expression matches. The system expression mapping uses the Perl5 Regular Expression Language, to specify patterns to be matched, combined with syntax similar to that of the Java conditional operator. Syntax:

```
expr:<expression>=<patternlist>?(<valuelist>):<default>
```

where

expr	is a prefix to indicate that expression mapping is used.
<expression>	is the expression to match against.
<patternlist>	is a set of patterns, separated by a pipe ().
<valuelist>	is a set of values, separated by a pipe (), corresponding to the set of patterns. Each value designates the return value if the expression matches the corresponding pattern.
<default>	is the return value to use if no pattern in the <patternlist> matches the <expression>.

For example:

```
expr:<expression>=
(<pattern1>|<pattern2>...<patternn>)?(<value1> | <value2> <valuen>):<default>
```

If <expression> matches <pattern1>, then return <value1>.

If <expression> matches <pattern2>, then return <value2>.

If <expression> does not match any pattern, then return <default>.

Each element (expression, pattern, or value) can contain a directory attribute name, delimited by the # symbol. For example, a pattern can be specified as “#givenName#_#sn#”, where both #givenName# and #sn# are attribute names:

In addition, parentheses can be used to group a series of pattern elements to a single element. When you match a pattern within parentheses, you can use back-references, in the form of \$1, \$2, and so on, to refer to the previously matched pattern.

Examples of Expression Data Mapping

A simple use of an expression applied to directory integration may be to translate one or more coded values in the directory to more user friendly descriptions or broader categories. For example, some services may need to differentiate between employees and contractors. The costCenter attribute is known to be “000000” for contractors. Therefore, the following expression could be applied to the “Employee Type” field:

```
expr:#costCenter#=(000000)?(Contractor):Employee
```

Another straightforward use of an expression may be to supply a default value for a field when the source attribute is blank. This may frequently be a “stop gap” measure, until directory data can be standardized. Or it could be standard; for example, if outside contractors are not assigned a department. The following expression could be applied to the “Home OU” field (a mandatory field for the mapping):

```
expr:#DeptLevel2#=(.+)?(#DeptLevel2#):Contractors
```

This expression uses the DeptLevel2 attribute if available, or defaults to the “Unknown” Business Unit for the user’s Home OU.

Similarly, the expression can be used to translate from a set of input values to a different set of return values. This is the equivalent of a case statement, or nested if/then construct. For example, the following expression could be applied to the “Locale ID” field, to assign a language for the user, based on his/her location:

```
expr:#country#=(United States | Germany)?(en_US | de_DE):en_US
```

If the user’s country is the United States, set the language to American English; if it is Germany, set the language to German. For any other country, set the language to American English.

Regular expressions can check the length of a source attribute and whether it is composed of alphabetic or numeric characters. For example, sometimes zip codes are stored as numeric data types, truncating leading zeroes. To restore a leading zero, an expression such as the following could be applied to the “Company Postal Code” field:

```
expr:#postalCode#=(^[1-9][0-9][0-9][0-9]$)?(0#postalCode#):#postalCode#
```

If the postalCode attribute consists of precisely four digits, add a leading zero to the value of the attribute. This converts zip code **1701** to **01701**, and leaves any source values which do not match the specified pattern unchanged.

A similar use of regular expressions might check that the format of an attribute value matches an expected pattern. Consider a use case in which a valid manager's user ID needs to consist of two letters followed by a series of numbers. Valid IDs would be, for example, fd1024 and ID3839. The following expression could be used:

```
expr:#manager#=(cn=([a-zA-Z][a-zA-Z][0-9]+),.*)?($1):None
```

Attributes can be used in the expression, pattern, or return value:

```
expr:#sn#, #givenname#=(Smith.*|Doe, John)?(All Smiths|Only John):Others
```

```
expr:#sn#, #givenname#=(Smith.*|Doe, John)?(#givenname#|Only John):Others
```

The last name and first name from directory records are combined into a string such as “Doe, Jane” before any attempt is made to match the patterns.

Embedded parentheses and back-references are useful for extracting a portion of the pattern. For example, the organization to which a person belongs is frequently embedded within a distinguished name (dn) attribute:

```
dn: cn=plee,ou=Corporate,dc=InfoSys,dc=com
```

The expression mapped to the “Home Organizational Unit” field might have the format:

```
expr:#dn#=((cn=[^,]+,ou=([a-zA-Z]+),dc=InfoSys,dc=com)?($1):Default
```

The returned value, “Corporate” is a back-reference value \$1, which equals the pattern matched by the expression within the first set of parentheses, ([a-zA-Z]+).

Usage of back-referenced variables may be required to parse overloaded attributes which include the values for more than one field. For example, an attribute can include the business address of a person, including the building name, floor (level), and office.

```
location=Corporate Headquarters-Fifth Floor-Office #5F
```

The same pattern could be used to match the three elements in the expression, by using different back-referenced variables as the value:

Office Building	expr:#Location#=(([^-]+)-([^-]+)-(.*))?(\$1): Unknown
Building Level	expr:#Location#=(([^-]+)-([^-]+)-(.*))?(\$2): Unknown
Cubic Location	expr:#Location#=(([^-]+)-([^-]+)-(.*))?(\$3): Unknown

Java Class Mapping

You will need to be familiar with Java programming and have a Java development environment set up in order to implement a custom Java class to map directory data to fields.

Any custom mapping class must follow the guidelines given in “[Using Custom Code in Directory Integration](#)” section on page 1-36. The mapping class must implement an `IEUIAttributeMapping` interface.

The developer must follow the guidelines below to test and install the custom code module.

1. Install a Java IDE of choice, and set up a project for developing custom mapping code.
2. Edit the custom code file to fulfill your requirements.
3. Compile.
4. The custom Java class must be installed on the Service Portal web archive (war), to be accessible to the Request Center service. Create a directory in `RequestCenter.war/WEB-INF/classes` to correspond to the package. Such directories are typically named:
`com/newscaler/client/<clientname>`, for example, `com/newscaler/client/aib`.
5. Copy the `CustomMapping.class` file to the directory created in the previous step.
6. Restart the Request Center service.
7. Specify the fully qualified name of the class file as the Mapped Attribute for the field to be populated.
8. Test the custom code by using the Directories Test feature.
9. Save your source in an appropriate repository.

Testing Mappings

You can use the Mapping Test feature to test that your data mapping settings are configured correctly and pulling the correct values from the directory server.

Using the Data Mapping Test feature involves:

- Enabling the Data Mapping Test Feature
- Using the Data Mapping Test Controls

Enabling the Directory Map Testing Feature

To enable the directory map testing feature, see [Figure 1-12](#) and follow the procedure below.

Figure 1-12 Enabling Mapping Testing

On	Off	Setting	Description:
<input type="radio"/>	<input type="radio"/>	Debug	Turns general site debugging on or off.
<input checked="" type="radio"/>	<input type="radio"/>	Directory Map Testing	Enable or disable the test feature on the mappings page of Directory Integration

Update

1	Debugging option	3	Update button
2	Directory Map Testing setting		

Step 1 Click the **Settings** tab of the Administration module to display the Settings page.

Step 2 In the page navigator, click the **Debugging** option.

The Debug Settings page appears.

Step 3 Next to the Directory Map Testing setting, click the **On** radio button.

Step 4 Click **Update**.

The system enables the Data Mapping Test feature. Now when you access the Data Mapping tab, the following additional features appear as shown in Figure 1-13:

- The Choose a Datasource for Testing drop-down menu
- The Fetch button
- The Clear button
- The Test Values column

Using the Data Mapping Test Controls

Figure 1-13 Mapping Test Controls

The screenshot shows the Cisco Service Portal Administration interface for Directory Integration. The main content area is titled 'Mappings' and contains a table with columns for 'Mapping Name' and 'Action'. Two mappings are listed: 'ADS' and 'SUNONE', each with an 'Edit' button. Below the table are 'Add', 'Remove', and 'Copy' buttons. The 'Mapping Configuration' section allows adding or editing a mapping name and description. The 'Configure mapping attributes' section includes a 'Choose a Datasource for testing' dropdown menu (set to 'SUNONE') with 'Fetch' and 'Clear' buttons. A table below shows 'Person Data' (First Name, Last Name, Login ID, Person Identification, Email Address, Home Organizational Unit, Password) mapped to 'Mapped Attributes' (givenName, sn, uid, mail, ou, sn) and 'Test Values' (LDAP_SIMPLE_ATTRMAP_ERR). At the bottom, a 'Filter String' and 'Records Fetched' (1 of 27) summary is displayed.

1	Mappings option	4	Fetch button
2	Edit button	5	Test Values column
3	Choose a Datasource for testing drop-down menu	6	Test summary area

To use the Data Mapping test controls:

-
- Step 1** Click **Mappings**, if you are not already on the Mapping page.
 - Step 2** Next to the mapping you wish to test, click **Edit**.
 - Step 3** From the “Choose a Datasource for testing” drop-down menu, choose the desired datasource.

- Step 4** In the Test Values column, enter test values. You can use simple, composite, Java, or expression mapping.
- Step 5** Click **Fetch**.
- Step 6** The test values appear in the Test Values column and a summary of the results appears at the bottom of the page.



Note Fetch returns values from only one datasource and does not search referrals. This is for convenience because it becomes difficult to debug with referrals search integrated.

- Step 7** To the right of the Fetch button, click **Clear** and retry new values until you have configured the desired mappings.
-

Configuring Directory Integration Events

You use the Events area of the Administration module's Directories tab to configure directory integration behavior for the following events:

- Login
- Person Lookup for Order on Behalf
- Person Lookup for Service Form
- Person Lookup for Authorization Delegate

To configure events, see [Figure 1-14](#) and follow the procedure below.

- Step 1** Navigate to the Directory Integration page of the Administration module.
- Step 2** In the Page Navigator, click **Events** to display the Events page.
- Step 3** Next to the type of event to configure, click **Edit**.
The Event Configuration area appears.
- Step 4** From the Event Status drop-down menu, choose **Enabled** to enable the event.
- Step 5** Click **Add step** to add a step for the system to initiate when the selected event occurs.
- Step 6** Choose an operation associated with the step you just added.
- All operations are available in this menu even though some operations, such as SSO and EUA, are not applicable for all event types.
- Step 7** Click **Options** to configure the options associated with the operation you just chose. The Options area appears. The Options area will differ according to which operation is chosen. Details on the available operations and their options are given in the next section.
- Step 8** Configure the associated options. See the relevant sections in this chapter on directory Events for a description of the operations available and options for configuring them.
- Step 9** Click **Update** and repeat these steps for each step and operation you wish to add.
-

Figure 1-14 Configuring Events

The screenshot shows the Cisco Service Portal Administration interface. At the top, there is a navigation bar with 'Home', 'Directories', 'Authorizations', 'Notifications', 'Lists', 'Settings', and 'Utilities'. The main content area is titled 'Directory Integration' and contains an 'Events' section. This section includes a table of events with columns for Name, Status, and Action. Below this is an 'Event Configuration' form with fields for Event Name, Event Status, and a table of Event Steps. The 'Event Steps' table has columns for Event Step, Operation, Mapping, Datasource, and Additional Options. At the bottom of the configuration form are 'Update' and 'Cancel' buttons.

1	Events option	5	Operation drop-down menu
2	Edit button	6	Options button
3	Event Status drop-down menu	7	Update button
4	Add step button		

Using Custom Code in Directory Integration

The directory integration framework is designed for flexibility and customization of the “Login” and “Person Lookup” events.

Standard operations for all events are available on the Administration module’s Directories tab. These include: SSO, External User Authentication, Import Person, Import Manager, and Person Search.

In cases where these standard operations do not fully satisfy a business scenario, the Directories tab also provides interfaces to execute custom Java code. This custom code should adhere to the interfaces described in this chapter, and you should develop any customized solutions using Service Portal exposed APIs.

The following are valid use cases for scenarios in which you may wish to customize an event operation:

If...	Then...
The format of SSO headers input through the HttpServletRequest cannot be parsed ...	Provide a custom code SSO operation to retrieve user credentials, in order to support the SSO integration with your vendor.
You wish to authenticate a user via a web service or database other than Service Portal...	Provide a custom code External Authentication operation.
The main user repository in your company is a database other than an LDAP directory...	Provide custom code External Authentication and custom code Import Person operations.

The directory integration custom code framework also defines interfaces that can be implemented to provide complex retrieval logic for a specific field in the person/user profile from a record in an external datasource.

Public APIs and interfaces for directory integration include the:

- **Custom Code Operation Interfaces**, which are used to customize directory integration operations.
- **Custom Java Class Mapping Interface**, which is used to provide customized retrieval of a specific attribute in an external datasource from its record.
- **Directory Server API**, used to query/authenticate against an external datasource and retrieve records.
- **Import/Refresh Person API**, used to update person attributes in the Service Portal database.

A typical custom code project will involve following types of activities:

- Identify the need for custom code.
- Configure the Directories tab in the Administration module to include the Datasource to be used by your custom code and, if relevant, the Mappings which your custom code will use.
- Develop the custom code. You will need to understand the public APIs and interfaces provided by Cisco for directory integration tasks.
- Build and deploy the custom code.
- Configure the Directories tab in the Administration module to use your custom code.

[Table 1-3](#) below summarizes the directory integration operations in more detail.

Table 1-3 *Directory Integration Operations*

Operation	Purpose	Input	Output
Single Sign-On	Identifies the login name of the user	HttpServletRequest	Login Name
External Authentication	Authenticates a user against an external datasource	Login Name Password	Authenticity of User
Person Search	Retrieves the list of persons matching first name or last name	First Name and Last Name	List of Persons
Import Person	Imports a person into the Service Portal database from an external datasource	Login Name	Imported person information, including the managerID
Import Manager	Imports a manager or chain of managers into the Service Portal database from an external datasource	Imported person information including manager information	Managers are imported into the system

Mixing and matching, or replacing, standard operations with custom code operations is also supported by the directory integration framework. Service Portal supports various combinations of operations per event, as described in the table below, using your own customized code and Service Portal public APIs, designed to help implement these interfaces.

It is important that custom code design and development engineers understand the directory integration framework, public APIs, and custom code interfaces, which are discussed in detail in this chapter.

[Table 1-4](#) below portrays the relationship between methods, events, and operation types for custom code operations. Combinations not listed in [Table 1-4](#) below are not supported.

Table 1-4 *Custom Code Operations*

Event	Operation Type	Interface	Method
Login	SSO	ISignOn	getCredentials
	EUA	ISignOn	authenticate
	Import Person	ISignOn	importPerson
	Import Manager	ISignOn	importManager
	Custom Code	ISignOn	performCustom
Person Search for:	Person Search	IPersonSearch	getCredentials
	• Order On Behalf	IPersonSearch	importPerson
	• Authorization Delegate	IPersonSearch	importManager
	• Service Form	IPersonSearch	performCustom

Custom Code Operation Interfaces

If you are providing a custom implementation of an operation configured within an event, you will need to implement a “custom code operation interface”.

Custom code operation interfaces define callback methods that are invoked when a particular operation is triggered. Exactly which method is invoked depends on the operation type chosen in the operation. For more details see the Method, Event, and Operation Type for Custom code Operations table. All methods defined in the custom code operation interfaces follow the same pattern:

Parameters

In the following list, “**” must be replaced by the operation type, which is one of:

- IEUISignon
 - IEUIPersonSearch
1. ****OperationDTO**: This object contains the information on how you have set the operation on the Directories tab of the Administration module. It includes mapping and datasource information.
 2. ****OperationContext**: The Context object is used to share information across method invocations. The directory Integration framework makes information stored in one context object available to other context objects during the same HttpServletRequest invocation.
 - a. Use `setLocalContextObject` and `getLocalContextObject` to set any custom information that does not fall as a part of results.
 - b. Use `get**Result` to get a result object. Result objects contain all the information about what happened throughout the event request. Results contain information that is supported in a productized import. The LocalContext object is used to store objects that were unforeseen during the implementation of productized operations.
 3. **Request**: This is the HttpServletRequest.
 4. ****ImportAPI**: This object is used to import a person. More details can be found in the Javadocs.
 5. ****LDAPAPI**: This API is used to make LDAP queries. More details can be found in the Javadocs.

Return

****Result**. After performing the custom task the API must return a valid return type with results populated. Return the same result object retrieved from OperationContext after updating relevant properties. There may be unexpected behavior if a new instance of the result object is returned.

[Table 1-5](#) below maps the expected input/return to the objects in the parameters of each of these callback methods:

Table 1-5 *Input for Custom Code Callback Methods*

Information	Object/Property
HttpServletRequest	Request
Login Name	<ul style="list-style-type: none"> • IEUISignOnOperationContext .IEUISignOnOperationResult.ssoLoginId • IEUIPersonSearchOperationContext .IEUIPersonSearchOperationResult.ssoLoginId
First Name and LastName	<ul style="list-style-type: none"> • First Name: IEUIPersonSearchOperationContext. firstNameSearchString • Last Name: IEUIPersonSearchOperationContext. lastNameSearchString

Table 1-5 *Input for Custom Code Callback Methods*

Information	Object/Property
List of Persons	IEUIPersonSearchOperationContext.EUIPersonSearchOperationResult.SearchPersonList. SearchPersonList is a collection with all elements of type IExtPersonDTO
Imported Person Information	<ul style="list-style-type: none"> • IEUISignOnOperationContext .IEUISignOnOperationResult.ImportedPersonExtDTO • IEUIPersonSearchOperationContext.EUIPersonSearchOperationResult.ImportedPersonExtDTO
Manager Id	IEUIPersonSearchOperationContext.EUIPersonSearchOperationResult.ImportedPersonExtDTO.PersonDTO.managerID

You must implement all methods to compile your implementation class. If you customize only limited operation types, you must provide an empty implementation of methods not relevant to the operation types.

For example, if you are only interested in a customized SSO, then provide a complete implementation of the `getCredentials` method. For all other methods, return null.

The system may pool an instance of an interface and may be concurrently accessed from multiple threads. Thus, it is recommended to keep the instance stateless.

There are two types of custom code operation interfaces:

- **ISignOn** is used for customizing the login.
- **IPersonSearch** is used for customizing the “Person Lookup” dialog box.

Custom Code Interface for Login Event – ISignOn

This is the interface that custom code should implement in order to customize login events: SSO, EUA, Import Person, Import Manager and custom code operations.

Customizing the SSO Operation

The primary purpose of an SSO custom code operation is to retrieve and return the Login Name from HttpHeader based Sign-On or from CGI Header (CGI variable `REMOTE_USER`) in the case of Remote NTLM/IWA type of Sign-On.

As outlined in [Table 1-4](#), you must provide a Java class that implements the `ISignOn` interface. Please provide a complete implementation of the `getCredentials` method in this interface, and read the documentation for the `ISignOn` interface for detailed specifications.

The following are some guidelines for implementing the `getCredentials` method. It is not required that all of these guidelines are implemented; There may be additional requirements, dependent on the customization, which are not covered by what is outlined below.

- Get `IEUISignOnOperationResult` from `IEUISignOnOperationContext`. This is the object that must be returned from this interface.
- Use the parameter `request` and process it to derive the login name of the person.
- Return `LoginId` back by calling `IEUISignOnOperationResult.setSsoLoginId(<login id>)`, if using the in-product directory lookup functionality.

- Call `IEUISignOnOperationResult.setSsoRedirectUrl("<any url or error page>")`, which are used for redirecting the user on SSO failure.

SSO Operation Options received through `IEUIEventSSOOperationDTO.getEventSsoDTO()` may be null as SSO options are not accepted in the Administration module for custom code operations.

Customizing the EUA Operation for Login Event

The primary purpose of an EUA custom code operation is to authenticate a user against an external system.

As outlined in [Table 1-4](#), you must provide a Java class that implements the `ISignOn` interface. Please provide a complete implementation of the `authenticate` method in this interface, and read the documentation for the `ISignOn` interface for detailed specifications.

The following are some guidelines for implementing an EUA operation. It is not required that all of these guidelines are implemented; There may be additional requirements, depending on the customization, that are not covered below.

- Get `IEUISignOnOperationResult` from `IEUISignOnOperationContext`. This is the object that must be returned from this interface.
- The `EUIDatasourceDTO` object from the `IEUIEventEUAOperationDTO` object contains the interface to the `Datasource` configured in the Administration module for this operation.
- Populate the `LDAPConfigInfo` object from the `EUIUtil` and pass `EUIDatasourceDTO`. This is needed to call LDAP API with the connection information to LDAP Server.
- Get the Login Name by calling `IEUISignOnOperationResult.getSsoLoginId()`.
- Form a `BindDN` and set it into `LDAPConfigInfo` by calling `setBindDN()`.
- Get the Password entered by the user in the Login page by calling `IEUISignOnOperationResult.getEuaPassword()`.
- Set it into `LDAPConfigInfo` by calling `setBindPassword()`.
- Authenticate the user against the Directory Server by passing the `LDAPConfigInfo` object `ILDAPApi.authenticate()` API.
- If the user has been authenticated, then call `IEUISignOnOperationResult.setEuaAuthenticated(true)`.
- If the user authentication failed or any exception occurred, then call `IEUISignOnOperationResult.setEuaAuthenticated(false)`.

EUA Operation Options received through `IEUIEventEUAOperationDTO.getEventEuaDTO()` will be empty as EUA options are not accepted in the Administration module for custom code operations.

Customizing the Import Person Operation for the Login Event

The primary purpose of the Import Person operation is to import/refresh a user from an external system, like a directory server or an external database, into the Service Portal application.

As outlined in [Table 1-4](#), you must provide a Java class that implements the `ISignOn` interface. Please provide a complete implementation of the `importPerson` method in this interface, and read the documentation for the `ISignOn` interface for detailed specifications.

The following are some guidelines for implementing an Import Person operation. It is not required that all of these guidelines are implemented; There may be additional requirements, dependent on the customization, which are not covered below.

- Get `IEUISignOnOperationResult` from `IEUISignOnOperationContext`. This is the object that must be returned from this interface.
- The `EUIDatasourceDTO` object from the `IEUIEventImportPersonOperationDTO` object contains the interface to the datasource configured in the Administration module for this operation.
- The `EUIDataMappingDTO` object from the `IEUIEventImportPersonOperationDTO` object contains the interface to the mapping configured in the Administration module for this operation.
- Using the Login Name retrieved from the `IEUISignOnOperationResult.getSsoLoginId()` method, query for the user on the external system either from an LDAP server or an external database and collect all information related to the Person profile, including organizational units, groups, and roles.
- Check to see if the user already exists in the Service Portal database by calling `ISignOnImportPersonAPI.getPersonByLoginName(<Login Id>)`. If the person already exists, this method returns the `IPersonDTO` object. If the person does not exist, the method throws a `signOnImportPersonAPIException`.
- If the person is not found, create an `IPersonDTO` object through the `PersonFactory.createPersonDTO()` method in preparation for importing the person.
- From the data fetched from the external system, create these DTOs using `PersonFactory` and populate them as well: `IPersonDTO`, `ILoginInfo`, `IContactDTO`, `IAddressDTO`, and `IPersonExtensionDTO`.
- Begin the database transaction by calling `ISignOnImportPersonAPI.beginTransaction()`.
- Check to see if an organizational unit (OU) exists by calling `ISignOnImportPersonAPI.getOrgUnitByName(<OU Name>)`. If it does, this method returns an `IOrganizationalUnitDTO` object. If the organizational unit does not exist, the method throws a `signOnImportPersonAPIException`.
- If an OU does not exist, it may be created by calling `ISignOnImportPersonAPI.createOrgUnit(<IOrganizationalUnitDTO>)`.
- If the user already exists, call `ISignOnImportPersonAPI.updatePerson(<IPersonDTO>)`. This updates a person's basic profile, login information, preferences, Home OU and extensions.
- If the user already exists, link/update addresses/location and contacts by calling `ISignOnImportPersonAPI.linkAddresses(<IAddressDTO collection>)` and `ISignOnImportPersonAPI.linkContact(<IContactDTO>)`.
- If the person is associated with one or more groups in the external system, first try getting all the existing groups by calling `ISignOnImportPersonAPI.getGroupByName (<ou name>)`. If not, create all the new groups by calling `ISignOnImportPersonAPI.createGroup(<IOrganizationalUnitDTO>)`.
- If the person is new, link all the lists of OUs and groups to the user by calling `ISignOnImportPersonAPI.linkPersonToOrgUnit()` and `ISignOnImportPersonAPI.linkPersonToGroup()`.
- If the person already exists, any OUs and groups may be unlinked from the user by calling `ISignOnImportPersonAPI.unlinkPersonToOrgUnit()` and `ISignOnImportPersonAPI.unlinkPersonToGroup()`.
- To find out the existing associations of OUs, including the home OU, and groups for a person, call the `ISignOnImportPersonAPI.getOrgUnitsForPerson()` and `ISignOnImportPersonAPI.getGroupsForPerson()` methods.
- To find out the existing associations to roles, call the `ISignOnImportPersonAPI.getRolesForPerson()` method.

- If the imported person needs to be associated with a role, first get the role using `ISignOnImportPersonAPI.getRBACRoleByLogicName(<roleLogicName>)`.
- Link/unlink roles to a person by calling `ISignOnImportPersonAPI.linkPersonToRole()` or `ISignOnImportPersonAPI.unlinkPersonToRole()`.
- If the person was imported/refreshed successfully, set the flag `ImportPersonDone = true` into `IEUISignOnOperationResult`.
- After successful import/refresh, also create an object of `IExtUserDTO` through `PersonFactory.createExtUserDTO()` and set `IPersonDTO` and `HomeOUDTO` (`IOrganizationalUnitDTO`) into `IExtUserDTO`, then return the `IExtUserDTO` of the imported person by calling `IEUISignOnOperationResult.setImportedPersonExtDTO(<IExtUserDTO>)`.
- If the import/refresh operation failed, set the flag `ImportPersonDone = false` into `IEUISignOnOperationResult`.
- End/commit the database transaction by calling `ISignOnImportPersonAPI.commitTransaction()`.
- If the transaction failed, roll back the transaction in the `exception` block by calling `ISignOnImportPersonAPI.rollbackTransaction()` and releasing the transaction in the `finally` block by calling `ISignOnImportPersonAPI.releaseTransaction()`.

Import Person operation options through the `IEUIEventImportPersonOperationDTO.getImportPersonDTO()` method will be empty as Import Person options are not accepted in the Administration module for custom code operations.

Customizing the Import Manager Operation for the Login Event

The primary purpose of the Import Manager operation is to import/refresh the Supervisor chain of the person from an external system, like a directory server, into Service Portal.

As outlined in [Table 1-4](#), you must provide a java class that implements the `ISignOn` interface. Please provide a complete implementation of the `importPerson` method in this interface, and read the documentation for the `ISignOn` interface for detailed specifications.

The following are some guidelines for the Import Manager operation:

- Get `IEUISignOnOperationResult` from `IEUISignOnOperationContext`. This is the object that must be returned from this interface.
- Get the user imported/refreshed user `ImportedPersonExtDTO` from `IEUISignOnOperationResult`.
- Get the Person who was imported through the `IEUISignOnOperationResult.getImportedPersonExtDTO()` method. This will return a `IExtUserDTO` object, from this get `IPersonDTO` object.
- Import all managers as needed from the external system, create/update each manager in the same way as explained in Import Person example above.
- Link a manager to a person, assuming `personDTO` is a reference to `IPersonDTO` for the imported manager and `managerDTO` is a reference to the `IPersonDTO` returned after the manager is imported.
- Use `personDTO.setManagerId(managerDTO.getId())` to set the manager association for `personDTO`.
- Save the association by saving `personDTO` using one of the mechanisms explained in the [“Import Person Operation”](#) section on page 1-16.

It is recommended that when importing the manager chain, you import the top level managers before persons. This avoids unnecessary updates for `personDTO` to update the link with the person’s manager.

Import Manager Operation Options received through `IEUIEventImportManagerOperationDTO`. `getImportManagerDTO()` will be empty as Import Manager Options are not accepted in the Administration module for custom code operations.

Customizing Custom Operations for the Login Event

The primary purpose of the custom code operation is to perform any custom operation that is needed and not represented elsewhere in the application.

The following are some guidelines for the Custom Code operation:

- Get `IEUISignOnOperationResult` from `IEUISignOnOperationContext`. This is the object that must be returned from this interface.
- Get `EUIDatasourceDTO` from `IEUIEventCustomOperationDTO`. This object contains the datasource configured in the Administration module for this operation.
- Get `EUIDataMappingDTO` from `IEUIEventCustomOperationDTO`. This object contains the mapping configured in the Administration module for this operation.
- Perform any custom operation as needed.
- `IEUISignOnOperationResult` should be populated appropriately based on previous examples.

Custom Code Interface for Person Lookup – `IPersonSearch`

This is the interface that a custom code should implement in order to customize Person Search events: Person Search, Import Person, Import Manager and custom code operations.

The implementation class is configured in the **Administration module > Directories tab > Events**, and can be configured for searching a for person in the following places within the Service Portal application:

- Person Search for Order On Behalf
- Person Search for Authorization Delegate
- Person Search for Service Form

Customizing the Person Search Operation

The primary purpose of the Person Search operation is to search for users from an external system, like a directory server.

As outlined in [Table 1-4](#), you must provide a Java class that implements the `IPersonSearch` interface. Please provide a complete implementation of the search method in this interface, and read the documentation for the `ISignOn` interface for detailed specifications.

The following are some guidelines for the Person Search operation:

- Get `IEUISignOnOperationResult` from `IEUISignOnOperationContext`. This is the object that must be returned from this interface.
- Since a custom Person Search operation can be configured using Person Search, we can add to, or manipulate, the search results from the previous operation in the Search Event by getting the list of persons already in the Search result by calling `IEUISignOnOperationResult.getSearchPersonList()`.
- Search the users on an external system, either a directory server using the API methods in the interface `ILDAPApi`, or in an external database using the API in `ISignOnImportPersonAPI` for connecting to SQL datasources.
- For every person found on the external system, create `IExtUserDTO`.

- Populate IExtUserDTO with IPersonDTO, IOrganizationalUnitDTO (for Home OU) and ILoginInfoDTO.
- Optional – based on the person popup global setting, also populate collection IContactDTO, collection of IAddressDTO, IPersonExtensionDTO.
- Get the flag “All Users For Order On Behalf” using ISignOnImportPersonAPI getCustomParam(“ShowAllUsersForOrderOnBehalf”).
- To make the custom code consistent with the standard platform behavior, if the flag is Off, and any mandatory attributes are missing for the person, remove the entry. This will prevent any incomplete persons from being shown in the popup.
- To make the custom code consistent with the standard platform behavior, if the flag is On and the Person is missing any mandatory attributes, call IExtUserDTO.setResultHasError(true). This includes the incomplete person in the popup, but displays a red asterisk “*” instead of the radio button. The starred user cannot be chosen by the end user or imported.
- Return the list of all persons searched by calling IEUISignOnOperationResult.setSearchPersonList(<List of all IExtUserDTO>).

Person Search Operation Options received through the IEUIEventPersonSearchOperationDTO.getPersonSearchOperationDTO() method will be empty as Person Search Options are not accepted in the Administration module for custom code operations.

Customizing the Import Person Operation for Person Search Event

As outlined in [Table 1-4](#), you must provide a Java class that implements the IPersonSearch interface. Please provide a complete implementation of the importPerson method in this interface, and read the documentation for the IPersonSearch interface for detailed specifications.

Steps to customize this are similar to the [“Customizing the Import Person Operation for the Login Event”](#) section on page 1-40.

Customizing the Import Manager Operation for Person Search Event

As outlined in [Table 1-4](#), you must provide a Java class that implements the IPersonSearch interface. Please provide a complete implementation of the search method in this interface, and read the documentation for the IPersonSearch interface for detailed specifications.

Steps to customize this are similar to the [“Customizing the Import Manager Operation for the Login Event”](#) section on page 1-42.

Customizing the Custom Operation for Person Search Event

As outlined in [Table 1-4](#), you must provide a Java class that implements the IPersonSearch interface. Please provide a complete implementation of the performCustom method in this interface, and read the documentation for the IPersonSearch interface for detailed specifications.

Steps to customize this are similar to the [“Customizing Custom Operations for the Login Event”](#) section on page 1-43.

Custom Java Class Mapping Interface

When simple, composite, or regular expression attribute mappings do not suffice, a custom Java class can be used in a directory integration attribute mapping.

Custom Java Class for Attribute Mapping – IEUIAttributeMapping

This is the interface that a custom code should implement in order to customize directory attribute mappings. The primary purpose of custom mapping class is to customize the attribute value fetched from the directory server.

The implementation class has to be configured in the **Administration module > Directories tab > Mappings**, and can be configured for any attribute in the mapping.

Figure 1-15 Custom Java Class for Attribute Mapping

Configure mapping attributes	
Person Data	Mapped Attributes
* First Name	<input type="text"/>
* Last Name	<input type="text"/>
* Login ID	<input type="text"/>
* Person Identification	<input type="text"/>
* Email Address	<input type="text"/>
* Home Organizational Unit	<input type="text"/>
* Password	<input type="text"/>
☐ Optional Person Data Mappings	
Person Data	Mapped Attributes
Title	<input type="text" value="com.newscale.bfw.eui.api.samples.custommapping.Custom"/>
Social Security Number	<input type="text"/>
Birthdate	<input type="text"/>
Hire Date	<input type="text"/>
Timezone ID	<input type="text"/>
Locale ID	<input type="text"/>

The following are some guidelines for using a custom Java class mapping class:

- The mapping class should only be used for simple logic to be applied to the value retrieved from the directory.
- For performance reasons, the mapping class should not be used to perform a call to a directory server using the Directory Server API or to execute any database operations. The Person Search or Login interfaces should be used for these use cases.
- Implement `IEUIAttributeMapping.getAttributeValue()` for returning a single value for the mapped attribute. This method should not be implemented for the OU List, Group List, or Role List mapping fields.
- Implement `IEUIAttributeMapping.getAttributeValueArray()` for returning multiple values for the mapped attribute. This method should only be implemented for the OU List, Group List, and Role List mapping fields.

Directory Server API

This is an API wrapper that Cisco provides for integrating with the directory server (LDAP) connection facility built into the product.

Authentication to, and querying, the directory server are the only features this API provides. This API supports all directory servers supported by Service Portal.

Typically, the Directory Server API works from the directory integration datasource and mapping configurations, and eliminates the need for hand-coding connection information, filters, and the attributes for querying.

Generally, to use the LDAP API, you also need the LDAPConfigInfo object. Use EUIUtil.getLDAPConfigInfo() from any datasource and mapping for this purpose.

The javadoc for LDAP API can be located in the javadocs folder of the product package.

Getting an Instance of ILDAPApi – API Implementation

An instance of ILDAPApi does not need to be created. It is available in all method arguments of both custom code API interfaces (ISignOn and IPersonSearch).

Directory Integration Utility (EUIUtil) Class

The directory integration utility class (EUIUtil) converts the datasource and mapping configured in the Administration module into a format that the Directory Server API can use as input for authentication, search, and query functions.

LDAP Configuration Info (LDAPConfigInfo) Class

An object of LDAPConfigInfo class encapsulates all the following configuration options that must be passed to the directory server API:

- Authentication information
- Connection information
- Query attributes
- Search filter

For more advanced users, if there is a need to override any configuration, LDAPConfigInfo provides getters and setters for all configurations. For further details on these methods, see the Javadoc for this class.

Main interface of the API – ILDAPApi

The ILDAPApi is the main interface that provides two basic operations on the directory server:

- Authenticate
- Search/Query

The ILDAPApi interface provides methods to interact with LDAP consistently throughout Service Portal.

LDAPEntryBean

After querying/searching the directory server using the ILDAPApi.query(...) method, the results are returned as a collection of LDAPEntryBean.

Import/Refresh Person API

This API can be used to import/refresh Person profile, create OUs or groups and also to link or unlink a person to an OU, group, or role. This API also supports transaction management for importing a person, and connectivity to SQL datasources. This API includes a method to read from the CnfParams table.

Import/Refresh Person API Interface – ISignOnImportPersonAPI

The Import/Refresh Person API interface provides methods for the following:

- Get a Person object by PersonID or LoginName. This returns the Person with login information, preferences, home OU, address, contact, location, and extensions.
- Create a Person with login information, preferences, home OU, address, contact, location, and extensions.
- Update a Person with login, preferences, home OU, and extensions.
- Get OU by OrganizationalUnitID, Name. This does not return the members of the OU.
- Get all the OUs for a given Person. This does not return the members of the OU.
- Create an OU.
- Link/unlink a Person with an OU.
- Get Group by GroupID, Name. This does not return all the members of the Group.
- Get all the groups for a given person.
- Create a group.
- Link/unlink a person with a group.
- Get a user-defined role by name.
- Get LogicName object for a system-defined role.
- Get system-defined role by LogicName object.
- Get all the roles for a given person.
- Link/unlink a person with a role.
- Link/update address or location for a person.
- Add/update/delete a contact for a person.
- Begin transaction, commit transaction and release transaction resources for Import Person.
- Get a connection to a SQL datasource.
- Rollback the transaction on the SQL datasource connection.
- Return the connection to the SQL datasource back to the connection pool.
- Get parameter values from the CnfParams table.

For further details see the Java documentation.

Customizing Java Class to Connect to a SQL Datasource

To customize the Java class to connect to a SQL datasource:

-
- Step 1** Get a connection to a SQL datasource database from `ISignOnImportPersonAPI` by passing the `DatasourceName`. The `DatasourceName` should be prefixed with the JNDI prefix, as defined by the “`DatasourceJNDIPrefix`” property in `newscale.properties` file.
 - Step 2** Use the above connection to execute any query using a JDBC statement.
 - Step 3** Commit the connection object directly at the end of the `try` block.
 - Step 4** Call `ISignOnImportPersonAPI` to roll back the connection when there are any failures/exceptions.
 - Step 5** In the final block, close the statement directly and call `ISignOnImportPersonAPI` to release the connection and return it to the connection pool.
-

Best Practices

Compiling Custom Code Java Files

The following are steps to compile and deploy custom code:

-
- Step 1** Copy the `build.xml` file given in the “[Sample build.xml File](#)” section on page 1-65 and paste it to any folder; for example, `C:\CustomCode`.
 - Step 2** Edit the `build.xml` file to change the property “`rcwar.dir`” to point to the full path where the `RequestCenter.war` is available.
 - Step 3** Edit the `build.xml` to change the property “`javax.servlet.dir`” to point to the full path where the `servlet-api.jar` is available. This is specific to the application server.
 - Step 4** Create a subfolder for the custom code java files; for example, `C:\CustomCode\src`.
 - Step 5** Create a custom code with a package name like “`com.newscale.SignOnCustomCode`” and place the `SignOnCustomCode.java` file in the following directory:
`C:\CustomCode\src\com\newscale\SignOnCustomCode.java`
 - Step 6** Run “`ant`” from a command line in the `C:\CusomCode` folder.
 - Step 7** The ant build file will compile all the java files under the “`src`” subfolder and place the class files in the “`out`” subfolder.
 - Step 8** The ant build file will also deploy the class files to the “`RequestCenter.war\WEB-INF\classes`” folder.
 - Step 9** Restart the application server.
-

Coding Guidelines

Package Names

- We recommend that the package name should be: `com.newscale.[yourcompanyname].*`.
- Use the key name “`com.yourcompanyname.*`” to store any `ContextLocalAttributes`. This eliminates clashes with the internal namespaces.

Logging

- Use the `Logger` to log messages to the server logs instead of using `System.out.println`.
- For debug logs, always begin by checking whether debugging is enabled. This is essential for performance.
- Always log the error in the exception block before propagating the exception back to the caller.

Exception Handling

- When `EUIException` is caught, throw it back as is.
- Wrap all other exceptions as `EUIException` and throw it back.

Configuring Custom Code in the Administration Module

After you have developed, compiled, and deployed the custom code, the Administration module must be configured to use the code. Configuration involves specifying when (in which event), in which operation and in what sequence (step) to invoke the custom code.

Step 1: Configure Global Settings

Ensure that the Directory Integration has been enabled by turning on this setting in the Administration module's Settings tab. Instructions for turning on Directory Integration are given in the [“Enabling Directory Integration”](#) section on page 1-20.

Step 2: Configure Datasources

Most operations, customized or not, require a datasource and mapping, so these two areas of the Directory Administration must be configured first.

Datasources are the external servers, such as LDAP, where your data is currently stored, and which Service Portal must access. The only custom operation which does not require a datasource is SSO.

See the [“Defining Datasources”](#) section on page 1-3 and the [“Configuring Datasource Information”](#) section on page 1-22 for more information on configuring datasources.

Step 3: Configure Attribute Mappings

Once you have set up the external datasource, you must map the person-related data available in Service Portal to the data in the LDAP directory (or other external datasource). These mappings tell Service Portal where to look and what to get during an event and sequence of operations.

To configure a mapping follow the guidelines and instructions in the [“Defining Mappings” section on page 1-4](#) and the [“Configuring Mappings” section on page 1-26](#).

Step 4: Configure Events/Customized Events

Customizing the Single Sign-On (SSO) and authentication operations for any event other than Login is considered an illegal action. There is no other time when these operations are necessary. Once a user is signed into and authenticated in the application from the external LDAP server, the process does not need to be replicated.

All events requiring connection to external datasources are configured here. When invoking the Custom Code APIs described in this guide, it is important to think through the sequence of operations for each event so that the custom operation does not occur out of order and fail.

-
- Step 1** From the Navigation Pane, click **Events**.
- Step 2** For the event you wish to customize, click **Edit**.
- Step 3** If the event is disabled, use the drop-down menu to choose **Enabled**.
- Step 4** Click **Add step** to add an operation. You can add as many steps as necessary now, or complete the details of each step before adding and configuring the next.
- Step 5** Choose the **Operation** from the drop-down menu.
- To simply invoke the code for SSO; for example, you can choose SSO from the menu. To *customize* the code for SSO, choose Custom Code, and then, in the next step, choose which operation you want to customize.
 - To configure a customized operation, choose **Custom Code**.
- Step 6** Choose your **mapping** and **datasource** from the drop-down menus.
- Step 7** Under the “Additional Options” heading, click **Options**.
- Step 8** Configure the options for that step:
- For Custom Code Operation Type, use the drop-down menu to choose the operation you wish to customize.
 - For Java Class, enter the entire package name for that operation, followed by the class name; for example, `com.newscale.bfw.eui.api.samples.operations.CustomCodeTester`.
 - In the above example, the Java class name is in *italics*. Both of these may be found in and copied from the code itself.
- Step 9** Click **Close** to close the additional options for the step.
- Step 10** Continue adding and configuring steps, as necessary.
- Step 11** Click **Update** to save all steps for that event.
-

Using Custom Code as an Operation Type

In the steps above, if you choose Custom Code as the operation and Custom Code again for the operation type, you are then calling an undefined Custom Code, which you must design.

In the Custom Code test example provided by Cisco, you can use the Java Class “performCustom” to define your own custom code.

Deploying Custom Code

All custom code must be packaged as a customization to the Service Portal installer. This allows the customizations to be reapplied if the installation needs to be upgraded or to install a new site.

Instructions for packaging and deploying custom code are dependent on the application server which hosts Service Portal. See the *Cisco Service Portal Installation Guide* and *Cisco Service Portal Configuration Guide* for further information.

Sample View/Usage of the API

The solution here satisfies these use cases:

- Create an event class that searches for a person using data collected from a container-managed SQL datasource.
- Create an event class that imports a person using data collected from a container-managed SQL datasource.
- Create an event class that modifies a person using data collected from a container-managed SQL datasource.
- Create an event class that can receive configuration parameters from the UI. The mappings interface is used in this example to pass the configuration parameters to the class.

It also creates the home OU for the person as a business unit, if it doesn't already exist in Service Portal.

Note that the solution requires a datasource to be configured on the application server. The following sections illustrate configuration and usage of the `EUIPersonSearchSQL` class.

SQL Datasource

Any SQL table or tables that contain data for the mandatory fields in a Person profile (or from which values for those fields can be derived) could be used as a datasource. Here is the table definition used in this example:

```
CREATE TABLE [psgextusers] (
  [login]      [nvarchar] (100) COLLATE Latin1_General_CI_AI NOT NULL,
  [firstname] [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [lastname]  [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [password]  [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [email]     [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  [homeOU]   [nvarchar] (100) COLLATE Latin1_General_CI_AI NULL,
  CONSTRAINT [PK_extuser] PRIMARY KEY CLUSTERED
  (
    [login]
  ) ON [PRIMARY]
) ON [PRIMARY]
```

GO

The following is some sample data to go with the above table definition:

```
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Moe', 'Moe', 'Howard', 'Moe', 'moe@stooge.com',
'Nyuk Nyuk Nyuk')
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Larry', 'Larry', 'Fine', 'Larry',
'larry@stooge.com', 'Nyuk Nyuk Nyuk')
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Curly', 'Curly', 'Howard', 'Curly',
'curly@stooge.com', 'Nyuk Nyuk Nyuk')
INSERT INTO [RequestCenter].[dbo].[psgextusers]([login], [firstname], [lastname],
[password], [email], [homeOU])VALUES('Shemp', 'Shemp', 'Howard', 'Shemp',
'shemp@stooge.com', 'Nyuk Nyuk Nyuk')
```

Datasource Definition

To use the Directory Integration interface you have to have an LDAP datasource configured. LDAP is the only UI supported datasource. You can create maps without a datasource, but you cannot test them without an LDAP datasource.

Figure 1-16 Sample Datasource Configuration

The screenshot displays the 'Datasources' management interface. At the top, a table lists existing datasources. Below the table are buttons for 'Add', 'Remove', 'Test Connection', and 'Copy'. The 'Add or Edit a Datasource' section is active, showing configuration fields for a 'DummySQL' datasource. The 'Authentication Method' is set to 'Simple', 'Mechanism' to 'Non SSL', and 'BindDN' to 'Bogus'. Other fields include 'Port Number' (99999), 'User BaseDN' (Bogus), and 'Host' (Bogus). The 'Optional Filter' field is empty. At the bottom, there are 'Update' and 'Cancel' buttons.

Datasource Name	Protocol	Action	Test Status
DummySQL	LDAP	Edit	

Datasource Configuration

Add or Edit a Datasource

* Datasource Name: DummySQL

Datasource Description: This is to satisfy the rule that requires a datasource in the event UI. THIS IS A BUG.

Select protocol and server product

Connection Information

* Authentication Method: Simple

* Mechanism: Non SSL

* BindDN: Bogus

* Host: Bogus

* Port Number: 99999

* Password: [Redacted]

* User BaseDN: Bogus

Optional Filter: [Empty]

Security Certificate Information

Referral Datasource

Update Cancel

Configuring a container-managed datasource depends on the container. Detailed instructions on configuring datasources are given in the *Cisco Service Portal Installation Guide*.

Sample Mapping

A mapping must be created for the `EUIPersonSearchSql` class.

Figure 1-17 Sample Mapping Configuration

Mapping Configuration

Add or edit a mapping name

* Mapping Name	<input type="text" value="EUIPersonSearchSql"/>
Mapping Description	<div style="border: 1px solid #ccc; padding: 5px; min-height: 40px;"> Mapping for the EUIPersonSearchSql class. This maps ResultSet columns to a Person profile. In addition, Custom 9 collects the JNDI datasource name and Custom 10 collects the table name. </div>

Configure mapping attributes

Choose a Datasource for testing None

Person Data	Mapped Attributes	Test Values
* First Name	<input type="text" value="firstName"/>	<input type="text"/>
* Last Name	<input type="text" value="lastName"/>	<input type="text"/>
* Login ID	<input type="text" value="login"/>	<input type="text"/>
* Person Identification	<input type="text" value="login"/>	<input type="text"/>
* Email Address	<input type="text" value="email"/>	<input type="text"/>
* Home Organizational Unit	<input type="text" value="homeOU"/> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block; margin-left: 5px;">Custom 9</div>	<input type="text" value="java:/PSGSQLDS"/>
* Password	<input type="text" value="password"/> <div style="border: 1px solid #ccc; padding: 2px; display: inline-block; margin-left: 5px;">Custom 10</div>	<input type="text" value="psgextusers"/>
[-] Optional Person Data Mappings		
Person Data	Mapped Attributes	Test Values
Title	<input type="text"/>	<input type="text"/>
Social Security Number	<input type="text"/>	<input type="text"/>
Birthdate	<input type="text"/>	<input type="text"/>
Hire Date	<input type="text"/>	<input type="text"/>
Custom 9	<input type="text" value="java:/PSGSQLDS"/>	<input type="text"/>
Custom 10	<input type="text" value="psgextusers"/>	<input type="text"/>

This mapping includes references to the JNDI as Custom 9 and the table name for Custom 10. Using a mapping like this, it is possible to do a simple query such as “select * from tablename” and use the metadata functionality in JDBC to select the column based on the mapping.

Sample Event Configuration

The “Person Lookup for Order on Behalf” event has two steps: The first must perform a “Person Search” operation. The name of the class is given as the mapping. The complete package specification is given as the Java class.

Figure 1-18 Custom Person Search Operation

Events				
Name	Status	Action		
Login	Disabled	Edit		
Person Lookup for Order on Behalf	Enabled	Edit		
Person Lookup for Service Form	Disabled	Edit		
Person Lookup for Authorization Delegate	Disabled	Edit		

Event Configuration				
Event Name	Person Lookup for Order on Behalf			
Event Status	Enabled			
<input type="checkbox"/> Event Step	Operation	Mapping	Datasource	Additional Options
<input type="checkbox"/> Step 1	Custom Code	EUIPersonSearchSql	DummySQL	Options
<input type="checkbox"/> Step 2	Custom Code	EUIPersonSearchSql	DummySQL	Options
<input type="button" value="Add step"/> <input type="button" value="Remove step"/>				
Options for Step1				
Custom Code Operation Type	Person Search			
Java Class	com.newscale.profsvcs.eui.EUIPersonSearchSql			
<input type="button" value="Close"/>				
<input type="button" value="Update"/> <input type="button" value="Cancel"/>				

The second step in the “Person Lookup for Order on Behalf” event is to import the selected person (“Import Person”). This configuration uses the same Java class, but a different Custom Code Operation Type. The Custom Code Operation Types in the drop-down menu correspond to the methods that are called in the interface class.

Figure 1-19 Event Step 2 – Custom Import Person Operation

Events		
Name	Status	Action
Login	Disabled	<input type="button" value="Edit"/>
Person Lookup for Order on Behalf	Enabled	<input type="button" value="Edit"/>
Person Lookup for Service Form	Disabled	<input type="button" value="Edit"/>
Person Lookup for Authorization Delegate	Disabled	<input type="button" value="Edit"/>

Event Configuration	
Event Name	Person Lookup for Order on Behalf
Event Status	Enabled <input type="button" value="v"/>

<input type="checkbox"/> Event Step	Operation	Mapping	Datasource	Additional Options
<input type="checkbox"/> Step 1	Custom Code <input type="button" value="v"/>	EUIPersonSearchSql <input type="button" value="v"/>	DummySQL <input type="button" value="v"/>	<input type="button" value="Options"/>
<input type="checkbox"/> Step 2	Custom Code <input type="button" value="v"/>	EUIPersonSearchSql <input type="button" value="v"/>	DummySQL <input type="button" value="v"/>	<input type="button" value="Options"/>

Options for Step2

Custom Code Operation Type Import Person

Java Class

Sample Code for SQL-Based Person Lookup

The following is the source for the custom class:

```
package com.newscale.profsvcs.eui;

import com.newscale.api.person.*;
import com.newscale.bfw.eui.EUIException;
import com.newscale.bfw.eui.api.*;
import com.newscale.bfw.ldap.ILDAPApi;
import com.newscale.bfw.logging.ILogUtil;
import com.newscale.bfw.logging.LogUtilFactory;
import com.newscale.comps.extuserintegration.session.*;

import javax.servlet.http.HttpServletRequest;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.*;

/**
 * Person Search to an external SQL datasource
```

```

*
* @author Lee Weisz
* @version $Revision$
*/
public class EUIPersonSearchSql implements IPersonSearch {
    /**
     * Logger instance
     */
    private ILogUtil log = LogUtilFactory.getLogUtil(EUIPersonSearchSql.class);

    /**
     * Implement Person Search Operation and fetch users from an external system
     *
     * @param euiOperationDTO
     * @param euiPersonSearchOperationContext
     *
     * @param request
     * @param signOnImportPersonAPI
     * @param ldapApi
     * @return
     * @throws EUIException
     */
    public IEUIPersonSearchOperationResult search(IEUIEventPersonSearchOperationDTO
euiOperationDTO,
                                                    IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                    HttpServletRequest request,
                                                    ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
        throws EUIException {

        log.debug("search: Entering search method...");
        IEUIPersonSearchOperationResult euiOperationResult = euiPersonSearchOperationContext
            .getEUIPersonSearchOperationResult();

        // Check if there is any SearchPerson List already available, if so we
        // can append to the existing List

        // Typically if there is a productized Person Search Operation is
        // configured before the custom code, this list would be populated

        // TODO Why is this an ArrayList? Can't it be a List?
        ArrayList personList = euiOperationResult.getSearchPersonList();

        if (null == personList) {
            personList = new ArrayList();
        }

        // Get the search criteria from the dialog box
        String searchFirstName = euiPersonSearchOperationContext.getFirstNameSearchString();
        String searchLastName = euiPersonSearchOperationContext.getLastNameSearchString();

        log.debug("search: Looking for " + searchFirstName + " " + searchLastName);

        EUIDataMappingDTO dataMappingDTO = euiOperationDTO.getEuiMappingDTO();
        Map attributeMap = dataMappingDTO.getAllAttributeMap();

        // What's in this map?
        if (log.isDebugEnabled()) {
            Set ks = attributeMap.keySet();
            for (Iterator it = ks.iterator(); it.hasNext();) {
                Object key = it.next();
                log.debug("search: " + key + " is " + attributeMap.get(key));
            }
        }
    }
}

```

```

    }

    // Use the map to map the columns to Person fields
    String firstNameColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_FIRSTNAME);
    String lastNameColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LASTNAME);
    String loginColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LOGINID);

    // Use the custom9 mapping to hold the datasource value and custom10 to
    // hold the tablename. Since we control the import as well, it won't show up
    // in the imported Person's profile
    String ds = (String) attributeMap.get("custom9");
    String sourceTable = (String) attributeMap.get("custom10");

    StringBuffer searchSQL;
    searchSQL = new StringBuffer().append("select ")
        .append(firstNameColumn).append(", ")
        .append(lastNameColumn).append(", ")
        .append(loginColumn).append(" from ")
        .append(sourceTable);

    if (searchFirstName != null && searchFirstName.trim().length() > 0 ||
        searchLastName != null && searchLastName.trim().length() > 0) {
        searchSQL.append(" where ");

        if (searchFirstName != null && searchFirstName.trim().length() > 0) {
            searchSQL.append(firstNameColumn).append(" like
'").append(searchFirstName.trim()).append("%'");
        }

        if (searchFirstName != null && searchFirstName.trim().length() > 0 &&
            searchLastName != null && searchLastName.trim().length() > 0) {
            searchSQL.append(" and ");
        }

        if (searchLastName != null && searchLastName.trim().length() > 0) {
            searchSQL.append(lastNameColumn).append(" like
'").append(searchLastName.trim()).append("%'");
        }
    }

    log.debug("search: " + searchSQL.toString());

    Connection conn = null;
    Statement s = null;

    // get a connection to the external db
    try {
        conn = signOnImportPersonAPI.getExternalDBConnection(ds);

        s = conn.createStatement();
        ResultSet rs = s.executeQuery(searchSQL.toString());

        while (rs.next()) {
            String fname = rs.getString(firstNameColumn);
            String lname = rs.getString(lastNameColumn);
            String login = rs.getString(loginColumn);

            IExtUserDTO extUserDTO = PersonFactory.createExtUserDTO();
            IPersonDTO personDTO = PersonFactory.createPersonDTO();

```

```

        personDTO.setFirstName(fname);
        personDTO.setLastName(lname);
        personDTO.setPersonIdentification(login);

        // Make the IPersonDTO into an IExtPersonDTO
        extUserDTO.setPersonDTO(personDTO);
        // Add IExtUserDTO to the collection of searched persons
        personList.add(extUserDTO);
    }
} catch (SQLException e) {
    log.error("search: " + searchSQL.toString(), e);
} catch (SignInImportPersonAPIException e) {
    log.error("search: Cannot get a connection to " + ds, e);
} finally {
    try {
        s.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
    try {
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// Set the list of Persons Searched into the Result to be returned
euiOperationResult.setSearchPersonList(personList);

log.debug("search: Leaving search method...");
return euiOperationResult;
}

/**
 * Implement the Import Person Operation to Import a user from External
 * system
 *
 * @param euiOperationDTO
 * @param euiPersonSearchOperationContext
 *
 * @param request
 * @param signInImportPersonAPI
 * @param ldapApi
 * @return
 * @throws EUIException
 */
public IEUIPersonSearchOperationResult importPerson(IEUIEventImportPersonOperationDTO
euiOperationDTO,
                                                    IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                    HttpServletRequest request,
                                                    ISignOnImportPersonAPI
signInImportPersonAPI, ILDAPApi ldapApi)
    throws EUIException {

    log.debug("importPerson: Entering importPerson method...");

    /* Potentially useful stuff on the request...
    Name : isOOB           Value : true/false
    Name : customerid     Value : personDTO.setPersonIdentification() from search
    Name : customerId     Value : personDTO.setPersonIdentification() from search
    Name : LDAPCustomerId Value : personDTO.setPersonIdentification() from search
    */

```



```

// What's on this request?
if (log.isDebugEnabled()) {
    log.debug("importPerson: Parameters collected from the search window...");
    Enumeration paramNames = request.getParameterNames();

    if (paramNames.hasMoreElements()) {
        while (paramNames.hasMoreElements()) {
            String paramName = (String) paramNames.nextElement();
            String paramValues[] = request.getParameterValues(paramName);
            if (paramValues != null) {
                log.debug("importPerson: Name : " + paramName);
                for (int i = 0; i < paramValues.length; i++) {
                    log.debug("importPerson: Value : " + paramValues[i]);
                }
            }
        }
    }
}

boolean refreshPerson = true;
String login = request.getParameter("customerId");

// Defaults
String homeOU    = "";
String firstName = "";
String lastName  = "";
String email     = "";
String password  = "password";

// Get the UI mapping
EUIDataMappingDTO dataMappingDTO = euiOperationDTO.getEuiMappingDTO();
Map attributeMap = dataMappingDTO.getAllAttributeMap();

// Use the map to map the columns to Person fields
String firstNameColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_FIRSTNAME);
String lastNameColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LASTNAME);
String loginColumn     = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_LOGINID);
String passwordColumn = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_PASSWORD);
String emailColumn     = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_EMAILADDRESS);
String homeOUColumn    = (String)
attributeMap.get(EUIAPIConstants.EUIMAPFIELDTYPE.ATTR_HOMEORGANIZATIONLUNIT);

// Use the custom9 mapping to hold the datasource value and custom10 to
// hold the tablename. Since we control the import as well, it won't show up
// in the imported Person's profile unless we screw up somehow and put it there...
String ds = (String) attributeMap.get("custom9");
String sourceTable = (String) attributeMap.get("custom10");

StringBuffer importSQL;
importSQL = new StringBuffer().append("select ")
    .append(firstNameColumn).append(", ")
    .append(lastNameColumn).append(", ")
    .append(loginColumn).append(", ")
    .append(passwordColumn).append(", ")
    .append(emailColumn).append(", ")
    .append(homeOUColumn)
    .append(" from ").append(sourceTable).append(" where ")
    .append(loginColumn).append("=").append(login).append("'");

```

```

log.debug("import: " + importSQL.toString());

Connection conn = null;
Statement s = null;

try {
    // get a connection to the external db
    conn = signOnImportPersonAPI.getExternalDBConnection(ds);
    s = conn.createStatement();
    ResultSet rs = s.executeQuery(importSQL.toString());

    while (rs.next()) {
        homeOU = rs.getString(homeOUColumn);
        firstName = rs.getString(firstNameColumn);
        lastName = rs.getString(lastNameColumn);
        email = rs.getString(emailColumn);
        password = rs.getString(passwordColumn);
    }
} catch (SQLException e) {
    log.error("import: " + importSQL.toString(), e);
} catch (SignOnImportPersonAPIException e) {
    log.error("import: Cannot get a connection to " + ds, e);
} finally {
    try {
        s.close();
    } catch (SQLException e) {
        log.error("import: ", e);
    }
    try {
        conn.close();
    } catch (SQLException e) {
        log.error("import: ", e);
    }
}

log.debug("import : Got " + login + ", " + firstName + ", " + lastName + ", " + email +
", " + password + ", " + homeOU);

IPersonDTO personDTO = PersonFactory.createPersonDTO();
try {
    // Get or Create the Person
    // This API throws an exception if the Person is not found in Request Center
    try {
        personDTO = signOnImportPersonAPI.getPersonByLoginName(login);
        log.info("importPerson: " + login + " exists in Request Center");
    } catch (SignOnImportPersonAPIException impEx) {
        log.info("importPerson: Creating new Person for " + login);
        refreshPerson = false;
        personDTO.setLogin(login);
    }

    // Get or Create the Home OU that the Person should be associated with
    // This API throws an exception if the OU is not found in Request Center
    IOrganizationalUnitDTO homeOUDTO;
    try {
        homeOUDTO = signOnImportPersonAPI.getOrgUnitByName(homeOU);
        log.info("importPerson: " + homeOU + " exists in Request Center");
    } catch (SignOnImportPersonAPIException impEx) {
        log.info("importPerson: Creating new OU " + homeOU + " for " + login);
        homeOUDTO = PersonFactory.createOrganizationalUnitDTO();
        homeOUDTO.setName(homeOU);
        homeOUDTO.setBillable(false);
        homeOUDTO.setOrganizationalUnitTypeId(2); // business unit.
        homeOUDTO.setRecordStateId(1); // active
    }
}

```

```

        homeOUDTO.setLocaleId(EUIAPIConstants.LOCALEID.USEN);
        try {
            homeOUDTO = signOnImportPersonAPI.createOrgUnit(homeOUDTO);
        } catch (SignOnImportPersonAPIException crEx) {
            log.error("importPerson: Can't create " + homeOU + " for " + login);
            throw crEx;
        }
    }
    personDTO.setHomeOrganizationalUnitId(homeOUDTO.getId());

    // Populate the Login Object...
    // Modify the login information only if this is a new Person
    if (!refreshPerson) {
        ILoginInfoDTO loginInfoDTO = PersonFactory.createLoginInfoDTO();

        loginInfoDTO.setLoginname(personDTO.getLogin());
        loginInfoDTO.setPrivateKey(personDTO.getLogin());
        // Set the un-encrypted password
        loginInfoDTO.setPassword(password);

        // Set ILoginInfoDTO to IPersonDTO
        personDTO.setILoginInfoDTO(loginInfoDTO);
    }

    // Populate the rest of the essential fields
    // Presumably, any expression on the mapping will have already been executed
    // and the result is what's returned in the personDTO
    personDTO.setFirstName(firstName);
    personDTO.setLastName(lastName);
    personDTO.setEmail(email);

    // Set the active status
    // TODO These methods are bogus...
    // personDTO.setIsInactive(false);
    // personDTO.setIsActive(true);
    // TODO What do these numbers mean? Is there a constants library to convert these
codes into something meaningful?
    personDTO.setRecordStateId(1);

    // Upsert the Person
    signOnImportPersonAPI.beginTransaction();
    if (refreshPerson) {
        // Update the existing Person
        // This method updates only Basic Info, LoginInfo, Preferences, Home OU and Person
Extension
        signOnImportPersonAPI.updatePerson(personDTO);
    } else {
        // Create the Person
        // This creates a Person with Basic Info, LoginInfo, Preferences, Home OU and
Person Extension
        personDTO = signOnImportPersonAPI.createPerson(personDTO);
        // From here on out it's a refresh
        refreshPerson = true;
    }
    signOnImportPersonAPI.commitTransaction();
} catch (Exception e) {
    log.error("importPerson: Exception during Import Person", e);
    try {
        // Rollback Transaction
        signOnImportPersonAPI.rollbackTransaction();
    } catch (SignOnImportPersonAPIException se) {
        log.error("importPerson: Error while Rolling back transaction", se);
    }
} finally {

```

```

        // Release Transaction
        signOnImportPersonAPI.releaseTransaction();
    }

    IExtUserDTO extUserDTO = PersonFactory.createExtUserDTO();
    extUserDTO.setPersonDTO(personDTO);

    IEUIPersonSearchOperationResult psor =
    euiPersonSearchOperationContext.getEUIPersonSearchOperationResult();
    psor.setImportedPersonExtDTO(extUserDTO);

    log.debug("importPerson: Leaving importPerson method...");

    return psor;
}

/**
 * Implement Import Manager Operation and Import all the Supervisors chain
 * of the Person being imported
 *
 * @param euiOperationDTO
 * @param euiPersonSearchOperationContext
 *
 * @param request
 * @param signOnImportPersonAPI
 * @param ldapApi
 * @return
 * @throws EUIException
 */
public IEUIPersonSearchOperationResult importManager(IEUIEventImportManagerOperationDTO
euiOperationDTO,
                                                    IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                    HttpServletRequest request,
                                                    ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
    throws EUIException {
    return null;
}

/**
 * Implement any Custom Operation
 *
 * @param euiOperationDTO
 * @param euiPersonSearchOperationContext
 *
 * @param request
 * @param signOnImportPersonAPI
 * @param ldapApi
 * @return
 * @throws EUIException
 */
public IEUIPersonSearchOperationResult performCustom(IEUIEventCustomOperationDTO
euiOperationDTO,
                                                    IEUIPersonSearchOperationContext
euiPersonSearchOperationContext,
                                                    HttpServletRequest request,
                                                    ISignOnImportPersonAPI
signOnImportPersonAPI, ILDAPApi ldapApi)
    throws EUIException {
    return null;
}
}

```

Supported Time Zones

The supported time zone values when mapping time zones are listed below.

Time Zone Name	GMT Equivalent
Etc/GMT+12	(GMT-12:00) International Date Line West
Pacific/Apia	(GMT-11:00) Samoa
US/Hawaii	(GMT-10:00) Hawaii
US/Aleutian	(GMT-10:00) Hawaii Aleutian Daylight Time
US/Alaska	(GMT-09:00) Alaska
America/Tijuana	(GMT-08:00) Pacific Time (US and Canada); Tijuana
America/Chihuahua	(GMT-07:00) Chihuahua, La Paz, Mazatlan
US/Arizona	(GMT-07:00) Arizona
Canada/Mountain	(GMT-07:00) Mountain Time (US and Canada)
Canada/Saskatchewan	(GMT-06:00) Saskatchewan
US/Central	(GMT-06:00) Central America
Canada/Central	(GMT-06:00) Central Time (US and Canada)
America/Mexico_City	(GMT-06:00) Guadalajara, Mexico City, Monterrey
America/Bogota	(GMT-05:00) Bogota, Lima, Quito
Canada/Eastern	(GMT-05:00) Eastern Daylight Time (US and Canada)
America/Jamaica	(GMT-05:00) Eastern Time (US and Canada)
US/East-Indiana	(GMT-05:00) Indiana (East)
America/Antigua	(GMT-04:00) Atlantic Time (Canada)
Canada/Atlantic	(GMT-04:00) Atlantic Daylight Time (Canada)
America/Manaus	(GMT-04:00) Manaus
America/Santiago	(GMT-04:00) Santiago
America/Caracas	(GMT-04:30) Caracas
America/La_Paz	(GMT-04:00) La Paz (Bolivia)
America/Sao_Paulo	(GMT-03:00) Brasilia
America/Godthab	(GMT-03:00) Greenland
America/Argentina/Buenos_Aires	(GMT-03:00) Buenos Aires
America/Guyana	(GMT-04:00) Georgetown
America/St_Johns	(GMT-03:30) Newfoundland and Labrador
Atlantic/South_Georgia	(GMT-02:00) Mid-Atlantic
Atlantic/Azores	(GMT-01:00) Azores
Atlantic/Cape_Verde	(GMT-01:00) Cape Verde Islands
Etc/Greenwich	(GMT) Greenwich Mean Time: Dublin, Edinburgh,
Africa/Casablanca	(GMT) Casablanca, Monrovia
Europe/Sarajevo	(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb
Europe/Brussels	(GMT+01:00) Brussels, Copenhagen, Madrid, Paris
Africa/Brazzaville	(GMT+01:00) West Central Africa
Europe/Amsterdam	(GMT+01:00) Amsterdam, Berlin, Bern, Rome,
Europe/Belgrade	(GMT+01:00) Belgrade, Bratislava, Budapest,

Time Zone Name	GMT Equivalent
Africa/Cairo	(GMT+02:00) Cairo
Europe/Helsinki	(GMT+02:00) Helsinki, Kiev, Riga, Sofia, Tallinn,
Europe/Minsk	(GMT+02:00) Minsk
Europe/Athens	(GMT+02:00) Athens, Bucharest, Istanbul
Asia/Jerusalem	(GMT+02:00) Jerusalem
Africa/Windhoek	(GMT+02:00) Windhoek
Africa/Harare	(GMT+02:00) Harare, Pretoria
Asia/Baghdad	(GMT+03:00) Baghdad
Africa/Nairobi	(GMT+03:00) Nairobi
Europe/Moscow	(GMT+03:00) Moscow, St. Petersburg, Volgograd
Asia/Kuwait	(GMT+03:00) Kuwait, Riyadh
Asia/Tehran	(GMT+03:30) Tehran
Asia/Baku	(GMT+04:00) Baku
Asia/Muscat	(GMT+04:00) Abu Dhabi, Muscat
Asia/Yerevan	(GMT+04:00) Yerevan
Asia/Tbilisi	(GMT+04:00) Tbilisi
Asia/Kabul	(GMT+04:30) Kabul
Asia/Karachi	(GMT+05:00) Islamabad, Karachi, Tashkent
Asia/Yekaterinburg	(GMT+05:00) Ekaterinburg
Asia/Kolkata	(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi
Asia/Kathmandu	(GMT+05:45) Kathmandu
Asia/Dhaka	(GMT+06:00) Astana, Dhaka
Asia/Novosibirsk	(GMT+07:00) Novosibirsk
Asia/Colombo	(GMT+05:30) Sri Jayawardenepura
Asia/Rangoon	(GMT+06:30) Yangon (Rangoon)
Asia/Bangkok	(GMT+07:00) Bangkok, Hanoi, Jakarta
Asia/Krasnoyarsk	(GMT+08:00) Krasnoyarsk
Asia/Irkutsk	(GMT+09:00) Irkutsk
Asia/Kuala_Lumpur	(GMT+08:00) Kuala Lumpur, Singapore
Asia/Taipei	(GMT+08:00) Taipei
Australia/Perth	(GMT+08:00) Perth
Asia/Chongqing	(GMT+08:00) Beijing, Chongqing, Hong Kong SAR,
Asia/Seoul	(GMT+09:00) Seoul
Asia/Tokyo	(GMT+09:00) Osaka, Sapporo, Tokyo
Asia/Yakutsk	(GMT+09:00) Yakutsk
Australia/Darwin	(GMT+09:30) Darwin
Australia/Adelaide	(GMT+09:30) Adelaide
Australia/Hobart	(GMT+10:00) Hobart
Australia/Canberra	(GMT+10:00) Canberra, Melbourne, Sydney
Australia/Brisbane	(GMT+10:00) Brisbane
Asia/Vladivostok	(GMT+10:00) Vladivostok

Time Zone Name	GMT Equivalent
Pacific/Guam	(GMT+10:00) Guam, Port Moresby
Pacific/Guadalcanal	(GMT+11:00) Solomon Islands, New Caledonia
Pacific/Auckland	(GMT+12:00) Auckland, Wellington
Pacific/Fiji	(GMT+12:00) Fiji Islands
Pacific/Tongatapu	(GMT+13:00) Nuku alofa

Sample build.xml File

```
<?xml version="1.0" ?>
<project name="Sample Project" default="all" basedir=".">
- <!-- Main target -->
  <target name="all" depends="init,build,deploy" />
  <!-- Set the following properties to point to appropriate folders -->
  <property name="rcwar.dir" value="<apps server path where Request Center application WAR
file is deployed>" />
  <property name="javax.servlet.dir" value="<path where
jboss-servlet-api_3.0_spec-1.0.0.Final.jar is available in the app server>" />
  <property name="rcwar_webinf_classes.dir" value="${rcwar.dir}/WEB-INF/classes" />
  <target name="init">
    <property name="dirs.base" value="${basedir}" />
    <mkdir dir="${dirs.base}/out" />
    <property name="src" value="${dirs.base}/src" />
    <property name="out" value="${dirs.base}/out" />
  </target>
  <path id="classpath">
    <fileset dir="${rcwar.dir}" includes="*.jar" />
    <fileset dir="${javax.servlet.dir}"
includes="jboss-servlet-api_3.0_spec-1.0.0.Final.jar" />
    <pathelement path="${rcwar_webinf_classes.dir}" />
  </path>
- <!-- Compile Java Files -->
  <target name="build" depends="init">
    <javac srcdir="${src}" destdir="${out}" debug="true" includes="**/*.java"
classpathref="classpath" deprecation="true" fork="true" memoryinitialsize="256M"
memorymaximumsize="512M" />
  </target>
  <target name="deploy" depends="init">
    <copy todir="${rcwar_webinf_classes.dir}">
      <fileset dir="${out}">
        <include name="**/*.class" />
      </fileset>
    </copy>
  </target>
</project>
```




CHAPTER 2

Service Link

- [Overview, page 2-1](#)
- [Service Link Design and Development, page 2-4](#)
- [Monitoring Service Link Transactions, page 2-34](#)
- [Service Link Adapters, page 2-42](#)
- [Integration Wizard, page 2-59](#)
- [Service Link Troubleshooting and Administration, page 2-66](#)
- [Prebuilt Functions, page 2-69](#)

Overview

Introduction

Service Link is the Service Portal module that provides integration with external systems. It supplies a framework for configuring interfaces that allow delivery tasks, authorizations or reviews defined within a Service Portal workflow to be performed by other systems and a user interface for monitoring the operation of these interfaces.

The most common scenario for the use of Service Link is where data associated with a delivery plan task needs to be passed outside of Service Portal in order to ensure that the service is delivered satisfactorily. For example, a message might be passed to a hardware vendor for a procurement action or to an inventory or asset management system for a data record update. The external application may then send one or more messages back to Service Portal. Each message, in turn, could update Service Portal with the current status of the task within the external system, eventually indicating that the task has been completed and that the Service Portal workflow (delivery plan) can continue with subsequent tasks.

Service Link provides a number of built-in adapters to facilitate communication with external applications using different transport mechanisms including the interchange of files; database updates; web communication via http post requests or web services; and queue-based messaging. In addition to these default adapters, developers may use the Service Link Adapter Development Kit (ADK) to develop and deploy custom adapters. Additional adapter kits designed to interface to third-party products such as HP Open View Service Desk, BMC Remedy and IBM Identity Management are available from Cisco Advanced Services.

Service Link Prerequisites

Developing Service Link integrations requires a range of technical skills. These include:

- Understanding of service design, including how to configure dictionary usage in Active Form Components (AFCs) and how to design tasks in a delivery plan.
- A thorough knowledge of the target third-party system, including the servers hosting the application.
- For all adapters but the VMware adapter, a basic understanding of XML tag structure since Service Link operates by sending XML messages between Service Portal and the external system.
- For all adapters but the VMware adapter, an intermediate grasp of configuring XML Stylesheet Language (XSL) transformations, to supplement the XML transformations which can be applied by use of the Service Link wizards.
- If database adapters are to be used, SQL knowledge is also needed.
- If an http/web services adapter is to be used to pass messages between Service Portal and a web service, knowledge of web services components like SOAP, WSDL, and web service security is helpful.

Service Link Design Methodology and Components

Service Portal offers two approaches to designing integrations.

- The Integration wizard, available in Service Designer, provides a wizard-driven approach for creating web services integration.
- The Service Link module provides capabilities for creating and maintaining all integrations, regardless of the messaging protocols used to communicate with the external system.

Once an integration has been created, it may be viewed and maintained through the advanced configuration capabilities available through Service Link. Advanced users may create even web services integrations using this functionality, bypassing the wizards if desired.

Administrators also use Service Link to administer and troubleshoot integrations in a production environment.

An integration consists of the following components:

- Adapters

An adapter is a logical representation of a transport component by which Service Portal sends XML documents or other messages to third-party systems. Prepackaged adapters support different message transport protocols; including file, http/web service, JMS, IBM MQ, and database.

Adapters are composed of two components:

- An inbound adapter

Inbound adapters manage messages coming from an external system. The external system message may be altered into a “standard” nsXML (formerly known as newScale XML) format through the use of transformations so that the data can be interpreted by Request Center.

There are two types of inbound adapters: pollers and listeners. A poller is a thread that periodically wakes up and looks for incoming messages, while a listener waits and is awakened by an incoming external message. An example of a poller is the inbound file adapter, which needs to periodically check for messages. An example of a listener adapter is the Web Services Listener Adapter which waits until an HTTP response is received.

- An outbound adapter

Outbound adapters manage the XML messages coming out of Service Portal and send them to the configured external system. A “standard” nsXML outbound message comes to Service Link which may then alter the message through the use of transformations, so that it meets the expected format of messages directed to the external system. The outbound adapters then apply the correct protocol and logic to send the messages to the external system.

- Agents

An agent is a logical representation of a transport mechanism by which Service Portal communicates to/from a third-party system. Agents may be used by service designers to direct tasks to their proper third-party destination. In addition to tasks, authorizations and reviews can be externalized by specifying an agent to direct this action to an external system.

An agent is composed of an inbound and outbound adapter, optional message transformation (XSLT) components, optional parameters, and other settings to address error conditions.

- Transformations

XML stylesheet (XSL) transformations transform outgoing messages into a format understood by a third-party system, and transform incoming messages into a format understood by Service Portal.

An agent which includes an outbound adapter automatically creates an nsXML message, containing information relevant to the current requisition and task. A transformation associated with the agent may then transform that message into an external message, which are delivered to the external system via the outbound adapter configured for the agent. Similarly, an inbound agent receives an external message via the associated adapter. A transformation must then transform the message into an incoming message type that is recognized and processed by the Business Engine, the Workflow Manager for Service Portal.

- Dictionaries and Active Form Components

A dictionary is the service design component that holds fields of data required to fulfill a specific service request. Agent parameters mapped to dictionary fields (or other data available in the service request) provide a standard outbound message format easily understood by external systems. Agent parameters in an inbound message received from an external system instruct Service Link to update the value of the dictionary fields mapped to those parameters. The changed form data is immediately available in the service form. The active form component in which the dictionary is included must, in turn, be included in the service that implements the Service Link integration.

The Integration Wizard automatically creates an agent and transformation, as well as an integration dictionary and active form component to complete the agent configuration. Once these components have been created, they are maintained through Service Link and Service Designer.

Business Engine and nsXML

The key to understanding Service Link is to understand its interaction with the Business Engine. The Business Engine is the component that is responsible for all workflow. Workflow actions include:

- Starting tasks in the correct sequence in a delivery plan.
- Marking a task as complete when all requirements for completion have been met.
- Sending emails as configured when the triggering event occurs.

In a task plan that doesn't use Service Link (that is, where all tasks are internal to Service Portal), the operation of the Business Engine is largely invisible. The use of the Business Engine becomes apparent in Service Link, because Service Link must handle or generate messages that the Business Engine understands in order for the status of external tasks to be changed.

When the Business Engine starts an external task (that is, a task which is to be handled by Service Link), it generates an outbound nsXML message. The Service Link agent that is handling the outbound task is then responsible for transforming that nsXML message into a format that can be understood by target system and delivering that message to the target system via the outbound transport mechanism (adapter) specified in the agent.

Similarly, if Service Link is configured to receive an inbound message from an external system, it must transform that message into an inbound nsXML message that can be understood by the Business Engine. Inbound messages are available to update the service form data for the current request; to complete the current task; or to add user comments to the current request.

Valid nsXML messages are discussed in more detail in the following section.

Service Link Design and Development

Overview

Request Center offers two approaches for designing, developing and deploying Service Link integrations with third-party systems:

- **Integration Wizard:** If the integration is via a web service, you can use the Integration Wizard via Service Designer to create all Request Center integration components. The Web Service Definition Language file (wsdl) must be available to use this approach.
- **Service Link configuration:** If the integration uses any other transport mechanism or you wish to review or modify components originally created via the Integration Wizard, use the screens provided by Service Link and Service Designer to configure the integration components.

Service Link and Service Designer configuration uses the following methodology to design, develop and deploy integrations with third-party systems:

- Design the communication protocol to be used with the third-party target system. This includes inbound and outbound adapter selection, message format and content.
- If necessary create and deploy a custom Service Link adapter.
- Use Service Designer to design the service that implements the Service Link integration. The design components typically include one or more dictionaries, which contain data to be passed to the external system via agent parameters, as well as the active form components that include those dictionaries and configure the display properties for the fields in those dictionaries.
- Create agents, selecting appropriate outbound and inbound adapters, defining the properties of each, together with parameters passed in either direction. Service Link includes wizards and drop-down lists to partially automate the definition of agent parameters.
- Create transformations, if needed, for Service Portal to understand messages from third-party systems, and for third-party systems to understand messages from Service Portal.
- Use Service Designer to associate the agent with a task in the service's delivery plan. If applicable, ensure that agent parameters are properly mapped to dictionary fields used in that service.
- Test the configuration by requesting the service containing your task and then monitoring messages and external tasks via the corresponding Service Link pages.

This process is discussed in more detail in the following sections.

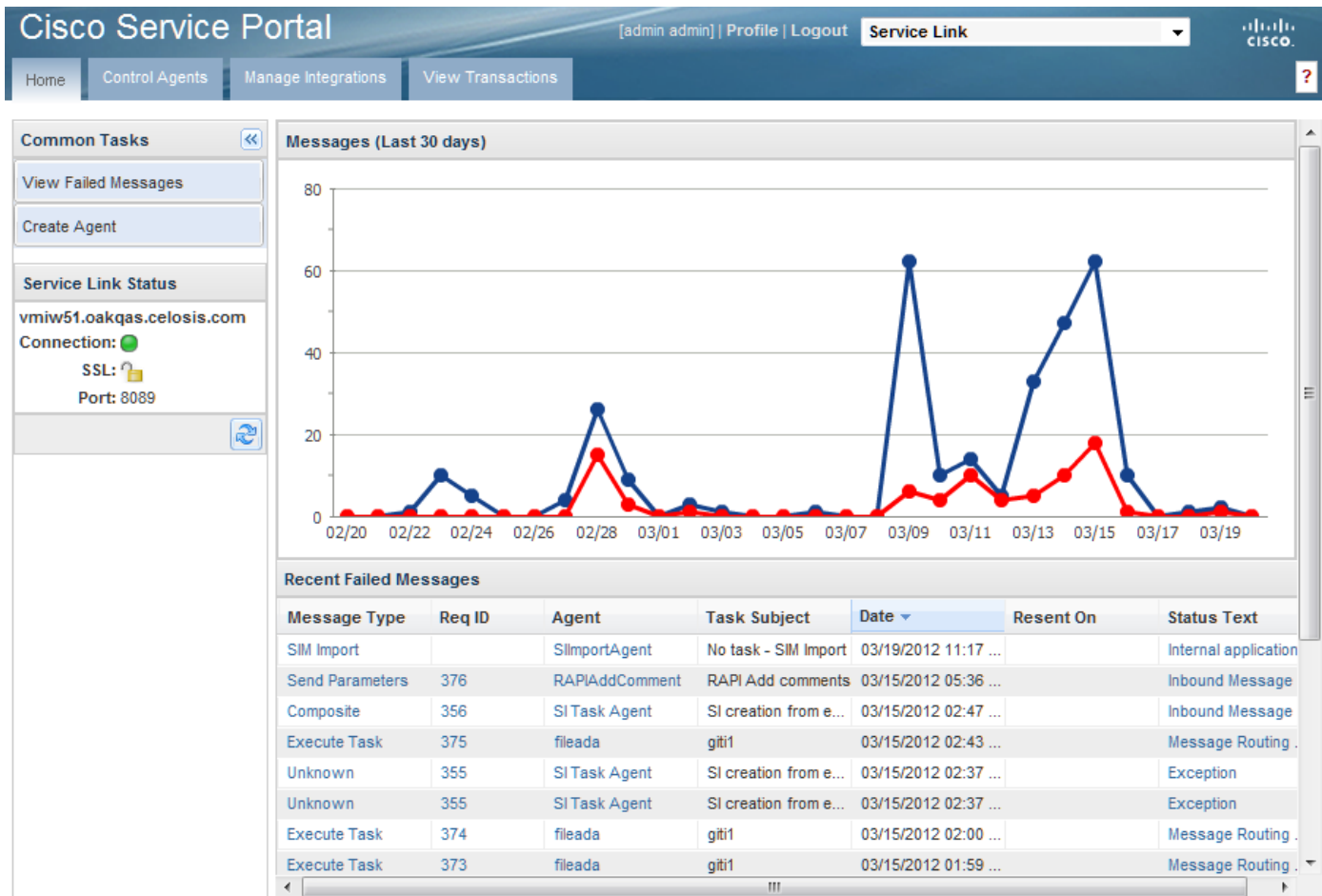
The Integration Wizard is described in the [“Integration Wizard” section on page 2-59](#).

Accessing Service Link

To access Service Link, choose **Service Link** from the Module Menu.

Service Portal System My Workspace
My Services My Services Executive
Relationship Manager Service Level Manager Service Manager
Organization Designer Portal Designer Portfolio Designer Service Designer Service Item Manager
Administration Catalog Deployer Service Link
Reporting Advanced Reporting

The Service Link home page appears, as shown below.



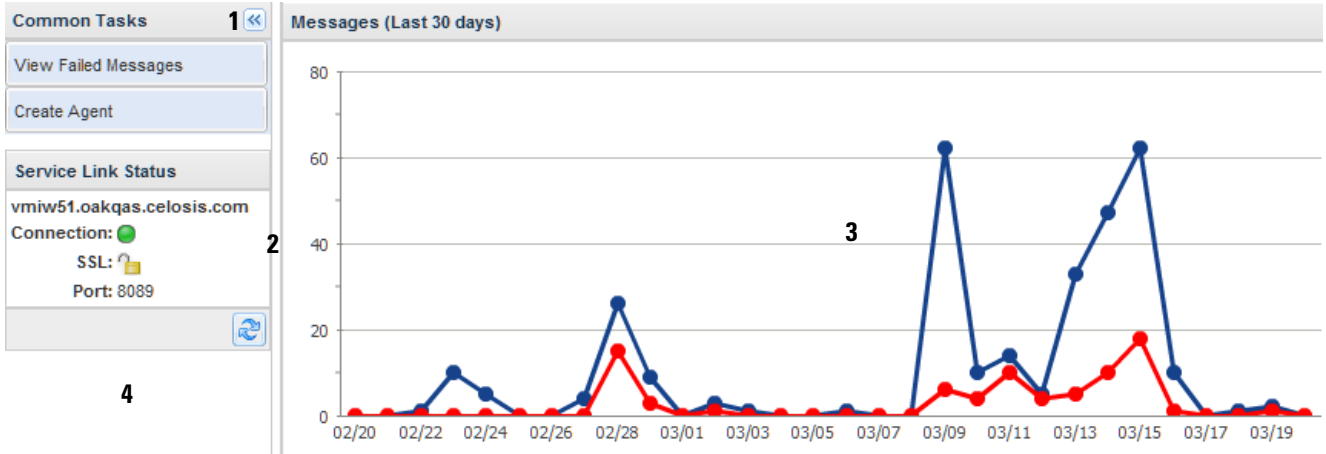
The Service Link home page contains the elements shown in [Table 2-1](#) below.

Table 2-1 Service Link Home Page Elements

Element	Description
Menu Bar	Located at the top of the page, contains the tabs used to administer a Service Link environment and to develop and maintain Service Link integrations.
Common Tasks	Located on the left side of the page, contains quick links to view a complete list of failed messages and to create an agent.
Service Link Status	Located on the left side of the page, shows the current status of Service Link.
Messages (Last 30 days)	Located in the right pane of the page, this graph summarizes message volume for the most recent 30-day period.
Recent Failed Messages	Located on the bottom right of the page, this grid lists the most recent Service Link messages that were not delivered successfully. Hyperlinks are provided to the message, requisition, and agent details.

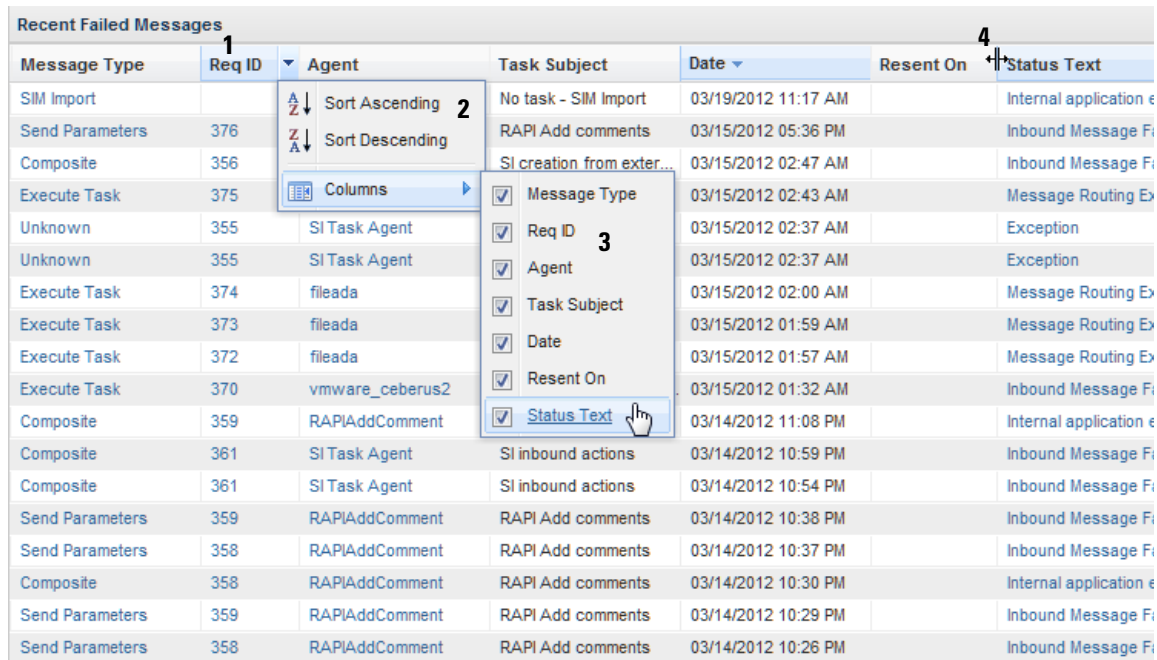
Managing the Service Link Screens

Like the home page, most Service Link pages consist of a list pane on the left and a content pane on the right. The list pane can be collapsed and expanded by clicking the collapse and expand pane icons. Even with the list pane displayed, the width of the content pane can be changed by dragging the divider.



1	Collapse pane icon	3	Content pane
2	Divider	4	List pane

You can also control the appearance of the Recent Failed Messages grid (and other grids displayed presented by Service Link):



1	Drop-down icon	3	Columns submenu
2	Sort options	4	Width cursor

- To change the width of a column, 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.
- To sort the display, move the mouse over the header of the column to sort by. A drop-down icon appears to the right of the header. Click the icon to show the sort options, then click the desired option.
- To change the columns shown in the grid, click the Columns submenu in the drop-down list, then check a column to display it, or uncheck the column to remove it from the grid.
- To change the order of the columns in the grid, click the mouse in the label of the column to be moved, drag the mouse and release it at the desired position for the column.

Designing the Communication Protocol

In addition to the configuration and testing work that are executed in Service Link, equivalent work must be undertaken by the technical resources responsible for the third-party system. A well considered design is essential if the interface is to operate robustly.

The interfacing capabilities of the system to be communicated to will normally dictate the basic design of the integration, that is, which adapters will be used.

File and database adapters are the simplest to configure. If JMS, MQ or http/web service adapters are deployed, some expertise from network management teams may be involved to ensure that connectivity issues do not prevent the data from moving from one system to the other. Data security concerns are likely to be a factor if the target system exists outside your company's network—for example, to use a SOAP message sent via http or https to communicate with an outside vendor. If a custom adapter is required, more technical resources will be required and more time to complete the integration should be budgeted for.

Normally the data required for the outbound Service Link communication would be assessed first. Consideration needs to be given to what fields are readily available (via the nsXML outbound message) and what additional data needs to be provided (via form data and agent parameters). While outbound communications only occur once per task (when the task starts), multiple separate inbound communications can be supported. On receiving its instruction to perform work, a third-party system may issue a single (inbound) communication on completion. Alternatively, multiple updates could be sent before the work is complete. Examples include where a reference ID may be communicated, textual status updates have to be sent back and finally the completion confirmation is communicated.

Adapters

Service Link adapters are preconfigured for use. Additional adapters may be developed using the Adapter Development Kit. You may review the available adapters using the Adapters page of the Manage Integrations tab in Service Link.

-
- Step 1** From the Service Link home page, click **Manage Integrations**. Then click the **Adapters** subtab. The Adapters page appears, as shown below.

The screenshot shows the Cisco Service Portal interface. At the top, there is a navigation bar with "Home", "Control Agents", "Manage Integrations", and "View Transactions". The user is logged in as "admin admin" and is viewing the "Service Link" page. Below the navigation bar, there are tabs for "Agents", "Transformations", and "Adapters". The "Adapters" tab is selected, and a list of adapters is displayed in a table. The table has columns for "Name", "Last Updated", and "Modified By". The adapters listed are: DB Adapter, Dummy Adapter, File Adapter, HTTPWS Adapter, JMS Adapter, MQ Adapter, Service Item Listener Adapter, VMware Adapter, and Web Service Listener Adapter. The "DB Adapter" is highlighted. Below the table, there are navigation controls for the table, including "Page 1 of 1" and "Displaying 1 - 9 of 9".

Name	Last Updated	Modified By
DB Adapter	02/09/2012 04:59 PM	
Dummy Adapter	02/09/2012 04:59 PM	
File Adapter	02/09/2012 04:59 PM	
HTTPWS Adapter	02/09/2012 04:59 PM	
JMS Adapter	02/09/2012 04:59 PM	
MQ Adapter	02/09/2012 04:59 PM	
Service Item Listener Adapter	02/28/2012 12:04 AM	
VMware Adapter	02/09/2012 04:59 PM	
Web Service Listener Adapter	02/09/2012 04:59 PM	

Step 2 In the Name column, click the desired adapter.

The Manage Adapter page appears for the chosen adapter. The details for the Database Adapter are shown below.

Manage Adapter	
Name	DB Adapter
Description	The purpose of the DB Adapter is to allow creation of Agents that communicate with other systems via tables in a database. It supports polling a table in a database and reading messages found there, as well as writing messages to a table in a database.
Created On	02/10/2012
Created By	admin admin
Last Updated On	02/10/2012
Last Updated By	null
Message Support	Bidirectional
Inbound Model	Poller
Inbound Class	com.newscale.is.adapter.db.DBInboundAdapter
Outbound Class	com.newscale.is.adapter.db.DBOutboundAdapter
Exception Class	
Polling Interval (in milliseconds)	<input type="text" value="10000"/>
Retry Interval (in milliseconds)	<input type="text" value="0"/>
Maximum Attempts	<input type="text" value="0"/>

Most of these general properties should typically not be changed by Service Link developers. The “Polling Interval”, “Retry interval” and “Maximum Attempts” may need to be changed as per requirement. Any changes are inherited by all agents that use this adapter type.

Additional outbound and inbound properties are specified when the adapter is used in an agent. These properties are described in the [“Service Link Adapters” section on page 2-42](#).

Agents

The Create Agent Wizard walks you through configuring an agent. The wizard consists of eight pages; some pages may be skipped, depending on options chosen on the previous pages.

The pages of the Create Agent Wizard are summarized in [Table 2-2](#) below.

Table 2-2 Create Agent Wizard Pages

Page	Description/Usage
General Information	The name, action, and description for the agent, as well as other general information regarding its configuration and behavior
End Points	Adapters and transformations used by the agent
Outbound Adapter	Detailed configuration options for the outbound adapter
Inbound Adapter	Detailed configuration options for the inbound adapter
Outbound Request Parameters	Outbound parameters for all adapter types except the VMware adapter
Outbound Response Parameters	Parameters that are received in a synchronous response to an outbound message send to an http/ws adapter
Inbound Parameters	Parameters received as part of the inbound message
Summary	Summary page displaying all information previously entered

To create an agent:

- Step 1** From the Common Tasks area of the Service Link home page, click **Create Agent**, or choose **Manage Integrations > Agents > Create Agent**.

The General Information page of the Create Agent wizard appears. This is the first of eight pages that comprise the wizard. Some pages might not be relevant for a particular agent configuration, and can be skipped.

The screenshot shows the 'General Information' page of the Create Agent Wizard. At the top, it indicates 'Step 1 of 8' and has buttons for 'Back', 'Next', 'Save', and 'Cancel'. The main form area contains the following fields:

- Name:** A text input field.
- Action:** A text input field.
- Outgoing Content:** A dropdown menu with the selected value 'Data and Parameters; No Service Details (default; small)'.
- Failed Email:** A dropdown menu with the selected value 'None'.
- Context Type:** A dropdown menu with the selected value 'Service Task'.
- Description:** A large text area for entering a description.

Provide values for the fields described in [Table 2-3](#) below, then click **Next**.

Table 2-3 *Creating Agents – General Attributes*

Setting	Description
Name	A name for the agent. The name must be unique.
Action	A description of the action performed by the agent, for example, “Service Portal To Remedy - Create Ticket”. Although uniqueness is not enforced, actions should be unique, since they are presented in a pick-list when you are asked to assign the agent to a task.
Outgoing Content	An option for specifying which nodes of the outbound nsXML message should be generated for this agent. More details are given below.
Failed Email	An email template that is sent if the outbound message cannot be delivered to its destination. Email templates must be defined via the Notifications option of the Administration module.
Context Type	The type of agent. Service Link agents are “Service Tasks”, to allow the agent to be used as an external task in a delivery plan. “Service Item” agents are used to import the definition or data or service items as configured via Service Item Manager.
Description	A detailed description of the agent. A full description here can assist with supporting the integration.

Failed Email

The Failed Email notification can be generated in case an outbound message cannot be delivered. In addition to the standard sets of namespaces available for all task-related emails, Service Link Failed Emails can include details about the message being generated at the time of the failure. Including these namespaces in the email template may help in diagnosing the problem. Details on available namespaces are given in the *Cisco Service Portal Designer Guide*.

Failed email is not applicable to inbound messages; a notification is not generated when Service Link fails to process an inbound message.

Outbound Message Content

The outbound adapter generates an nsXML message that is stored in the Service Portal database. This message is then subject to a transformation, and the resultant external message is delivered via the specified adapter to the desired destination.

The format of the message is documented in [Chapter 3, “Service Link Adapter Development Kit”](#) and by the corresponding XML schema available on the application server at `ISEE.war/WEB-INF/classes/nsxml.xsd`. The complete message includes all information about the service request. Such messages can get quite large (easily exceeding 500 K, depending on the number of dictionaries and fields used in the service) and consequently consume large amounts of storage within the database, as well as consuming significant amounts of CPU to produce.

To reduce resource consumption, Service Portal offers the following options.

- The Administration Setting to “Compress messages” can compress Service Link messages stored in the database. This greatly reduces storage requirements, but potentially complicates troubleshooting, since messages are no longer human readable by a DBA or support personnel.

- Service Link Message Purge scripts are available. These scripts can purge messages for completed tasks from the transactional database. This reduces storage requirements for the messages in the database. Details on using the Message Purge scripts are given in the [“Service Link Troubleshooting and Administration”](#) section on page 2-66.
- The outgoing content can be configured to include only selected nodes of the nsXML message by manipulating the “Outgoing message content” property. This reduces memory and CPU requirements for processing outbound messages.

The default message content is “Data and parameters; no Service Details (small),” which generates a nsXML message that does not include content nodes describing the service requested. Agent parameters and transformation must be designed with the outgoing content type in mind, to ensure that all required content is included in the nsXML message. Specifically, if eliminating the dictionary data from the outbound message, agent parameters must be mapped to appropriate form fields (or constants). In cases where a service has many form fields that are not needed for an external task, the XML size reduction and associated CPU utilization reduction are substantial.

Outgoing content options are summarized in [Table 2-4](#) below.

Table 2-4 *Outgoing Message Content – Options*

Option	Description
All Message Details (large)	The complete Service Link message is generated.
Data, Form and Parameters (medium-large)	Information about the service and its tasks is omitted. The message is restricted to data (field values on the service form), form (complete metadata about the dictionaries and fields on the service form) and parameters (values supplied to agent parameters).
Data and Parameters (medium)	The message is restricted to information about the requisition, all data values entered on the service form and parameter values.
Data and Parameters; No Service Details (small)	The message is restricted to information about the requisition, all data values entered on the service form and parameter values (the default). The “small” option must be specified for the VMware adapter.
Only Parameters (minimal)	The message is restricted to information about the requisition and the agent parameters.

Adapter Selection

An agent may be configured to manage both outbound and inbound communications; just outbound communications; or just inbound data. It is possible to use different adapter types for each direction, for example, a database adapter could be used to write data outbound but that system would then respond by writing files into a directory that would be read by an inbound file adapter.

Once an adapter type is chosen, subsequent pages of the wizard are adjusted to display properties relevant to the adapter type and usage (outbound or inbound). Property values must be supplied as part of the agent definition.

The End Points page of the Create Agent wizard (page 2 of 8) allows you to designate the adapters to be used in the agent as well as any transformations to be applied to the outgoing or incoming message. Transformations must have previously been defined using the Transformations subtab of the Manage Integrations option.

Step 2 of 8 Back Next Save Cancel

End Points

Outbound Adapter:

Inbound Adapter:

Outbound Transformation:

Inbound Transformation:

Table 2-5 Creating Agents – End Point Properties

Setting	Description
Outbound Adapter	The default or custom adapter used for sending a message from Service Portal to the external system.
Outbound Transformation	The transformation (or none) to apply to the nsXML message produced by the outbound adapter, in order to generate an external message which can be understood by the external system
Inbound Adapter	The adapter to be used for receiving messages from an external system, or “Auto Complete” if no inbound message is expected.
Inbound Transformation	A transformation (or none) to apply to the incoming message in order to produce a nsXML message understood by the Business Engine; applicable to Service Tasks only.

Properties applicable to each type of adapter are described later in this chapter. In fact, the next two pages of the wizard, dedicated to configuring the outbound and inbound adapters, will vary, depending on which adapter has been chosen. However it is worth discussing two special cases: Dummy adapters and the auto complete option.

Dummy adapters can be configured within an agent as either the outbound or inbound adapter. If a dummy adapter is chosen, it means that the agent is only operating in a single direction. For example, an agent configured with a dummy inbound adapter means that the agent is only responsible for outbound communications. In turn, there could be a separate (inbound only) agent configured that would be dedicated to updating and closing tasks.

The auto complete option is available only as the choice for the inbound adapter part of the agent. Its effect is similar to choosing “Dummy adapter”, that is, the agent will only be managing outbound communications. The key difference is that after the outbound communication associated with the task has been sent, the task will automatically be completed and the rest of the delivery plan will continue to be executed.

Agent Parameters

Agent parameters may be used in conjunction with both outbound and inbound adapters.

Parameter mappings specified as part of the agent definition provide default value to be used. These mappings can be overridden on a service-by-service basis by editing the task definition for the service in Service Designer.

Outbound Request Parameters

Agent parameters used in an outbound message provide a way to supplement the content nodes in the standard nsXML outbound message with additional data and to organize content nodes in an easy-to-address format. The parameters are easily accessible via the XSL transformation, to allow their inclusion in the external message.

A parameter mapping is assigned by typing the source elements in the Service Data Mapping area, or by building an expression by using the elements available in the drop-down lists to the left of the Parameter Mappings pane. A mapping may consist of a combination of:

- A constant value
- A dictionary field on a service form
- A nsXML element
- A prebuilt function applied to one of the above elements

Mapping an agent parameter has the following advantages:

- A transformation that extracts that parameter value does not need to refer to the name of the dictionary field in which the value was supplied, but may refer simply to the agent parameter by name. This encourages agent reuse across different services and dictionaries.
- A smaller outbound message content type may be used, provided all other content required in the message is supported, since parameters are included in all content types.
- nsXML elements and XPATH operations that would not be accessible without using a transformation can be included in the external message.

To create an outbound parameter:

Step 1 On the Outbound Request Parameters page, click **Add Mapping**.

The Edit Parameter Values dialog box appears at the bottom of the page.

- Step 2** Enter a name for the parameter.
Parameter names can include spaces, but should not include special characters (such as “>” or “&”), which have significance in XML messages.
- Step 3** Specify the value/mapping for the parameter, using the guidelines given below.

Constant Values

Sometimes a constant value that is not dependent on the requisition or service details must be passed to the external system. For example, if the system needs a name for the source of the external system, “Service Portal” or “Request Center” can simply be typed as the Service Data Mapping (without the quotation marks).

Outbound nsXML Mappings

Selected elements of the standard nsXML outbound message are available to be mapped to agent parameters. These are summarized in the table below.

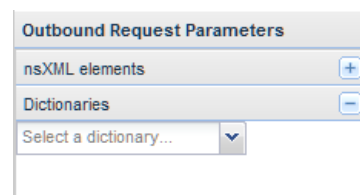
Table 2-6 *Outbound nsXML Mappings*

nsXML Element	Contents/Description
Customer	The login name of the customer for the requisition
Initiator	The login name of the requestor (initiator) of the requisition
requisition-entry-id	Number that uniquely identifies a service request within a requisition; typically used to differentiate multiple services within a shopping cart
expected-cost	Transactional price for the service
expected-duration	Service standard duration
requisition-id	Number that uniquely identifies the requisition; referenced in My Services and Service Manager
channel-id	Globally unique identifier (GUID) that identifies an external task

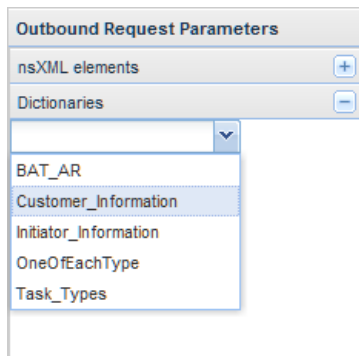
Dictionary Mappings

To map an agent parameter to a dictionary field:

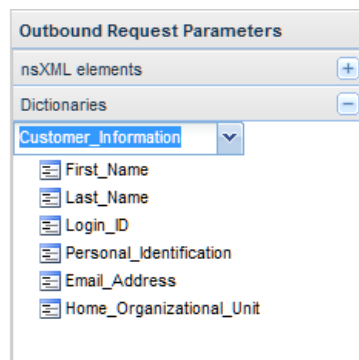
- Step 1** Expand the Dictionaries node so that the “Select a dictionary” option appears.



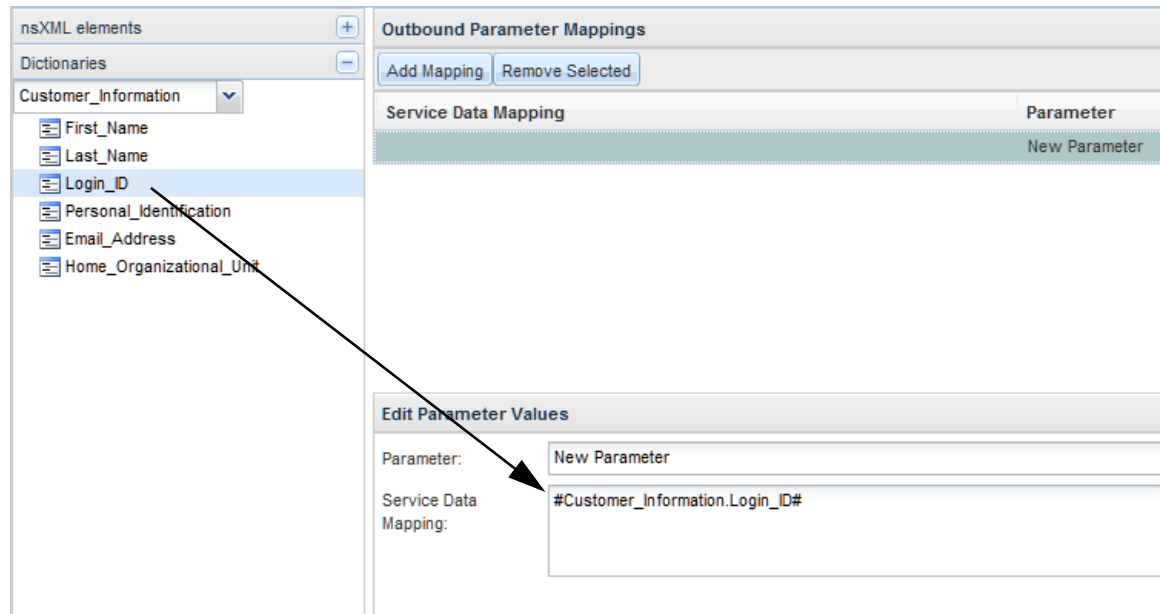
- Step 2** Click the **Select a dictionary** drop-down menu to display a list of all Request Center dictionaries.



- Step 3** Choose the dictionary containing the field to be mapped to the agent parameter. A list of all fields in the dictionary appears.



- Step 4** Click the field to be mapped to the agent parameter and drag it to the Service Data Mapping text area. When the drag icon changes to a green check mark, release the mouse. A lightweight namespace for the selected field appears.



Since the agent is defined independent of a service, with the exception of grid dictionaries, any dictionary field can be chosen. It is the responsibility of the service designer to ensure that the referenced dictionary is, in fact, included in the service in which this agent is used.

For any integration that passes the contents of one or more grid dictionaries, you will need a transformation to handle the rows in each grid dictionary, which are stored as multiple dictionary instances. They follow the naming convention of “DictionaryName-n”, where n is the grid row number, in the <data-values> section of the outbound nsXML. For example, if a grid dictionary named VMOperation has two rows of data in the service request, the values are represented as below:

```
<data-values>
  <data-value multi-valued="true">
    <name>VMOperation-1.Name</name>
    <value>vmgw01</value>
  </data-value>
  <data-value multi-valued="false">
    <name>VMOperation-1.GuestOS_Name</name>
    <value>winNetStandardGuest</value>
  </data-value>
  <data-value multi-valued="false">
    <name>VMOperation-1.CPUCount</name>
    <value>1</value>
  </data-value>
  <data-value multi-valued="false">
    <name>VMOperation-1.Memory</name>
    <value>2048 MB</value>
  </data-value>
  <data-value multi-valued="true">
    <name>VMOperation-2.Name</name>
    <value>vmgw01</value>
  </data-value>
  <data-value multi-valued="false">
    <name>VMOperation-2.GuestOS_Name</name>
    <value>winNetStandardGuest</value>
  </data-value>
  <data-value multi-valued="false">
    <name>VMOperation-2.CPUCount</name>
    <value>1</value>
  </data-value>
  <data-value multi-valued="false">
    <name>VMOperation-2.Memory</name>
    <value>2048 MB</value>
  </data-value>
</data-values>
```

There is no support for inbound agent parameter mapping and update to grid dictionary fields.

Prebuilt Functions

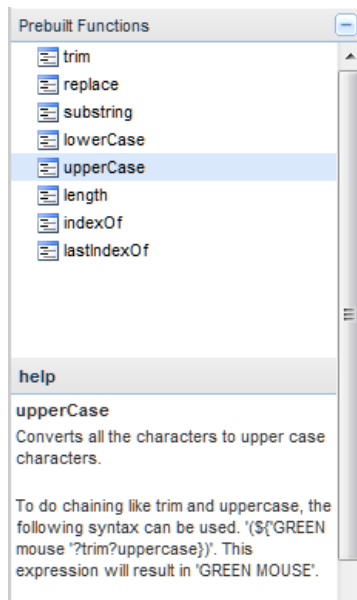
Prebuilt functions can be applied to the mapped elements, so that the parameter value fits the semantics or formatting requirements expected in the target system. For example, a field may be shortened, by applying a substring function, if the data definition for the field in the target system accommodates fewer characters than are maintained in Service Portal. Prebuilt functions are summarized below and explained in more detail in the [“Prebuilt Functions” section on page 2-69](#).

Table 2-7 Prebuilt Functions

Function	Usage/Description
trim	Trims leading or trailing spaces from the value; especially useful for form data and for incoming messages from the database adapter, which encloses values in CDATA tags.
replace	Replaces all occurrences of one character or pattern with another.
substring	Selects a portion of the string, specified by a starting point and optional length.
lowerCase	Converts the value to all lower case.
upperCase	Converts the value to all capital letters.
length	Returns the number of characters in the string.
indexOf	Returns the index within this string of the first occurrence of the specified substring.
lastIndexOf	Returns the index within this string of the last occurrence of the specified substring.

To apply a prebuilt function to an agent parameter:

- Step 1** Expand the Prebuilt Functions node so that the function names appear.
- Step 2** Highlight the function you want to use—notice that Help appears, explaining function usage, at the bottom of the pane.



- Step 3** Drag the function into the Service Data Mapping box for the parameter. When the drag icon changes to a green check mark, release the mouse.

Edit Parameter Values

Parameter:

Service Data Mapping:

Apply

Step 4 The function is defined, with `$Parameter$` as a placeholder for the actual value.

Edit Parameter Values

Parameter:

Service Data Mapping:

Apply

Step 5 Replace the placeholder with the dictionary field or element of the nsXML message to be used. You need to drag the field or nsXML element to the Service Data Mapping text box, then manually edit the parameter definition.

Edit Parameter Values

Parameter:

Service Data Mapping:

Apply

Outbound nsXML with Agent Parameters

Agent parameters are added to the end of the outbound nsXML message. For example, agent parameters shown below generate the nsXML snippet immediately following.

Service Data Mapping	Parameter
\$initiator\$	Initiator
\$expectedCost\$	Transactional Price
\$customer\$	Customer

```
<agent-parameter multi-valued="false">
  <name>Initiator</name><value>ltierstein</value></agent-parameter>
<agent-parameter multi-valued="false">
  <name>Transactional Price</name><value>0.0</value></agent-parameter>
<agent-parameter multi-valued="false">
  <name>Customer</name><value>Customer</value></agent-parameter>
```

Outbound Response Parameters

Outbound response parameters may be used in conjunction with an outbound http/web service adapter. If the adapter's Process Response setting is true, the response to the original request is processed. That response may include a “Send Parameters” message type. Parameters are defined as for Inbound Agent Parameters.

Inbound Agent Parameters

When used in conjunction with an inbound “Send Parameters” message type, agent parameters allow the external task to update the dictionary field to which the parameter is mapped.

Table 2-8 Inbound Agent Parameter Settings

Setting	Description
Parameter	The parameter name.
Dictionary Field	Select a dictionary field from the drop-down list that displays a list of all dictionary fields. The field name is in the format: DictionaryName.FieldName without enclosing hash marks (#).
Mapping	A prebuilt function applied to derive the value that should be used to update the specified dictionary field.
Mandatory	Check Mandatory if the field must be present in the Inbound message. This check box is typically unchecked if a change is made to a service and the parameter is no longer required. Obsolete parameters should not be deleted.

Transformations

To create a transformation:

-
- Step 1** On the Manage Integrations tab, click the **Transformations** subtab.
 - Step 2** Click **Create Transformation**.
- The Create Transformation page appears.

* Name:

* Description:

* Direction:

Created By:

Created On:

Updated by:

Last Updated:

Request Response

A transformation may be applied to either an outbound and inbound message, by designating the Direction. Two transformations may need to be used for web services outbound messages—one is applied to the outbound request and the second may be applied to a response to that request, if the Process Response setting is turned on.

The Validate button parses the XSLT to ensure that the transformation is well-formed. If it is not (for example, if an XML tag is misspelled or missing), a diagnostic message appears. You need to fix the error before saving the transformation.

However, Service Link does not validate the transformation. For example, no error is detected if a transformation refers to an element that does not exist in the source message; a well-formed XML message would be produced, but it would not be valid for the target system. Therefore, a runtime error would be produced if Service Link produced an external message that was not recognizable by the target application or an inbound message that was not recognizable to the Business Engine.

If you have access to an XML development environment and are familiar with its usage, it may be efficient to use that environment to test the transformation. For an outbound transformation, simply copy the nsXML produced by the agent (before you apply a transformation) into the XML development environment and use this as the source XML. Once you have validated the transformation, copy and paste the XSLT code into a Service Link transformation and associate it with the appropriate agent.

The process of manually developing and debugging a transformation is eliminated for outbound web service integrations that are developed using the Integration Wizard. A transformation is automatically created that will transform outbound nsXML into a format compatible with the specified WSDL for the web service. However, if the wsdl or integration requirements change, you will need to follow the steps outlined above to update the transformation.

Step 3 Provide values for the fields described in [Table 2-9](#) below.

Table 2-9 Transformation Settings

Setting	Description
Name	A name for the transformation. Transformation names should be descriptive of the nature of the interface. They may include the names of the source and target systems, for example “Service Portal To Remedy”.
Direction	Inbound or Outbound.
Description	A description of the transformation; required.
Request subtab	The XSLT code which is applied to the nsXML message for outbound adapters and to the external message for inbound adapters.
Response subtab	The XSLT code that is applied to the response received from an http/web service request if the <i>Process Response</i> setting for the agent is set to true.

Step 4 Click **Validate** to check that the transformation contains well-formed XSL.

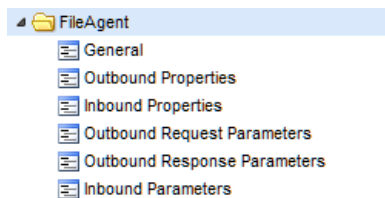
Step 5 Click **Save**.

Reviewing Agent Definitions and Property Sheets

You can review or revise any agent definition, whether the agent was created in Service Link or via the Integration Wizard. Once the Agents subtab of the Manage Integrations tab appears, you can either:

- Use the List pane on the left-hand side of the page and click the agent name.
- Scroll through agent information listed on the right-hand side of the page and click the agent name.

The Agent entry in the list pane is expanded. Click the property sheet for that portion of the agent definition to be edited or reviewed. Once the property sheet appears, enter any changes and click **Save** to save them.



Clicking on the agent name provides an overview of the agent definition:

Agent Information	
Name:	FileAgent
Action:	FileAction
Status:	Not sending or receiving
Description:	This is a file agent
Service Tasks using this agent	
Service	Name
Configurable Tasks File Adapter	Second Task
Configurable Tasks File Adapter	Delivery Task
Configurable Tasks File Adapter	Service Group Review
Configurable Tasks File Adapter	Service Group Authorization
<input type="button" value="Reset Agent Parameters"/> <input type="button" value="Page 1 of 1"/> <input type="button" value="Display"/> 	

This page is read-only except for the button to Reset Agent Parameters.

Configuring a Task to use a Service Link Agent

Once the agent has been defined, it can be used in a service by creating an external task whose workflow invokes the agent. Once the workflow has been configured, you may review or override any agent parameters defined for the included agent.

Creating an External Task

To direct a task to an external (third-party) application using a Service Link agent:

- Step 1** Start Service Designer. Select the service that is to include the external task. Click the **Plan** tab.
- Step 2** You can use either the Tasks subtab or the Graphical Designer subtab to specify the external task.
 - Using the Tasks subtab for the service, define the task and place it in the correct sequence in the delivery plan.
 - Using the Graphical Designer subtab, create a task on the diagram, and place it in the correct sequence in the diagram using the Associate tools. Double-click the task to display its property sheet.
- Step 3** Using the Workflow Type drop-down list on the General tab, select the desired action from the drop-down list. Note that it is the defined actions in the Service Link module and not the agent names that are listed in this drop-down.
- Step 4** Save the workflow/task plan. If you were using the Workflow Designer, return to the Tasks subtab.
- Step 5** Once you have saved the external task, an ellipsis button appears next to the Workflow Type. Clicking the ellipsis allows you to review the parameter mappings currently in effect for the agent (if any) or to change these mappings for this specific service. Click the **ellipsis** button.

General	Participants	Email	Task Instructions	Checklist
<input type="button" value="Save"/>				
Workflow Type:	<input type="text" value="Update Inventory"/> <input type="button" value="..."/> <input type="button" value="Create Agent"/>			
Task name:	<input type="text" value="Update Inventory"/>			

- Step 6** The Agent Parameter Override popup window appears. Review the agent parameters or change the mapping for one or more parameters. Be sure to click **Apply** as you change each parameter and **Save** before closing the window.

Service Data Mapping	Parameter
#RAPI_GetServiceDef.RQ_Password#	RQ_Password
#RAPI_GetServiceDef.RQ_Username#	RQ_Username
#RAPI_GetServiceDef.RQ_commentText#	RQ_commentText
#RAPI_GetServiceDef.RQ_loginUserName#	RQ_loginUserName
#RAPI_GetServiceDef.RQ_requisitionId#	RQ_requisitionId

- Step 7** You may wish to define a performer (person, queue, or functional position) on the Participants tab. The calendar of that performing entity will then be used to calculate the Due Date of the task. If you do not set a value on the Participants tab for external tasks, the calendar of the Default Service Queue is used to calculate the Due Date. In this fashion, due dates are set in the plan and Service Portal can calculate delivery Operating Level Agreement (OLA) compliance for external tasks and compliance with the Service Level Agreement (SLA) for services containing such tasks.

Synchronizing Agent Mappings and Service Definitions

When you create and save a task that uses an agent, the agent parameter mappings specified for the agent are automatically inherited by that individual task. As described above, a service designer may override any of the agent mappings at the task level.

However, if the agent is subsequently modified, to include a different set of agent parameter mappings, such changes are not automatically inherited by tasks that were previously defined to use the agent. Such changes may include:

- Adding an agent parameter
- Deleting an agent parameter
- Changing the mapping of an existing agent parameter

Propagating these changes to services that use the agent can be automated, by following the procedure below:

- Step 1** From the Service Link Manage Integrations tab, click the **Agents** subtab.
- Step 2** Click the name of the agent whose parameter mappings have been changed.
The Agent Information page appears.

Agent Information

Name: FileAgent
Action: FileAction
Status: Not sending or receiving
Description: This is a file agent

Service Tasks using this agent

Service	Name
Configurable Tasks File Adapter	Service Group Authorization
Configurable Tasks File Adapter	Service Group Review
Configurable Tasks File Adapter	Delivery Task
Configurable Tasks File Adapter	Second Task

Reset Agent Parameters

Page 1 of 1

Displaying 1 - 4 of 4

- Step 3** Choose the service or services whose agent parameter mappings need to be resynchronized with the updated agent definition.
- Step 4** Click **Reset Selected Tasks**. This button automatically resets all parameter mappings to their agent defaults, so any task-specific mappings would need to be reapplied.

nsXML Messages

The transformation must not only contain well-formed XML, it must produce a well-formed and valid nsXML message. All nsXML messages must conform to the nsXML schema (an XML document that describes the structure of an XML document). The schema is available on the application server at ISEE.war/WEB-INF/classes/xsl/nsxml.xsd.

Outbound nsXML Message

When an external task moves to a status of Ongoing, an outbound nsXML message is generated.

The generated nsXML for each message can be viewed in the Service Link module, by clicking on the nsXML message in the Message Details page. It contains task related data as well as data associated with the parent requisition.

The most important element within the nsXML is the channel-id, an ID that uniquely identifies the external task. This ID is provided to the third-party system and needs to appear in their response if the corresponding data update is to be successfully applied by the business engine.

The channel-id is formatted as a Globally Unique Identifier (GUID). GUIDs are most commonly written in text as a sequence of hexadecimal digits such as:

```
3F2504E0-4F89-11D3-9A0C-0305E82C3301
```

This text notation consists of 5 sets of data, each separated by a hyphen. The GUID consists of 32 characters plus 4 hyphens, for a total length of 36 characters.

There are two outbound message types.

task-started

The task-started message type is generated when an external task is started. A detailed description of the elements of the task-started message is available in [Chapter 3, “Service Link Adapter Development Kit”](#).

```
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
  <task-started task-type="task">
    <task>
      .
      .
      .
    </task>
  </task-started>
</message>
```

task-canceled

The task-canceled message type is generated when request which includes an external task is cancelled. If the user is not allowed to cancel a request once the external task has been performed (via the corresponding setting in the service's delivery plan), this message would never be generated. If, however, canceling the request is allowed, the transformation used in the agent responsible for the external task must be “smart enough” to handle both a task-started and task-canceled message. The transformation would need to test for the task-canceled message type and to send an appropriate message to the external system:

```
<xsl:if test="/message/task-started">
  <!-- Original XSLT goes here-->
</xsl:if>

<xsl:if test="/message/task-canceled">
  <!-- XSLT for the cancel message goes here-->
</xsl:if>
```

Inbound nsXML Message

Two types of operations are supported for inbound messages from the third-party system—requisition operations and service item operations. Requisition operations are used for the update of request data and task status. Service item operations are used for the addition, modification, deletion, and retrieval of service items associated with the request. Certain operations may be combined in one inbound message, known as a “Composite Message”. The details and restrictions for each operation are described in the following sections.

Requisition Operations

The third-party system may send one or multiple inbound messages for an external task by referencing the channel-id of the corresponding outbound message. The external task is completed when one of the take-action operations is sent and this allows the next task in the authorization/delivery plan to proceed.

take-action

A take-action message may be applied to an authorization or delivery task, to change the status of the task. The action attribute of the take-action tag identifies the action to be taken. Valid actions are summarized in [Table 2-10](#) below.

Table 2-10 Take-action Messages

Action	Task Type	Description
done	Delivery task	Mark the delivery task as completed.
cancel	Delivery task	Cancel the delivery task.
ok	Review task	Mark the review as completed.
reject	Authorization task	Reject the authorization.
approve	Authorization task	Approve the authorization.

When the last delivery task in a task plan is marked as done, the requisition is closed (completed). An approval task can be marked as Approved or Rejected, by setting the “action” attribute of the take-action tag to the corresponding value.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
  <take-action action="done"/>
</message>
```

send-parameters

Parameters are data elements that are bound to dictionary fields within the agent definition. The send-parameters message type allows one or more specified parameters to be updated which, in turn, updates the corresponding dictionary fields in the service. Using this type of inbound message is the preferred way for the external system to update dictionary fields used in a service request.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
  <send-parameters>
    <agent-parameter>
      <name>Status</name>
      <value>Resolved</value>
    </agent-parameter>
    <agent-parameter>
      <name>ResolvedBy</name>
      <value>Help Desk</value>
    </agent-parameter>
  </send-parameters>
</message>
```

add-comments

An add-comments message is used to add comments to the System Comments section of the requisition.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
  <add-comments>
    <comment>Test Comment</comment>
  </add-comments>
</message>
```

Service Item Operations

Service items are entities defined in Lifecycle Center. Their lifecycles are associated with service requests—from the point the service item instances are provisioned, to the point when they are decommissioned. In the cases when the service item lifecycle events are handled by external systems, service item data can be synchronized with Lifecycle Center via Service Link service item create, update, delete, and get messages. These message types are supported only through the web service-based Service Item Listener Adapter (see the “[Service Item Listener Adapter](#)” section on page 2-56).

One or more service item types and service item instances can be included in these messages. Multiple service item operations cannot be combined in a message. In other words, create, update, or delete operations have to be sent in separate inbound messages. Whenever an error condition is encountered, all the service item operations in the same message are rolled back.

Service item attributes of datetime type must be specified in the format YYYY-MM-DD HH:MI:SS or YYYY-MM-DD. All times are stored as UTC time.

create

Service item subscriptions can be included optionally at the time a service item instance is created. 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 values for either the customer login ID or Organizational Unit name are specified in the subscription section of the message, those values are used to override the default service item assignment. For more details about subscription processing rules, see the Service Designer chapter in the *Cisco Service Portal Designer Guide*.

```
<?xml version="1.0" encoding="UTF-8" ?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
<create>
  <serviceitem>
    <name>LaptopComputer</name>
    <serviceItemData>
      <serviceItemAttribute name="Name">LT-LENT60-17032</serviceItemAttribute>
      <serviceItemAttribute name="Model">Thinkpad T60</serviceItemAttribute>
      <serviceItemAttribute name="Brand">LENOVO</serviceItemAttribute>
      <serviceItemAttribute name="Price">899.99</serviceItemAttribute>
      <serviceItemAttribute name="Memory">3</serviceItemAttribute>
      <serviceItemAttribute name="ManufactureDate">2009-04-15
12:00:00</serviceItemAttribute>
    <subscription>
      <loginID>jsmith</loginID>
      <ouname>Finance</ouname>
    </subscription>
  </serviceItemData>
</serviceitem>
<serviceitem>
  <name>DesktopComputer</name>
  <serviceItemData>
    <serviceItemAttribute name="Name">DT-DELLV200-02274</serviceItemAttribute>
    <serviceItemAttribute name="Model">Vostro 200</serviceItemAttribute>
    <serviceItemAttribute name="Brand">DELL</serviceItemAttribute>
    <serviceItemAttribute name="Price">755.99</serviceItemAttribute>
    <serviceItemAttribute name="Memory">4</serviceItemAttribute>
    <serviceItemAttribute name="ManufactureDate">2010-03-01
12:00:00</serviceItemAttribute>
  </serviceItemData>
</serviceitem>
</create>
</message>
```

update

In update messages, omitting a service item attribute results in no change to the attribute value. When an attribute is explicitly specified in the message but contains no value, the value of the attribute for the service item is set to blank for text fields and zero for numeric fields.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
<update>
  <serviceitem>
    <name>LaptopComputer</name>
    <serviceItemData>
      <serviceItemAttribute name="Name">LT-LENVT60-6122</serviceItemAttribute>
      <serviceItemAttribute name="Memory">4</serviceItemAttribute>
      <subscription>
        <loginID>dcohen</loginID>
      </subscription>
    </serviceItemData>
  </serviceitem>
  <serviceitem>
    <name>DesktopComputer</name>
    <serviceItemData>
      <serviceItemAttribute name="Name">DT-DELLV200-00394</serviceItemAttribute>
      <subscription>
        <loginID></loginID>
        <ouname></ouname>
      </subscription>
    </serviceItemData>
  </serviceitem>
</update>
</message>
```

delete

Delete service item requests require only the names for the service item type and instance. Additional service item attribute and subscription information is ignored.

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
<delete>
  <serviceitem>
    <name>LaptopComputer</name>
    <serviceItemData>
      <serviceItemAttribute name="Name">LT-TOSH900-0021</serviceItemAttribute>
    </serviceItemData>
  </serviceitem>
  <serviceitem>
    <name>DesktopComputer</name>
    <serviceItemData>
      <serviceItemAttribute name="Name">DT-DELLV100-0394</serviceItemAttribute>
    </serviceItemData>
  </serviceitem>
</delete>
</message>
```

getRequest

The getRequest operation is used for retrieving service item instances. The channel-id and topic-id attributes are optional, unlike the create/update/delete service item requests. Each inbound message may contain only one getRequest operation and within it, only one service item type. There is no logging of the request as seen in the Service Link user interface.

Service item instances are retrieved according to the search filters specified in the request XML, using the service item attributes and subscription data (that is, Customer Login ID and Organizational Unit Name). Up to five filters may be used in a `getRequest` and they are interpreted as AND joins. Table 2-11 below shows the operators that are supported in search filters.

Table 2-11 Search Filter Operators for `getRequest`

Datatype	Supported Filter Operators
STRING(32)	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
STRING(128)	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
STRING(512)	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
INTEGER	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
MONEY	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
LONG INTEGER	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
DOUBLE FLOAT	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, StartsWith, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo
DATE TIME	Equals, LessThan, LessThanOrEqualTo, GreaterThan, GreaterThanOrEqualTo, EqualsIgnoreCase, isNull, isNotNull, Between, NotEqualsTo

The response data from the `getRequest` contains service item attribute names and values, as well as its subscription information. The maximum number of records returned in each `getRequest` operation is 100. The next set of records can be retrieved by specifying the `startRow` and `count` elements in the request. The `startRow` element indicates the beginning row number of the result set. The `count` element indicates the number of records to be returned. The `startRow` and `count` values are defaulted to 1 and 100, respectively, if they are absent in the request XML. The value for `count` is limited to 100 for performance reasons.

Here is an example of the `getRequest` XML:

```
<getRequest>
  <serviceItemType>LaptopComputer</serviceItemType>
  <startRow>1</startRow>
  <count>1</count>
  <subscription>
    <loginID>jsmith</loginID>
    <ouname>Finance</ouname>
  </subscription>
  <filters>
    <!--1 to 5 repetitions:-->
    <filter attributeName="Name" operator="Equals" value="LT-LENT60-17032" />
    <filter attributeName="Price" operator="GreaterThan" value="800" />
    <filter attributeName="ManufactureDate" operator="GreaterThan" value="2004-04-10" />
  </filters>
</getRequest>
```

Response for the above request:

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body xmlns:ns1="http://externaltask.api.newscale.com">
    <response channel-id="CHANNELID not retrieved"
xmlns="http://externaltask.api.newscale.com">
      <status-code>success</status-code>
      <status-message>Service item data read successfully.</status-message>
      <getResponse>
        <serviceitem>
          <name>LapTopComputer</name>
          <serviceItemData>
            <serviceItemAttribute
name="Name">LT-LENVT60-17032</serviceItemAttribute>
            <serviceItemAttribute name="Brand">LENOVO</serviceItemAttribute>
            <serviceItemAttribute name="Memory">3</serviceItemAttribute>
            <serviceItemAttribute name="Model">Thinkpad T60</serviceItemAttribute>
            <serviceItemAttribute name="Price">899.99</serviceItemAttribute>
            <serviceItemAttribute name="ManufactureDate">Fri Apr 16 00:00:00
GMT+05:30 2004</serviceItemAttribute>
            <subscription>
              <loginID>jsmith</loginID>
              <ouname>Finance</ouname>
              <requisitionID>0</requisitionID>
              <requisitionEntryID>0</requisitionEntryID>
              <assignedDate>2012-07-20T05:21:29.187+05:30</assignedDate>
              <submittedDate>2012-07-20T05:17:21.503+05:30</submittedDate>
            </subscription>
          </serviceItemData>
        </serviceitem>
      </getResponse>
    </response>
  </soap:Body>
</soap:Envelope>

```

getDefinitionRequest

The `getDefinitionRequest` operation is used for retrieving the metadata or definition of a service item type. Like the `getRequest` operation, the `channel-id` and `topic-id` attributes are optional. Each inbound message may contain only one `getRequestDefinition` operation, and within it, only one service item type. There is no logging of the request as seen in the Service Link user interface.

Here is an example of the service item `getDefinitionRequest`:

```

<getDefinitionRequest>
  <serviceItemType>LaptopComputer</serviceItemType>
</getDefinitionRequest>

```

Response for the above request:

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body xmlns:ns1="http://externaltask.api.newscale.com">
    <response channel-id="" xmlns="http://externaltask.api.newscale.com">
      <status-code>success</status-code>
      <status-message>Service item definition read successfully.</status-message>
      <getDefinitionResponse>
        <serviceItemDef>
          <name>LaptopComputer</name>
          <classification>Laptops</classification>
          <displayName>LaptopComputer</displayName>
          <serviceItemProperty name="Name" type="string" />
          <serviceItemProperty name="Model" type="string" />
        </serviceItemDef>
      </getDefinitionResponse>
    </response>
  </soap:Body>
</soap:Envelope>

```

```

        <serviceItemProperty name="Brand" type="string" />
        <serviceItemProperty name="Price" type="real64" />
        <serviceItemProperty name="Memory" type="sint32" />
        <serviceItemProperty name="ManufactureDate" type="datetime" />
    </serviceItemDef>
</getDefinitionResponse>
</response>
</soap:Body>
</soap:Envelope>

```

Composite Messages

The above message types can be combined in a single inbound message. Such a combination is known as a “composite” message. The order of execution matters; you must send the parameters or add comments before including the take-action tag, and place the service item operation tags last.

```

<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
  <add-comments>
    <comment>Task closed per override ...</comment>
  </add-comments>
  <send-parameters>
    <agent-parameter>
      <name>Status</name>
      <value>Resolved</value>
    </agent-parameter>
  </send-parameters>
  <take-action action="done" />
  <update>
    <serviceitem>
      <name>LaptopComputer</name>
      <serviceItemData>
        <serviceItemAttribute name="Name">LT-LENVT60-6122</serviceItemAttribute>
        <serviceItemAttribute name="Memory">4</serviceItemAttribute>
        <subscription>
          <loginID>dcohen</loginID>
        </subscription>
      </serviceItemData>
    </serviceitem>
  </update>
</message>

```

SIM Import Messages

A Service Item Manager (SIM) Import message type supports importing service item and standards definitions and data from an external system into Service Portal. Unlike the service item create/update/delete operations, SIM import is based on the File Adapter protocol which polls for incoming files located in a specific directory. In addition to service item instance operations, SIM Import also supports the maintenance of service item groups and service item types. For details on Service Item Manager imports, see the *Cisco Service Portal Designer Guide*.

Transformations and nsXML

Outbound nsXML messages will typically be quite large and complex, often in excess of 500 KB. Although it is not mandatory to use transformations to alter the message format, it is unlikely that external systems would be configured to read nsXML. Consequently using transformations to alter the outbound message formats is normally unavoidable.

However as formats for inbound messages will probably be negotiated with those responsible for the third-party system, it is quite possible that a specification could be agreed that aligns closely to the nsXML message formats. If this is the case, the Inbound transformation could be much simpler than the corresponding outbound one.

Although we refer to XSL Transformations (XSLT), the technology used is actually called eXtensible Stylesheet Language and also includes XPATH. XPATH is a language for finding information and navigating through elements and attributes in an XML document. XPATH includes built-in functions for string values, numeric values, date and time comparison, sequence manipulation, Boolean values and other methods.

Creating and Deploying a Service Link Agent

The procedure below shows the typical sequence of tasks required to deploy a Service Link integration using a file adapter. It can also be used to validate a Service Link installation.

-
- Step 1** From the Common Tasks area of the Service Link home page, create an agent that uses an outbound file adapter by clicking the **Create Agent** wizard, filling in the location fields and supplying other outbound adapter properties. (Details on these properties are explained in the [“File Adapter” section on page 2-48.](#))
 - Step 2** Start the agent by navigating to the **Control Agents** tab, locating the agent, choosing it by clicking the mouse anywhere on that line except on the agent name (which is a link to the agent definition) and clicking **Start Selected** at the upper right of the page. Did it start? If not, one of your Service Link configuration settings is wrong or the Integration Server (ISEE) did not start correctly.
 - Step 3** Verify that the file directories you entered exist on the application server; if not, create them. Assure that both Service Portal and the external application have appropriate access (write or read) to the directories. If these conditions are not met, file transmission will fail at runtime.
 - Step 4** Go to Service Designer and create a service to use this agent.
 - Step 5** Go to My Services and order the service.
 - Step 6** If the requisition is created successfully, congratulations! the ISEE outbound queue is working. If you get an “our apologies” page, the JMS queues are not working.
 - Step 7** Go to the Messages page, accessible from the View Transactions tab. If you see messages from the requisition you just created, congratulations. Your message should have status of “Message sent”.
 - Step 8** Go to the outbound files directory (for example, C:\cisco\SL\OutboundFiles). If there is an XML file there (verify the date time stamp of the XML file to make sure that it is a new one corresponding to your requisition), your outbound trip for the file agent is completed. Congratulations! The outbound XML file would be a valid nsXML message.
 - Step 9** For your requisition in the Message Type column, click the **Execute Task** link. The Message Details page appears.
 - Step 10** Verify that the Requisition ID is correct. Copy the “Channel ID” from the message details screen.
 - Step 11** Create an XML file named SampleInbound.xml as follows. Where it says “insert your Channel ID here”, paste the value of the Channel ID that you copied in the last step. (Leave the double-quotes intact).

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="insert your Channel ID here">
  <take-action action="done"/>
</message>
```

For example, after pasting the Channel ID value, the SampleInbound.xml file would look something similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="3F2504E0-4F89-11D3-9A0C-0305E82C3301">
  <take-action action="done"/>
</message>
```

- Step 12** Put the SampleInbound.xml file in the inbound files directory (for example, C:\cisco\SL\InboundFiles).
 - Step 13** When the File Agent polls for input, it will automatically pick up the inbound file. (The default setting for the File Adapter's Polling Interval Time is every 10 seconds.) If the SampleInbound.xml file is processed successfully, it will disappear from the directory.
 - Step 14** Go to the Messages page of the View Transactions tab and look for your requisition. If there is another message for your requisition with the Type=Take Action, and the Status=Inbound Message Completed, then you have achieved a roundtrip.
 - Step 15** Click the **Requisition ID** link to open the Requisition Status page. Verify that your requisition has the status of "Closed (1 of 1 completed)".
-

Monitoring Service Link Transactions

There are multiple ways to monitor Service Link usage:

- The Service Link home page shows a graph of message volume over the last 30 days and provides Common Tasks and the View Transactions tab to access other monitoring options.
- The option to view Recent Failed Messages, also on the Service Link home page, shows all messages that could not be delivered.
- The option to view Messages, accessible from the View Transactions tab, shows all messages sent to or received by Service Link, and allows administrators to filter and search to show messages of interest.
- The option to view External Tasks, accessible from the View Transactions tab, shows all tasks that remain ongoing because a Service Link message could not be delivered, and allows administrators to filter and search to show tasks of interest.

All Service Link monitoring/administration pages are displayed using configurable "data tables". The appearance of these tables (the columns displayed, the width of each column and the order in which data is presented) can be customized. In addition, Filter and Search capabilities allow administrators to view only those rows which are of interest.

Viewing Messages from the Service Link Home Page

The Recent Failed Messages pane of the Service Link home page displays Service Link messages that could not be delivered to their destination within the past 30 days. By default, messages are displayed in reverse chronological order based on the date and time when they were sent.

Recent Failed Messages						
Message Type	Req ID	Agent	Task Subject	Date ▾	Resent On	Status Text
Take Action	166	Douglas_DB_Agent	configurable task d...	12/05/2011 03:55 PM		Inbound Message l
Take Action	166	Douglas_DB_Agent	configurable task d...	12/05/2011 03:54 PM		Inbound Message l
Composite	181	Douglas_DB_Agent	configurable task d...	12/05/2011 03:51 PM		Inbound Message l
Composite	180	Douglas_DB_Agent	DT1	12/05/2011 03:51 PM		Inbound Message l
Composite	179	Douglas_DB_Agent	configurable task d...	12/05/2011 03:50 PM		Inbound Message l
Composite	178	Douglas_DB_Agent	configurable task d...	12/05/2011 03:49 PM		Inbound Message l
Composite	177	Douglas_DB_Agent	configurable task d...	12/05/2011 03:47 PM		Inbound Message l
Unknown		Douglas_DB_Agent		12/05/2011 03:21 PM		Unknown Channel
Take Action	172	Douglas_DB_Agent	configurable task d...	12/02/2011 02:58 PM		Inbound Message l
Take Action	171	Douglas_DB_Agent	configurable task d...	12/02/2011 02:56 PM		Inbound Message l
Take Action	170	Douglas_DB_Agent	DT1	12/02/2011 02:06 PM		Inbound Message l
Take Action	170	Douglas_DB_Agent	DT1	12/02/2011 02:05 PM		Inbound Message l
Take Action	169	Douglas_DB_Agent	configurable task d...	12/02/2011 02:05 PM		Inbound Message l

Click one of the column links in the Failed Messages grid to view associated information:

Table 2-12 Service Link Failed Messages Clickable Columns

Column	Link
Message Type	Message details on Service Link Message Details popup pages
Req ID	Requisition details
Agent	Agent details in Service Link Agents page

The Messages page, available from the View Transactions tab, allows you to view all messages, both inbound and outbound, regardless of their status; to explicitly filter the messages that appear on the page; and to search messages which fit specified search criteria.

Viewing Messages

The Messages page displays all or selected Service Link messages, depending on which filters have been set. To display the Messages page, click the **View Transactions** tab from the Service Link home page. Then click the **Messages** subtab. The View Failed Messages link in the Common Tasks area of the Service Link home page also displays the Messages page, with a filter set to show only messages with a status of “Failed”.

The Messages page appears, as shown below.

Direction	Message Type	Status	Status Text	Date	Req ID	Agent Name	Task Subject	Resent on
Inbound	SIM Import	Completed	Inbound Messag...	03/19/2012 11:19...		SIImportAgent	No task - SIM Imp...	
Inbound	SIM Import	Failed	Internal applicatio...	03/19/2012 11:17...		SIImportAgent	No task - SIM Imp...	
Inbound	SIM Import	Completed	Inbound Messag...	03/17/2012 06:06...		SIImportAgent	No task - SIM Imp...	
Outbound	Execute Task	Completed	Message sent	03/16/2012 12:02...	378	DummyAgentMin...	Dummy adapter -...	
Outbound	Execute Task	Waiting	Agent Stopped.M...	03/16/2012 12:02...	378	DummyAgentMed...	Dummy adapter -...	
Outbound	Execute Task	Waiting	Agent Stopped.M...	03/16/2012 12:02...	378	DummyAgentMed...	Dummy adapter -...	
Outbound	Execute Task	Completed	Message sent	03/16/2012 12:02...	378	DummyAgentLar...	Dummy adapter -...	
Outbound	Execute Task	Completed	Message sent	03/15/2012 06:36...	377	DummyAgentMed...	Dummy adapter -...	
Outbound	Execute Task	Completed	Message sent	03/15/2012 06:36...	377	DummyAgentMin...	Dummy adapter -...	
Outbound	Execute Task	Completed	Message sent	03/15/2012 06:36...	377	DummyAgentMed...	Dummy adapter -...	
Outbound	Execute Task	Completed	Message sent	03/15/2012 06:36...	377	DummyAgentLar...	Dummy adapter -...	
Outbound	Execute Task	Completed	Message sent	03/15/2012 05:36...	376	RAPIAddComment	RAPI Add comme...	
Outbound	Send Parameters	Failed	Inbound Messag...	03/15/2012 05:36...	376	RAPIAddComment	RAPI Add comme...	
Outbound	Execute Task	Completed	Message sent	03/15/2012 03:51...	131	SI Task Agent	SI creation from ...	
Outbound	Execute Task	Completed	Message sent	03/15/2012 03:51...	131	SI Task Agent	SI creation from ...	
Outbound	Execute Task	Completed	Message sent	03/15/2012 03:50...	131	SI Task Agent	SI creation from ...	
Inbound	Composite	Failed	Inbound Messag...	03/15/2012 02:47...	356	SI Task Agent	SI creation from ...	
Inbound	Composite	Processing	Service Item Mes...	03/15/2012 02:47...	356	SI Task Agent	SI creation from ...	
Inbound	Add Comment	Completed	Inbound Messag...	03/15/2012 02:47...	356	SI Task Agent	SI creation from ...	
Inbound	Add Comment	Completed	Inbound Messag...	03/15/2012 02:45...	361	SI Task Agent	SI inbound actions	

Table 2-13 Service Link Messages Clickable Columns

Column	Link
Message Type	Message details on the Service Link Message Details popup pages.
Status Text	For failed messages, a link is available to the error messages written to the adapter-specific log file and the server log. See the “Service Link Troubleshooting and Administration” section on page 2-66 for more details.
Req ID	Requisition details.
Agent	Agent details in the Service Link Agents page.
Task Subject	Task details in Service Manager.

Message Details

The Message Details popup pages allows you to view both the Service Portal and external messages. This page also displays the channel Id, which uniquely identifies the task in this requisition. You can use this Id when working out issues with the third-party system.

The screenshot shows a 'Message Details' popup window with three tabs: 'Message Details', 'nsXML Message', and 'External Message'. The 'Message Details' tab is active, displaying a table of message attributes. At the bottom right of the window are buttons for 'Save', 'Validate', and 'Close'.

Column	Value
ID	564
Name	take-action
Channel ID	a60f8dba-cd14-43d6-ace3-25eb337aae95
Message State	Completed
Message Type	Take Action
Date Created	12/06/2011 10:23 AM
Last Updated On	12/06/2011 10:23 AM
Last Updated By	
Last Resent On	
Last Resent By	
Agent	Douglas_DB_Agent
Tries	1
Requisition ID	198
Direction	Inbound
Task Subject	DT2
Status	Inbound Message Completed

Click one of the tabs on the Message Details popup page to view associated information.

Table 2-14 Service Link Message Details Subtabs

Column	Link
Message Details	Details about the message.
nsXML Message	For outbound messages, the message produced by the Business Engine, to be processed (transformed) by the Service Link agent; for inbound messages, the message received from the external system, transformed by the agent transformation (if any), and to be processed by the Business Engine.
External System Message	For outbound messages, the message after the transformation associated with the agent has been applied; for inbound messages, the message as it was received from the external system.

Filter and Search

You can use the search functionality to view a subset of messages, for example, all messages with a **Failed** status. Search allows you to specify one of the columns in the Messages window as the search target and to select or type a value to be matched.

Click **Filter and Search** (at the top of the Messages page).

The Filter and Search dialog box also allows you to:

- Filter a particular column by using any relational operator appropriate for the semantics of that column. For example, a date range may be chosen, or any status not equivalent to the specified status can be chosen.
- Filter by the logical ‘AND’ of all criteria specified for columns.

The Filter and Search dialog box is nonmodal. You can fill out the desired criteria and click **Apply** to view the results of the current settings. If required, simply adjust the settings and **Apply** again. Remember that you can also display the messages in ascending or descending order by any column, or change the columns that are displayed by using the techniques explained in the “[Managing the Service Link Screens](#)” section on page 2-7.

Resending Failed Messages

During Service Link development, you may generate many messages that fail to be delivered because of errors in the agent or transformation configuration. These messages should not be resent. Similarly, messages generated via a Service Item Import task should not be resent—the import file format should be adjusted, and the import task tried again.

In a production environment, however, messages may fail to be delivered because of an outage of the external system or other external factor that can be corrected. Once the cause of the delivery failure has been corrected, failed messages can be resent.

To resend failed messages:

-
- Step 1** In the Messages page of the View Transactions tab, click the row containing the Failed message or messages.
- Step 2** In the bottom left corner of the Messages page, click **Resend Message**.
-

Service Link will attempt to resend the message to its designated destination. If the resend succeeds, the message status and date are updated, and the resend date is recorded and displayed in the Resent On column.

Transformations are not reapplied while resending a message. The agent tries to send the already transformed message to its destination.

Resending of failed inbound messages for service item operations is not supported. The process attempts to retry task actions. Hence the destination for those messages is the Business Engine, not the Service Item import processor.

Viewing External Tasks

To view External Tasks:

- Step 1** From the Service Link home page, click **View Transactions**. Then click the **External Tasks** subtab. The External Tasks page appears, as shown below.

Messages		External Tasks					Filter and Search
Task Subject	Started On	Req ID	Status	Completed On	Agent Name	Service	
DT2	12/06/2011 10:22 AM	198	Completed	12/06/2011 10:23 AM	Douglas_DB_Agent	configurable task db	
DT1	12/06/2011 10:22 AM	198	Completed	12/06/2011 10:22 AM	Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/06/2011 10:22 AM	198	Skipped	12/06/2011 10:22 AM	Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/06/2011 10:22 AM	198	Skipped	12/06/2011 10:22 AM	Douglas_DB_Agent	configurable task db	
DT2	12/06/2011 10:21 AM	197	Completed	12/06/2011 10:22 AM	Douglas_DB_Agent	configurable task db	
DT1	12/06/2011 10:21 AM	197	Completed	12/06/2011 10:21 AM	Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/06/2011 10:21 AM	197	Skipped	12/06/2011 10:21 AM	Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/06/2011 10:21 AM	197	Skipped	12/06/2011 10:21 AM	Douglas_DB_Agent	configurable task db	
DT2	12/06/2011 10:20 AM	196	Completed	12/06/2011 10:21 AM	Douglas_DB_Agent	configurable task db	
DT1	12/06/2011 10:19 AM	196	Completed	12/06/2011 10:20 AM	Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/06/2011 10:19 AM	196	Skipped	12/06/2011 10:19 AM	Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/06/2011 10:19 AM	196	Skipped	12/06/2011 10:19 AM	Douglas_DB_Agent	configurable task db	
DT2	12/06/2011 10:19 AM	195	Completed	12/06/2011 10:19 AM	Douglas_DB_Agent	configurable task db	
DT1	12/06/2011 10:18 AM	195	Completed	12/06/2011 10:19 AM	Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/06/2011 10:18 AM	195	Skipped	12/06/2011 10:18 AM	Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/06/2011 10:18 AM	195	Skipped	12/06/2011 10:18 AM	Douglas_DB_Agent	configurable task db	
DT1	12/05/2011 05:08 PM	194	Ongoing		Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/05/2011 05:08 PM	194	Skipped	12/05/2011 05:08 PM	Douglas_DB_Agent	configurable task db	
configurable task db needs...	12/05/2011 05:08 PM	194	Skipped	12/05/2011 05:08 PM	Douglas_DB_Agent	configurable task db	
DT1	12/05/2011 05:07 PM	193	Ongoing		Douglas_DB_Agent	configurable task db	

Send Manual Message Page 1 of 30 Displaying 1 - 20 of 593

- Step 2** Click one of the following column links to view associated information.

Table 2-15 Service Link External Tasks Clickable Columns

Column	Link
Task Subject	Task details in Service Manager
Req ID	Requisition overview in My Services
Agent Name	Agent details in the Service Link Agents page

Filter and Search

Like the Messages display, the External Tasks page offers the ability to customize the columns and order of data displayed in the data table and to filter and search on that data.

Sending a Manual Message

A task that has been started and is expecting to receive an inbound message is in an “Ongoing” state. The incoming message will typically update the task or change its status. No subsequent tasks in the requisition's delivery plan can be performed until a message is received and the task is completed. If you suspect (or can confirm by conferring with administrators of the external system) that the expected message has already been sent, but has somehow been “lost”, you can emulate receipt of the message by sending a manual message.

Manual messages cannot be used to emulate failed service item operations.

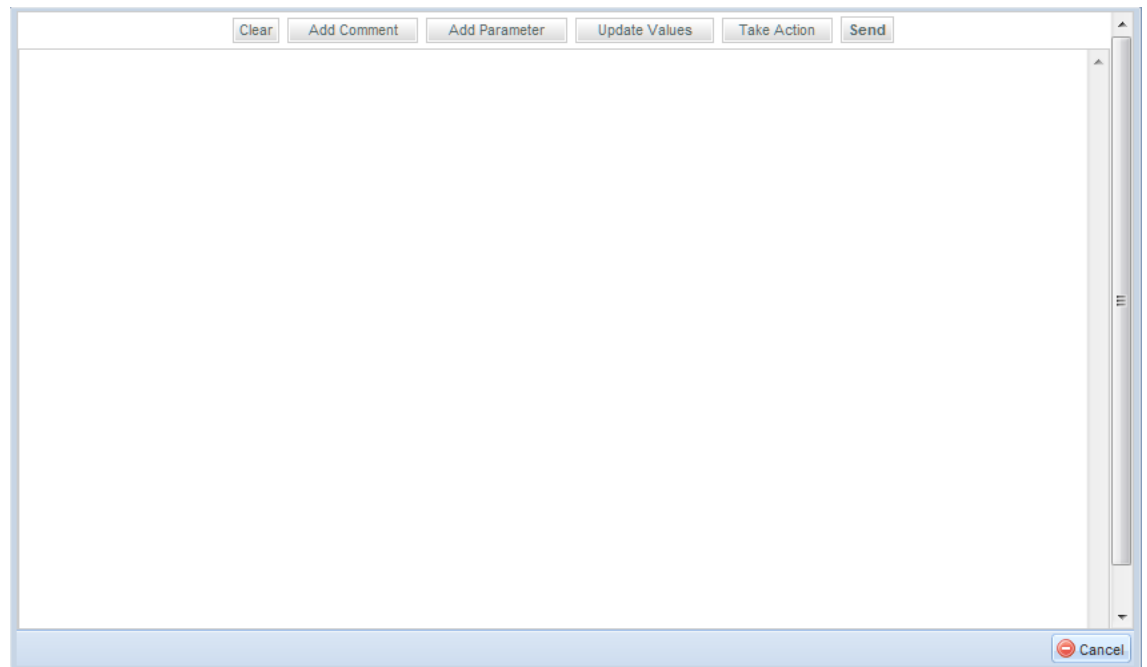


Note

Use this feature carefully. This feature overrides all the communication protocols in the system, and using it may leave artifacts in the third-party system to which Service Link may no longer be able to respond. Also, if you use this feature to cancel a requisition, for example, Service Link will not notify the interested parties, so you will have to follow up on your own.

To send a manual message to the Business Engine:

-
- Step 1** From the Service Link Home page, click **View Transactions**. Then click **External Tasks**.
The External Tasks page appears.
 - Step 2** In the bottom left corner of the External Tasks page, click the line containing the task for which you want to send a manual message.
 - Step 3** Click **Send Manual Message**.
The Send Manual Message dialog box appears, as shown below.



- Step 4** Click the button corresponding to the type of message you want to send—**Add Comment**, **Add Parameter**, **Update Values** or **Take Action**.

Action	Description
Add Comment	Send an add-comments message, to add a system comment to the requisition.
Add Parameter	Send a send-parameters message. Modify one or more inbound agent parameter values and the value of the corresponding dictionary field to which the parameter is bound.
Update Values	Send a message to modify the contents of the specified dictionary field. (This message type is provided primarily for backward compatibility with previous versions; field contents should typically be updated via inbound agent parameters.)
Take Action	Send a take-action message. Mark the task as done (completed) or canceled; approve or reject an authorization; or mark a review as OK.

- Step 5** Respond to the associated popup dialog boxes (turn off your popup blocker) for the message type chosen. This will populate the message window with a well-formed XML message of the appropriate type. An <add-comments> message will also be included, to indicate that this message was not received through normal channels, but manually generated.

- Step 6** If desired, you may edit the generated message. When you have constructed the entire message, click **Send**. An inbound message is sent to the Business Engine.

Republishing Service Link Messages

In the rare occasion of extended outage or incorrect configurations of the Service Link application, external tasks might not have corresponding outbound messages created in Service Link.

Once the underlying issue is resolved in Service Link and the application is up and running again, the problem external tasks can be republished to Service Link to allow the outbound messages to be created and the delivery process to resume.

To republish outbound messages:

-
- Step 1** From the Service Link Home page, click **View Transactions**. Then click **Message Republish**.
 - Step 2** On the left-hand pane, enter the Requisition ID for the requests which have one or more missing outbound Service Link messages. All authorization and delivery tasks associated with the requisition are evaluated and only those tasks that require republishing are processed for outbound message creation. Up to 20 requisitions can be entered at a time.
 - Step 3** Click **Republish**.
 - Step 4** Review the processing status on the right-hand pane once the republish process is completed.
-

Service Link Adapters

All Service Link adapters support nsXML as the data exchange format. For more information about the nsXML format, see [Chapter 3, “Service Link Adapter Development Kit”](#).

All poller-based adapters support processing on only one message per invocation.

The Service Link Adapters installed in all application instances are:

- [Dummy Adapter](#)
- [Database Adapter](#)
- [File Adapter](#)
- [HTTP/WS Adapter](#)
- [JMS Adapter](#)
- [MQ Adapter](#)
- [Service Item Listener Adapter](#)
- [VMware Adapter](#)
- [Web Service Listener Adapter](#)

In addition to these adapters, Service Link supports an [Auto-Complete Adapter](#).

Additional adapters may be installed and configured using the Service Link Adapter Development Kit (ADK). Any such custom adapters also appear on the Adapters page, and their properties may be reviewed. For details on building and installing custom adapters, see [Chapter 3, “Service Link Adapter Development Kit”](#).

The following sections describe these adapters.

Auto-Complete Adapter

The Auto-Complete adapter allows an agent to send an outbound message and to mark the task as complete without waiting to receive an acknowledgement from the external system. If the outbound message is successfully sent (for example, a file is written to the specified directory by an outbound file adapter), the auto-complete adapter generates an incoming message for the same task. That incoming message has the message type “take-action”. This message is processed normally by the Business Engine, marking the action as done and completing the external task.

Dummy Adapter

The Dummy Adapter is a placeholder. It can be used in several processing scenarios:

- Using the dummy adapter as the inbound adapter allows an external task initiated by Service Link to remain in Ongoing status.
- Using the dummy adapter as an outbound adapter and the auto-complete adapter as the inbound adapter allows service designers to implement Auto-Complete Agents in external tasks. The task can then be used in part of the workflow, for example, to generate an email to participants, or to close a request which has no other tasks. This combination can also be used to verify if communications between Request Center and Service Link are working correctly.

Database Adapter

The Database (DB) adapter uses one or more tables in a database to pass data between Service Portal and external applications.

Database Connection

Inbound and outbound database adapters are capable of communicating with any JDBC-compliant relational database that supports ANSI-standard SQL. Valid connection criteria must be provided, as well as the JDBC URL, and a database driver. If the external database is SQLServer or Oracle, Cisco-provided drivers may be used. Drivers available from Cisco are:

```
com.newscale.jdbc.sqlserver.SqlServerDriver
com.newscale.jdbc.oracle.OracleDriver
com.newscale.jdbc.UnifiedDriver
```

The unified driver supports all Service Portal supported databases (SQLServer and Oracle) and should be used in preference to database-specific drivers. The database-specific drivers are supported to provide backward compatibility with previous versions of Service Portal.

The JDBC URL has the format:

```
jdbc:newscale:dbtype://machine:port;DatabaseName=dbname
jdbc:newscale:dbtype://machine:port;SID=sid
```

where

- dbtype is sqlserver or oracle
- machine is the name of the database server
- port is the port through which to connect to the database; typically 1433 for SQLServer and 1521 for Oracle

- The database name must be specified for SQLServer; the SID (System Identifier) must be specified for Oracle

EXAMPLES:

```
jdbc:newscale:sqlserver://localhost:1433;DatabaseName=RCDev
```

```
jdbc:newscale:oracle://PRODSRVR2:1521;SID=RCProd
```

A user-supplied driver may be used if supporting jar files are installed on the directory ISEE.war/WEB-INF/lib in the Request Center directory structure.

-
- Step 1** Obtain the appropriate third-party JDBC driver. For example, the Sybase JDBC Driver can be downloaded from Sybase's website. Oracle drivers are available online or as part of the Oracle distribution.
- Step 2** Copy any required jars to the ISEE.war/WEB-INF/lib folder.
- Step 3** Modify the Agent settings to use the custom driver and the correct JDBC URL format. For example, the format for the JDBC URL for the Sybase driver is:
- ```
jdbc:sybase:Tds:host:port/database
```
- The format for the Oracle thin driver would be
- ```
jdbc:oracle:thin:@host:port
```
- Step 4** Restart the Service Link and Request Center services.
-

The format of the JDBCUrl may also be influenced by the application server on which Service Link is deployed. For example, a possible JDBC URL to establish a connection to SQLServer database from a WebSphere application server would be:

```
jdbc:newscale:sqlserver://hostname;DatabaseName=databasename;selectMethod=direct;alwaysReportTriggerResults=true;insensitiveResultSetBufferSize=16384;useServerSideUpdatableCursors=false;maxPooledStatements=0
```

Inbound Properties

When the database adapter is used as an inbound adapter, the agent properties include a SQL statement to be executed against the specified database connection. The SQL is typically a select command which returns a set of rows. These rows are then formatted into an external XML message. The message must be transformed via an inbound transformation (specified in the agent) into a valid nsXML inbound message. That message is, in turn, passed to the Business Engine. If the Business Engine finds an open task identified by the Channel ID specified in the inbound message, the inbound message is processed and the specified action taken.

The Property sheet for the database inbound adapter prefixes the property names given below with “DBInboundAdapter”.

Table 2-16 DB Adapter Inbound Properties

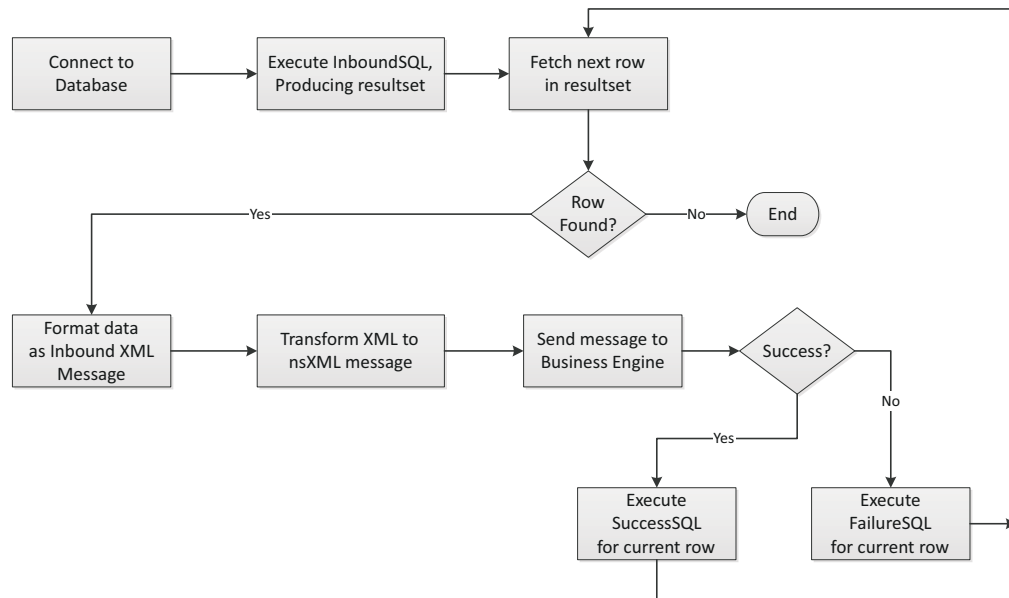
Property	Description
DBPassword	Password for the specified user.
DBUserName	Database user name.

Table 2-16 DB Adapter Inbound Properties (continued)

Property	Description
InboundSql	The SQL statement to be executed for the inbound transaction. This should be a SELECT statement that returns a set of rows. Transactional SQL (that is, a procedure) is not supported.
InboundSuccessSql	The SQL to be executed on success of the inbound transaction, typically a SQL update or delete statement which marks the current row as successfully processed.
InboundFailureSql	The SQL to be executed on failure of the inbound transaction typically a SQL update or delete statement which marks the current row as successfully processed.
JDBCUrl	JDBC URL to connect to the database.
JDBCDriverClass	The class name of the driver to be used to connect to the database.

Inbound Message and Work Flow

The process flow for the inbound database adapter is shown below:



For each row in the result set, the adapter generates an XML message with the following structure:

- The root element of the message is <inbound-results>.
- The required child element is <row>. Each message has exactly one <row> element.
- Each <row> element has multiple <column> elements, one for each column included in the InboundSQL statement specified for the adapter.
- The <row> element has attributes for the column name (<name>) and JDBC data type (<type>; 12 for character and 1 for numeric).
- The value of each <column> element is the value returned for the corresponding column in the SQL statement.

For example, the SQL statement

```
SELECT channelId, task, status, processType FROM rcInterface
WHERE status = 'UPDATED'
```

might yield an XML stream like the following:

```
<?xml version='1.0' encoding='UTF-8'?>
<inbound-results>
  <row>
    <column name="channelid" type="12" >
      "3F2504E0-4F89-11D3-9A0C-0305E82C3301"
    </column>
    <column name="task" type="12" >Task</column>
    <column name="status" type="12" >UPDATED</column>
    <column name="processtype" type="1" >null</column>
  </row>
</inbound-results>
```

A transformation must then be applied to this XML stream to produce a valid nsXML inbound message. For example, a transformation which would complete an ongoing task might include the following code:

```
<xsl:template match="/inbound-results/row">
  <xsl:variable name="status" select="column[@name='status']" />
  <xsl:choose>http://www.w3schools.com/xsl
    <xsl:when test="$status='Complete'">http://www.w3schools.com/xsl
      <message>http://training2.cisco.com/RequestCenter
        <xsl:attribute name="channel-id">
          <xsl:value-of select="column[@name='channelId']" />
        </xsl:attribute>http://training2.cisco.com/ServiceLink
        <take-action action="done" />
      </message>
    <xsl:otherwise>
```

The Business Engine processes the resultant nsXML message. If the message was applied successfully, the SuccessSQL specified in the agent is executed. The SuccessSQL typically updates the columns in the source table that caused the row to be selected for processing, so that the row will not be found again in the next polling interval. To specify that Service Link should update the current row, identify the column or columns that comprise the row's unique identifier. Those columns must have been included in the inbound SQL statement. For example:

```
UPDATE rcInterface
  SET status = 'DONE'
  WHERE channelId = #ChannelId#
```

Similarly, the FailureSQL is executed if the Business Engine failed to apply the nsXML message—for example, if an error occurred during processing of the message. The FailureSQL typically updates the status of the current row to indicate that the row was not correctly processed. For example:

```
UPDATE rcInterface
  SET status = 'FAILED'
  WHERE channelId = #ChannelId#
```

Outbound Properties

When the database adapter is used as an outbound adapter, it provides a “staging table” style interface between Service Portal and the external system. The nsXML outbound message which is provided to the agent by the Business Engine must be transformed into an external message containing one or more SQL statements. These SQL statements are then executed in the specified database, using the specified connection.

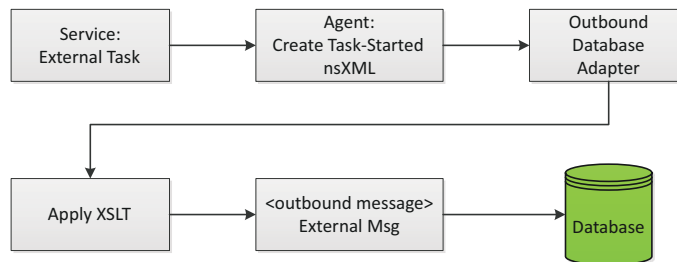
The Property sheet for the DB outbound adapter prefixes the property names given below with “DBOutboundAdapter”.

Table 2-17 DB Adapter Outbound Properties

Property	Description
DBPassword	Database User password
DBUserName	Database user name
JDBCUrl	JDBC URL to connect to the database
JDBCDriverClass	The class name of the specific driver to be used to connect to the database

Outbound Message and Workflow

The process flow for the outbound database adapter is shown below:



The outbound message produced by the XSLT transformation must have the format:

```

<?xml version="1.0" encoding="UTF-8"?>
<outbound-message>
  <execute-sql-list>
    <execute-sql> SQLStatement
  </execute-sql>
</execute-sql-list>
</outbound-message>
  
```

The message can contain multiple SQL statements, each within an <execute-sql> tag. These statements typically insert or update rows in SQL tables. Any SQL statement supported by the JDBC driver specified for the adapter can be used. Stored procedures (in SQLServer Transact-SQL or Oracle PL/SQL) are not supported, although the SQL statement can include user-defined functions. Since each external task is uniquely identified by a Channel ID, the target table for the outbound SQL statement must include a column for the Channel ID in order for that task to be updateable by an inbound message.

File Adapter

The File Adapter provides support for reading files from a specified directory or writing files to a specified directory.

- The adapter cannot be configured for processing files from multiple directories or sub directories of a specified directory.
- The oldest file of a set of files of a directory would be processed by an inbound file adapter when invoked.
- Only one agent should be configured for a specified directory to process the files.
- The directories (locations) specified must be on the file system of the application server where Request Center is installed or accessible from the application server. All directories must be on the same physical device, since files are moved from one directory to another as Service Link processing proceeds.

File Adapter Inbound Properties

Following are the properties with the default values for the File Adapter.

The Property sheet for the File inbound adapter prefixes the property names given below with “FileInboundAdapter”.

Table 2-18 File Adapter Inbound Properties

Property	Description
BackupLocation	Location where the files are backed up after they have been processed, if the Final Resolution or OnError property is “Preserve”.
BackupSuffix	File extension for the backup files; default is .bak.
FileLocation	Location (directory) that is polled for inbound files to be read; a unique location should be used for each inbound file adapter.
FileNameDateFormat	Date format for the files; default is .yyyyMMddHHmmssSSS.
FinalResolution	Action to take on the file after transaction completion. Options are: <ul style="list-style-type: none"> • Preserve – Moves the file to the backup location • Delete – Deletes the file default is Preserve.
OnError	Action to take on the file when an error occurs. Options are: <ul style="list-style-type: none"> • Preserve – Moves the file to the backup location • Delete – Deletes the file default is Preserve.
TempLocation	Temporary folder used for processing inbound files.

File Adapter Outbound Properties

The outbound file adapter produces an XML file on the specified file location. The name of the file contains the channel-id, a unique identifier for the external task that included the agent and created the message. The file name ends with the date format specified as an outbound property.

The Property sheet for the File outbound adapter prefixes the property names given below with “FileOutboundAdapter”.

Table 2-19 File Adapter Outbound Properties

Property	Description
BackupLocation	Outbound files backup location; may be any valid file system directory accessible from the application server.
BackupSuffix	File extension for the backup files; default is .bak.
ConflictResolution	Action to take in case of conflict in the outbound file names. Options are: <ul style="list-style-type: none"> • Preserve – Moves the file to the backup location • Delete – Deletes the file Default is Rename.
FileLocation	Location to which the file is written; may be any directory accessible from the application server.
FileNameDateFormat	Date format for the file name; default is: .yyyyMMddHHmmssSSS.
OnError	Action to take on the file when an error occurs. Options are: <ul style="list-style-type: none"> • Preserve – Moves the file to the backup location • Delete – Deletes the file Default is Preserve.
TempLocation	Temporary folder used for processing the outbound file.

HTTP/WS Adapter

The HTTP/WS adapter is used to send or receive HTTP requests or web service requests and responses. HTTPS is also supported.

The use of a proxy server in connecting to the web service is not supported.

When used to call web services, only synchronous calls are possible. The outbound transformation must be configured such that a SOAP wrapper is placed around the message.

Outbound Properties

The HTTP/WS Adapter outbound properties specify the behavior of the outbound adapter.

The Property sheet for the http/ws outbound adapter prefixes the property names given below with “HttpOutboundAdapter”.

Table 2-20 HTTP/WS Adapter Outbound Properties

Property	Description
WsdIURL	The URL of the wsdl that includes the operation to be performed; used only with the Integration Wizard.
WsdIOperation	The operation to be performed by the web service; documentation only except when using the Integration Wizard. A drop-down list of all operations included in the specified WSDL is available.

Table 2-20 HTTP/WS Adapter Outbound Properties (continued)

Property	Description
RoutingURL	URL to route all outbound messages to be posted; the web service end point.
AcceptUntrustedURL	Option to allow accepting untrusted certificates from external systems; default is true.
ContentType	Content type; default is text/xml; charset=ISO- 8859-1.
TimeOut	Timeout to get the http url connection; default is 180,000 microseconds.
ProcessResponse	Option to treat the result of the post or the response to a SOAP message as an inbound message; default is false.
RequestHeaders	Any custom header parameters that must be included in the HTTP request header, typically in the form of name-value pairs, with parameters separated by ampersands (&). For example, a SOAPAction might be entered in the format: SOAPAction=OpCreate A SOAPAction and a custom header called “referrer” might be entered in the format: SOAPAction=OpCreate&referrer=www.test.com
AuthenticationSchema	The type of authentication to be used to request the web service or post to the URL; options are basic, anonymous, digest, or NTLM; details explanations of these options are given below.
AuthenticationScopeHost	The host to which the authentication credentials apply. May be left empty if credentials are applicable to any host.
AuthenticationScopePort	The port to which the authentication credentials apply. May be left empty if credentials are applicable to any port.
AuthenticationScopeRealm	The realm to which the authentication credentials apply. May be left empty if credentials are applicable to any realm.
Username	User name for authentication to the target system.
Password	Password for authentication to the target system.
Host	Host credential that may be required for some authentication schemas (like NTLM).
Domain	Domain credential that may be required for some authentication schemas. NTLM does not use the concept of realms. The authentication domain should be specified as the value of the ‘realm’ attribute. May be left empty if credentials are applicable to any domain.
SaveRefField	Boolean used when <i>ProcessResponse</i> is true. Indicates that the response will contain a field which the external system uses as a unique identifier (or TopicID) for this task; see the “Response to the http/ws Request” section on page 2-52 for more information.
RefFieldXPath	The XPath expression in the response that identifies the reference field (Topic ID).

Table 2-20 HTTP/WS Adapter Outbound Properties (continued)

Property	Description
RefFieldPattern	A regular expression to be applied to be reference field.
CancelIdentifierXPath	The XPath expression whose presence specifies that an ongoing task should be canceled.

Authentication Schemes

Some properties of the outbound http/ws adapter are required only for certain authentication schemes and, then, perhaps only for web servers with customized authentication. [Table 2-21](#) below summarizes authentication schemes supported by the outbound http/ws adapter.

Table 2-21 Authentication Schemes

Authentication Type	Description
Anonymous	The request is not required to supply user credentials; access to the web server is typically via a service account.
Basic	User name and password are required; password is sent in clear text.
Digest	User name and password are required, but password is transmitted as an MD5 hash.
NTLM	Integrated Windows Authentication on Windows 2003.
NTLMv2	Integrated Windows Authentication on Windows 2008.

Response to the http/ws Request

When a request is posted to a web site or a message sent to a web service, the target site typically sends a response to the message originator. If that response is unlikely to contain information useful to Service Link, you may set the *Process Response* property to false, to instruct Service Link to ignore any such messages. However, such responses might include additional information, such as the external system's ticket number or case number assigned to the task that originated in Request Center. In this case, you can set both the Process Response and Save Ref Field properties to true and specify the xpath for the Reference field for Service Link to capture the reference from the web service response. In addition, a transformation can be applied to the response to invoke actions to update the service form with information from the external system.

Reference Field and TopicID

External systems generally have their own means for identifying incidents, requests, or other objects, whether opened by a third-party system or maintained via the product's user interface. A designated Reference Field (TopicID) allows Request Center to maintain a cross-reference between the external system's unique identifier and the Request Center Channel-Id. Once the TopicID is identified in the initial response to the web service request and saved, further messages from the external system, received via the web services listener adapter, can use the TopicID to identify the Request Center external task.

Inbound Properties

- The HTTP/WS inbound adapter is a listener adapter and does not support polling based invocations.
- Only one HTTP/WS inbound agent should be configured for a given URL. Either the http or https protocol may be used.

No properties may be specified for an inbound http adapter. All http posts should be directed to the Integration Server's URL:

```
<ServerName>:<Port>/IntegrationServer/ishttplistener?channel-id=<channel-id>
```

where

- <ServerName> is the Service Portal application server.
- <Port> is the port on which Service Portal is listening.
- <channel-id> is the channel ID which uniquely identifies the task to be affected by the inbound message. Error 503 (Application Error) is returned to the third-party system if the channelID does not apply to an ongoing task.

Web Service Invocation

A web service is not-so-simply "XML over HTTP". For an outbound adapter, an XML message is sent via http (or https) to a web service. The message, created by application of a transformation to the outbound message, must be enclosed within a SOAP envelope. A sample XML message to a web service might look like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
  <soap:Header
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <AuthenticationInfo>
      <userName>ns28sbd</userName>
      <password>09rbc19</password>
    </AuthenticationInfo>
  </soap:Header>
  <soap:Body
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <OpCreate>
      <Assigned_To_Group>CSCC</Assigned_To_Group>
      <Case_Type>Problem</Case_Type>
      <Category>Computer/Printer/Server</Category>

      <!-- additional tags as required -->

      <txt_internalticketid/>
      <txt_requestid >40</txt_requestid >
      <Type>New Hardware Request</Type>
    </OpCreate>
  </soap:Body>
</soap:Envelope>
```

JMS Adapter

The JMS inbound adapter is a listener adapter and does not support polling based invocations.

The JMS adapter can read and write messages from a queue or publish/subscribe to a particular topic. Only one JMS inbound agent should be configured for a given queue. It is not possible to use the same agent to subscribe to multiple topics. The topic must be fully specified; for example, “topic.sample.exported”.

Inbound Adapter Properties

The Property sheet for the JMS inbound adapter prefixes the property names given below with “JMSInboundAdapter”.

Table 2-22 JMS Adapter Inbound Properties

Name	Description
JndiProviderUrl	JNDI provider URL for looking up JMS administered objects for the inbound agent; default is jnp://localhost:4099.
JndiFactory	JNDI Naming factory for inbound agent; default is org.jnp.interfaces.NamingContextFactory.
JmsTopicFactory	Topic Connection factory for getting JMS Topic Connection for inbound agent; not used.
JmsQueueFactory	Queue Connection factory for getting JMS Queue Connection for inbound agent; default is ConnectionFactory.
MessageMode	Whether JMS destination is Queue or Topic. Valid values are Queue or Topic; default is Queue.
JmsQueue	Named JMS queue if message mode is Queue for inbound agent.
JmsTopic	Named JMS topic if message mode is Topic for inbound agent.
MessageType	Type of the message for the inbound agent. Valid value is Text
Publisher.isAdapter	If the publisher is adapter; default is True.
Listener.UseCallback	Whether to use callbacks; default is True.
UserName	User name for JNDI Security credentials for the inbound agent.
Password	Password for JNDI Security credentials for the inbound agent.

Outbound Adapter Properties

The Property sheet for the JMS outbound adapter prefixes the property names given below with “JMSOutboundAdapter”.

Table 2-23 JMS Adapter Outbound Properties

Name	Description
JndiProviderUrl	JNDI provider URL for looking up JMS administered objects for the outbound agent; default is jnp://localhost:4099.
JndiFactory	JNDI Naming factory for the outbound agent; default is org.jnp.interfaces.NamingContextFactory.

Table 2-23 JMS Adapter Outbound Properties (continued)

Name	Description
JmsTopicFactory	Topic Connection factory for getting JMS Topic Connection for outbound agent; not used.
JmsQueueFactory	Queue Connection factory for getting JMS Queue Connection for outbound agent; default is ConnectionFactory.
MessageMode	Whether JMS destination is Queue or Topic. Valid values are Queue Topic.
JmsQueue	Named JMS queue if message mode is Queue for outbound agent.
JmsTopic	Named JMS topic if message mode is Topic for outbound agent.
MessageType	Type of the Message for the outbound agent. Valid value is Text.
Publisher.isAdapter	If the publisher is adapter; default is True.
UserName	User name for JNDI Security credentials for outbound agent.
Password	Password for JNDI Security credentials for outbound agent.

MQ Adapter

The MQ inbound adapter is a poller adapter which uses the IBM WebSphere Message Queue (MQ) system. The adapter supports IBM MQ Series versions 5.x and above. It uses IBM MQ Series Java API for the integration. IBM MQ software is not included with Service Portal, and a license must be obtained from IBM.

Inbound Properties.

The Property sheet for the MQ inbound adapter prefixes the property names given below with “MQInboundAdapter”.

Table 2-24 IBM MQ Inbound Adapter Properties

Name	Description
ManagerName	Name of the IBM MQ Manager
HostName	Host name of the IBM MQ Server
Port	Port for the IBM MQ Server for Inbound
UserName	User Name for authentication
Password	Password for authentication
ChannelName	IBM MQ Channel Name for inbound messages
QueueName	Queue Name for inbound messages
MsgFormat	Message Format for inbound messages; default is Text

Outbound Properties

The Property sheet for the MQ outbound adapter prefixes the property names given below with “MQOutboundAdapter”.

Table 2-25 IBM MQ Adapter Properties

Name	Description
ManagerName	Name of the IBM MQ Manager for outbound messages
HostName	Host name of the IBM MQ Server
Port	Port for the IBM MQ Server
UserName	User Name for authentication
Password	Password for authentication
ChannelName	IBM MQ Channel Name for outbound messages
QueueName	Queue Name for outbound messages
MsgFormat	Message Format for Outbound; default value is Text

Service Item Listener Adapter

Similar to the Web Service Listener Adapter (see the [“Web Service Listener Adapter”](#) section on page 2-58), the Service Item Listener Adapter provides a Web service (SOAP) end point to be used by external systems to send updates to external tasks. In addition to task updates, the adapter allows the creation, update, and deletion of service items in Lifecycle Center as part of the inbound SOAP message. The adapter also allows the retrieval of service item metadata and the data for service item instances.

The SOAP message sent by an external system must invoke the “processMessage” operation. The message content within the soap body is transformed into a message that Service Link understands, then segregated based on the operation type, and forwarded to the Business Engine and Service Item Import processor, respectively. Up to two messages may result in the View Transactions page for an inbound SOAP message—one for task update operations (take-action, add-comments, send-parameters) and one for service item operations (create, update, delete). The latter has “Service Item” as the message type.

Inbound Properties

The Property sheet for the Service Item Listener inbound adapter prefixes the property names given below with “ServiceItemListenerInboundAdapter”.

Table 2-26 Service Item Listener Inbound Adapter Properties

Property	Description
WsdL	<p>The URL of the wsdl that describes the Request Center inbound Service Item for the current installation has the format:</p> <pre><Protocol>://<ServerName>:<Port>/IntegrationServer/webservices/wsdl/ServiceItemTaskService.wsdl</pre> <p>where:</p> <ul style="list-style-type: none"> <Protocol> is either http or https. <ServerName> is the server where Service Link is installed. <Port> is the communication port specified for Service Link. <p>For example,</p> <pre>http://ccp-prod.cisco.com:8089/IntegrationServer/webservices/wsdl/ServiceItemTaskService.wsdl</pre> <p>This property is read-only. It is made available so that designers can consult the WSDL, which is useful in understanding the services and writing a webservice client.</p>
RoutingURL	<p>The URL to which the SOAP message should be sent has the format:</p> <pre><Protocol>://<ServerName>:<Port>/IntegrationServer/services/TaskService</pre> <p>This property is read-only. It is made available so that external systems integrators can write clients that post SOAP messages to this URL.</p>

Outbound Properties.

The Service Item Listener Adapter is unidirectional—inbound only. Therefore, there are no Outbound Properties.

VMware Adapter

A Lifecycle Center license is required to use the VMware adapter. The VMware adapter integrates with vSphere 4.1 vCenter server, using the VMware API.

The use of the VMware adapter to process grid dictionaries is not currently supported.

Outbound Properties

The Property sheet for the VMware outbound adapter prefixes the property names given below with “VMwareOutboundAdapter”.

Table 2-27 VMware Outbound Adapter Properties

Name	Description
VMServerURL	The complete URL of the vCenter server with which the adapter needs to communicate, for example, https://<ServerName>/sdk. See below for more information.
UserName	User name for login in the format <network domain>\<network userid>.

Table 2-27 VMware Outbound Adapter Properties

Name	Description
Password	Password of the user.
TimeOut	Timeout, in seconds.
IgnoreVMServerCertificate	Parameter indicating if invalid server certificates should be ignored. Leave this set to false .
AutoInstallCertificate	Leave this set to true .
ServerCertificate	String containing the X.509 server certificate, if SSL connections should be honored for servers with invalid/self-signed certificates. You can leave this blank when the AutoInstallCertificate option is set to true .

VMware Server URL

Https is enabled in most vCenter installations. Please check with the VMware administrator to ensure you have specified the correct protocol.

You will need a separate agent for each vCenter server. Further, if there are different logins for managing different types of activities or different data centers of the same vCenter server, you will have to define one agent for each login.

VMware Adapter and Agent Architecture

The architecture of the VMware adapter, and agents which use this adapter, differs from the architecture of the other Service Link adapters. The VMware adapter communicates directly with the VMware API. Data required by the operations supported by the API is provided via dictionaries used in the service which includes the VMware integration.

For configuring an agent that uses the VMware adapter:

- The outgoing message content must be set to “small” (the default).
- A user-specified transformation is not used. The nsXML message is always identical to the “external” message. The contents of that message are used to produce a call to the VMware API.
- The integration designer must specify one outbound request parameter for the agent, indicating the name of the dictionary which contains the fields required for the VMware operation. The outbound parameter must be named “Dictionary_Lookup_Name” with its value set to the appropriate dictionary name.
- The VMware adapter is outbound only. The results of the VMware operation performed, and an error message, if applicable, are returned in response to the outbound message.

For details on configuring the VMware adapter, see the *Cisco Service Portal Designer Guide*.

Web Service Listener Adapter

The Web Service Listener Adapter provides a Web service (SOAP) end point to be used by external systems to send updates to external tasks. The SOAP message sent by an external system must invoke the “processMessage” operation. The message content within the soap body is transformed into a message that Service Link understands, then forwarded to the HTTP/WS inbound adapter to be processed further.

The Web service Listener Adapter uses an underlying Web Service Listener.

Inbound Properties.

The Property sheet for the Web Service Listener inbound adapter prefixes the property names given below with “WSListenerInboundAdapter”.

Table 2-28 Web Service Listener Inbound Adapter Properties

Property	Description
WsdIURL	<p>The URL of the wsdl that describes the Request Center inbound Web Service for the current installation has the format:</p> <pre><Protocol>://<ServerName>:<Port>/IntegrationServer/webservices/wsdl/TaskService.wsdl</pre> <p>where:</p> <ul style="list-style-type: none"> <Protocol> is either http or https. <ServerName> is the server where Service Link is installed. <Port> is the communication port specified for Service Link. <p>For example,</p> <pre>http://ccp-prod.cisco.com:8089/IntegrationServer/webservices/wsdl/TaskService.wsdl</pre> <p>This property is read-only. It is made available so that designers can consult the WSDL, which is useful in understanding the services and writing a webservice client.</p>
WsdIRoutingURL	<p>The URL to which the SOAP message should be sent has the format:</p> <pre><Protocol>://<ServerName>:<Port>/IntegrationServer/services/TaskService</pre> <p>This property is read-only. It is made available so that external systems integrators can write clients that post SOAP messages to this URL</p>

Outbound Properties.

The Web Service Listener Adapter is unidirectional—inbound only. Therefore, there are no Outbound Properties.

Integration Wizard

Overview

The Integration Wizard automates many of the steps involved in creating an integration 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 all components required to support the integration.

- The Service Link agent that can be used in an external task to perform the integration is created and referenced in the delivery plan of the current service.
- A transformation to transform nsXML into the SOAP message required by the web service is created and referenced in the Outbound Adapter of the agent.
- Agent parameters for all data required both in the initial web service request and the response are added to the agent definition.
- A dictionary containing fields to hold agent parameter values is created, and dictionary fields are mapped to corresponding agent parameters.
- An active form component containing the dictionary is created and included in the current service.

The Integration Wizard uses some default options in defining the authentication method and behavior of the integration. If these settings are not appropriate, or if the integration must be modified after it has been created, the advanced configuration options available in Service Link Manage Integration pages can be used to edit the agent definition.

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.

WSDL's to be accessed by the Integration Wizard must comply with Web Services Operability (WS-I) best practices.

Using the Integration Wizard

To use the Integration Wizard:

-
- Step 1** Edit the service in Service Designer.
 - Step 2** Go to the **General** subtab of the Plan tab for the service. Optionally fill in other data relating to the task.
 - Step 3** Click **Create Agent**.

The screenshot shows a configuration form for the Integration Wizard. At the top left is a 'Save' button. Below it, the 'Workflow Type' is set to 'Agent Action' in a dropdown menu, followed by a three-dot menu icon and a 'Create Agent' button. The 'Task name' field contains the text 'Task1 - Tasks for Queues'. Below that, the 'Subtasks execute' dropdown is set to 'one after the other (sequentially)', and the 'Priority' dropdown is set to 'Normal'.

The first page of the Integration Wizard appears. The wizard may consist of up to eight pages, depending on how the agent is configured. As each page is completed, click **Next** to advance to the next page, or **Previous** to return to a previous page. When you are finished, click **Save** to save the definition of the agent (and other design components) or **Cancel** to exit without saving your work.

General Information

Start by specifying general information about the agent:

Create Agent

General Information

Name: AgentName

Action: Agent Action

Outgoing Content: Data and Parameters; No Service Details (default; small)

Failed Email: None

Description:

Previous | **Next** | Save | Cancel

The dictionary and active form component to be created will have the same name as the agent. Therefore, since naming standards for dictionaries are more stringent than for agents, the agent name can contain only letters, numbers and the underscore, and cannot start with a number.

All other settings on this screen match those available in the Agents page of Service Link.

Click **Next** to proceed to the next page of the wizard.

Outbound Properties

The screenshot shows a 'Create Agent' dialog box with the following fields and values:

- WSDL URL:
- Operation:
- Routing URL:
- Advanced Properties** (expanded):
 - User Name:
 - Password:
 - Accept Untrusted URL:
 - Content Type:

At the bottom of the dialog are buttons for 'Previous', 'Next', 'Save', and 'Cancel'.

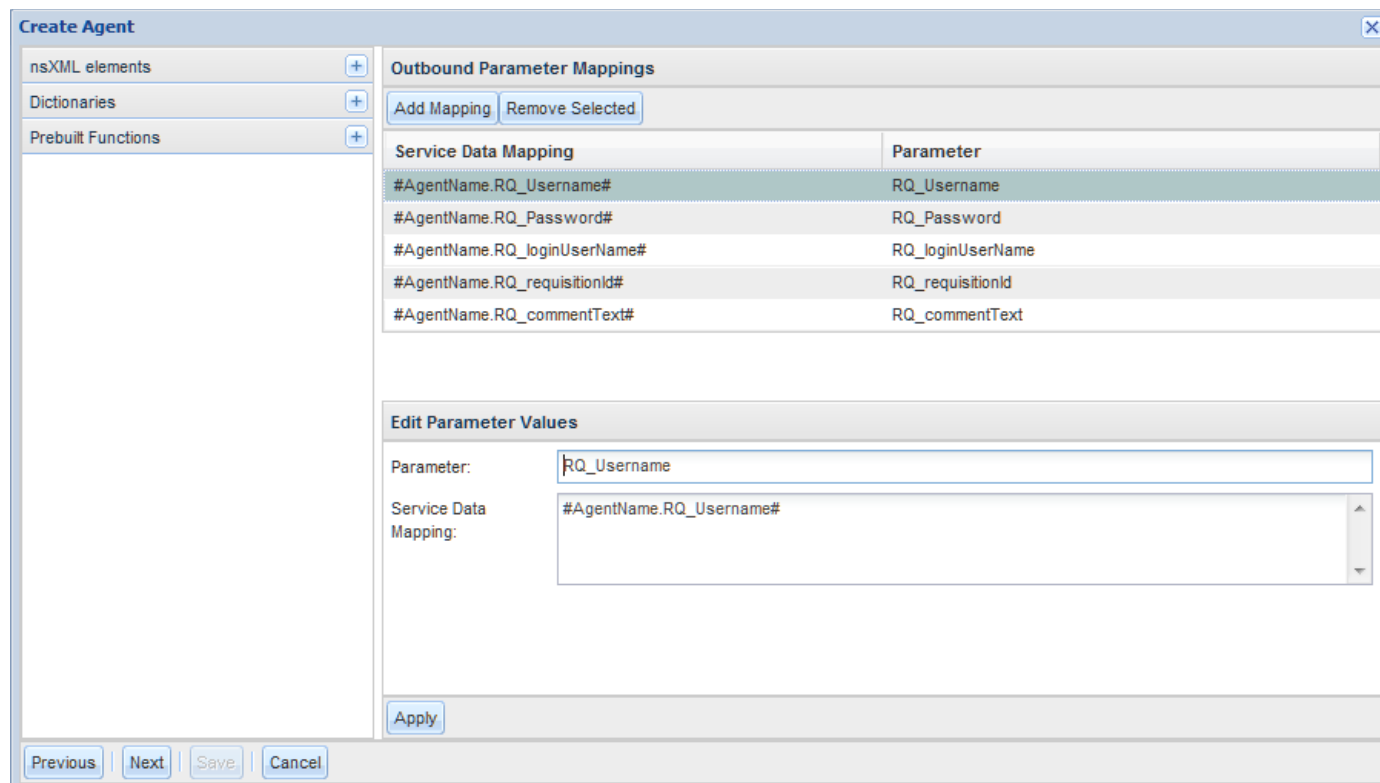
Enter the location of the WSDL containing the operation to be performed by the integration. This will typically be the URL where the WSDL resides.

The Integration Wizard reads the wizard and displays a list of supported operations. Select the desired operation. The attributes specified for the operation will drive the definition of agent parameters on subsequent pages of the wizard.

If the wsdl includes a routing url, that, too, is displayed.

If desired, click the **Advanced Properties** drop-down button to display additional settings for the integration. These may be entered now or specified later via Service Link. Only basic authentication can be specified via the wizard.

Outbound Request Parameter Mappings



The wizard parses the wsdl and, in the sample shown, determines that it includes two attributes that must be used in the web service outbound request. Therefore, it creates two agent parameters whose names match the names of the attributes in the wsdl.

The agent parameters are mapped to dictionary fields. The field names match the names of attributes in the wsdl, and the dictionary name matches the agent name. This dictionary is automatically created when you save the agent.

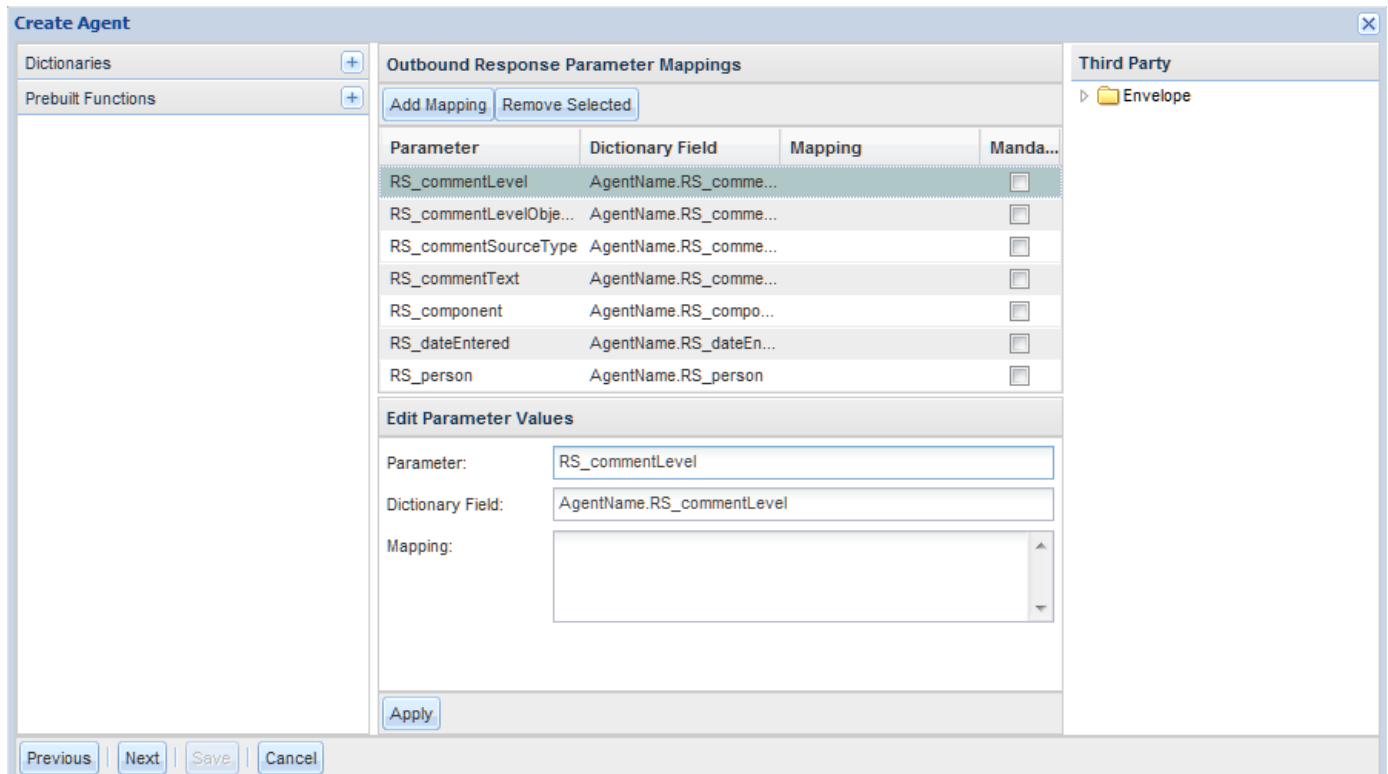
If desired, you can change the Service Data Mapping to refer to a dictionary and field that have previously been defined in Service Designer. This effectively changes the agent parameter mapping. However, the dictionary created by the wizard will still contain the original field. You may remove this by editing the dictionary definition.

A short digression might be useful here about structuring and using dictionaries in services. The primary purpose of a Request Center dictionary is to structure the data to be shown to users on a service form. Therefore, service designers typically design dictionaries with the user interface in mind, grouping and arranging fields to optimize the experience of both customers and service team members.

In principle, an outbound message might need to include data that has been entered (or defaulted or computed) in fields in many dictionaries. However, this would make maintaining the agent parameter mappings more complicated and prone to error—integration designers would have to be well acquainted with the design of the service form and its dictionaries. Therefore, it is recommended practice to create a dictionary solely for the purpose of containing integration data. Some fields in the dictionary may be redundant with fields displayed on the service form. In this case, the service designer should supply conditional rules to copy the value of the field from the displayed dictionary to the integration dictionary. Further, the integration dictionary is not displayed on the service form (it is typically hidden via an active form rule); however, it can be kept visible during development to facilitate debugging.

Outbound Response Parameter Mappings

By default, the Integration Wizard assumes that a response received from the target system will be processed. Any attributes sent in the response have corresponding agent attributes that are mapped to dictionary fields.



In addition to a agent-to-field correspondence, the mapping may include simple XSLT operations, available via the Prebuilt Functions drop-down arrow to the left of the page.

As for outbound parameters, the inbound parameter could also be mapped to an alternative dictionary field. All dictionaries can be browsed via the Dictionaries drop-down arrow to the left of the page.

Integration Summary

The last page of the wizard summarizes the integration as defined. You may return to any previous page to make corrections or click **Save** to save the agent and all other integration components created. By default, the agent is started when the integration is saved. You can alter this behavior by unchecking the “Start agent upon saving” check box.

Name	Value
Name	AgentName
Action	Agent Action
Active Form Component Name	AgentName
Dictionary Name	AgentName
Outgoing Content	Data and Parameters; No Service Details (default; small)
Failed Email	None
Description	
Outbound Transformation	AgentName Transformation

Start agent upon saving

Outbound Properties

Outbound Parameter Mapping

Outbound Response Parameter Mapping

Previous | Next | Save | Cancel

All components are now available for editing via Service Link and Service Designer screens. These components are shown in [Table 2-29](#).

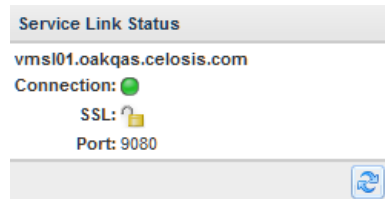
Table 2-29 **Integration Components**

Integration Component	Description
Agent	An agent using an http/ws outbound adapter, with agent parameters and an associated transformation.
Transformation	The transformation required to convert outbound nsXML to the expected format for the WSDL operation selected.
Dictionary	Dictionary containing fields corresponding to all outbound and inbound parameters. The dictionary is created in the Integration group; it can be moved if desired.
Active Form Component	Active form component containing the dictionary created. The form component is created in the Integration group; it can be moved if desired.

Service Link Troubleshooting and Administration

Checking Service Link Status

The starting point for checking the operational status of Service Link is the Service Link Status display. The Service Link status is always displayed beneath the Common Tasks area of the Service Link Home page.



This feature helps you verify that Service Link is communicating with the Request Center service via its assigned port.

The Service Link Status display indicates whether the Service Link connection status is operational, and shows the port and protocol being used.

Starting and Stopping Agents

Agents can be started and stopped individually by using the Control Agents page.

If the Service Link service is stopped and restarted, all agents that were running when the service was stopped are automatically restarted.

Logging

All adapters log their activities into the server log file.

In addition, each standard adapter has its own log file on the Service Link\log directory. The degree of detail written to the log is configurable; instructions for doing so are application-server specific.

For details on managing both server and adapter-specific log files, see the *Cisco Service Portal Configuration Guide*.

JBoss Logging

In a JBoss installation, Service Link adapter logs can be segregated from the server log into separate files by modifying the logging.properties file under the "<JBoss_DIR>\standalone\configuration" directory. Examples of such configurations can be found in the sample property files "\preinstall\jboss\templates" directory in the product package.

WebLogic Logging

WebLogic does not allow separation of log files per adapter, and the IntegrationServer component is configured to use the WebLogic logger by default. If separation of logs is desired, edit the file `newscalelog.properties` under `ISEE.war/WEB-INF/classes`. Uncomment the line that specifies commons logging as the logging mechanism. It is also very important that you uncomment and set a valid value for `logger.directory` to a valid and existing directory in the system, where the user that is used to run IntegrationServer has full write access. The file `newscalelog.properties` has additional instructions. In addition, if additional settings for other adapters are desired, edit the file `log4j.xml` and use the `FILE_ADAPTER` appender and category as a base and adjust the appender name and reference, the package of the appender and the file name.

WebSphere Logging

WebSphere logging of Service Link is based by default on log4j as included in the WebSphere application server. The log4j implementation in WebSphere is powerful and configurable through the administration console and other tools. However, it does not allow for easy separation of log files. If you want to separate log files per adapter in WebSphere, follow the steps below:

Step 1 Under “`ISEE.war/WEB-INF/classes/config`”, locate the file `newscalelog.properties` and open it with an editor.

Step 2 Uncomment the line:

```
logger.class.name=com.newscale.bfw.logging.LogUtilCommonsImpl
```

Step 3 Locate the line for `logger.directory`. Specify the log directory; for example:

```
logger.directory=I:/logfiles/servicelinkserver2
```

It is very important that you enter a valid directory where these log files reside and the user that is used to run IntegrationServer has full write access to it.

Step 4 Under the “`ISEE.war/META-INF/`” directory, manually create a folder named “`services`”.

Step 5 Under this `services` folder, manually create a text file named “`org.apache.commons.logging.LogFactory`”. Within the file, add one line as follows:

```
org.apache.commons.logging.impl.Log4jFactory
```

Message Purging

Utilities to purge Service Link messages are available as a database script. To execute the purge script, access the RequestCenter database with a SQL tool appropriate for your database as a DML user, and execute the procedure “`sp_CleanUpSIMessageContent`”. You will need to provide the number of calendar days of messages to retain as the input parameter for this script. The utility actually does not purge message data, but sets the message content to “Message has been purged” for all completed or failed messages older than ninety days.

Application Server Configuration Files

You can analyze the following files when troubleshooting.

- **rcjms.properties file** – This file contains the information about the integration outbound JMS queue and can be located in the “/RequestCenter.war/WEB-INF/classes/config” directory. The business engine puts the message in the queue specified in this file. The values for the following properties should match with those in the integrationserver.properties file in the ISEE.war file:

ISEEOutbound.JndiProviderUrl

ISEEOutbound.JndiFactory

ISEEOutbound.JmsTopicFactory

ISEEOutbound.JmsQueueFactory

ISEEOutbound.JmsQueue

ISEEOutbound.JmsTopic

- **integrationserver.properties file** – This file contains the information about inbound and outbound JMS queues and can be located in the “/ISEE.war/WEB-INF/classes” directory. Verify the JMS properties specified in this folder.
- **newscale.properties file** – This file contains the property for **isee.base.url**. Ensure that it points to the Service Link server url.

Online Error Log

In addition to the server log file and adapter-specific log files, any errors detected by Service Link can also be viewed online. The message text for a failed message shown on the Messages page is a hyperlink to the detailed error for that message.

The error messages are exactly those that appear in the server logs and may be highly technical. Some sample error messages, and an explanation, are given below.

```
com.newscale.is.core.RoutingException: Routing exception found: Reference Field not
retrieved from response
```

An outbound web services message was sent, but the inbound response could not be processed because the specified referenced field was not in the response message.

```
com.newscale.is.core.TransformationException: javax.xml.transform.TransformerException:
javax.xml.transform.TransformerException: Tag is not allowed in this position in the
stylesheet!
```

The transformation produced an invalid XML message.

Prebuilt Functions

Overview

Prebuilt functions provide the ability to manipulate the values of agent parameters included in a nsXML message.

Prebuilt functions were developed using the FreeMarker template engine, version 2.3.12, available as open source software and developed by the Visigoth Software Society. Cisco has certified only those functions documented below and available in the drop-down list when building agent parameters. Other functions supported by the FreeMarker framework may be used, but should be extensively tested.

Function Usage

Basic function usage consists of applying to the function to an expression, specifying an argument list for the function if required. In general terms:

```
${Expression?function(argumentList)}
```

For Service Link, the expression is typically either a dictionary field, specified via lightweight namespace syntax, or an nsXML element. It must be enclosed in quotes:

```
${"#Customer_Information.Login_ID#"?upper_case}
```

Two or more functions can be chained-applied to the same expression-by using the syntax:

```
${Expression?function1(argumentList)}${"$Parameter$"??function2}
```

For example, the service data mapping below first trims any leading or trailing spaces from the designated dictionary field, then converts the result to lower case.

Edit Parameter Values	
Parameter:	<input type="text" value="RQ_loginUserName"/>
Service Data Mapping:	<input #customer_information.login_id"?trim}\${"\$parameter\$"??lower_case}"="" type="text" value="\${"/>

Multiple elements can be combined in one mapping, as shown below. The elements are implicitly concatenated together to form one string.

Inbound Parameter Mappings

[Add Mapping](#) [Remove Selected](#)

Parameter	Dictionary Field	Mapping	Mandatory
Name	Temp.text1		<input type="checkbox"/>
Domain	Temp.text2		<input type="checkbox"/>
Email	Customer_Information.Email_Address	\${"\$Name\$"?lower_case}@\${"\$Domain\$"...	<input type="checkbox"/>

Edit Parameter Values

Parameter:

Dictionary Field:

Mapping:

[Apply](#)

This scenario also shows another coding technique—the use of “temporary” fields to hold input values so they can be used in a mapping expression.

Function Synopsis

substring

The substring function has the syntax:

```
exp?substring(from, toExclusive), also callable as exp?substring(from)
```

A substring of the string, *from* is the index of the first character. It must be a number that is at least 0 and less than or equal with *toExclusive*, or else an error will abort the template processing. The *toExclusive* is the index of the character position after the last character of the substring, or with other words, it is one greater than the index of the last character. It must be a number that is at least 0 and less than or equal to the length of the string, or else an error will abort the template processing. If the *toExclusive* is omitted, then it defaults to the length of the string. If a parameter is a number that is not an integer, only the integer part of the number is used.

index_of

Returns the index within this string of the first occurrence of the specified substring. For example, `“abcabc”?index_of(“bc”)` will return 1 (don't forget that the index of the first character is 0). Also, you can specify the index to start the search from: `“abcabc”?index_of(“bc”, 2)` will return 4. There is no restriction on the numerical value of the second parameter: if it is negative, it has the same effect as if it were zero, and if it is greater than the length of this string, it has the same effect as if it were equal to the length of this string. Decimal values are truncated to integers.

If the 1st parameter does not occur as a substring in this string (starting from the given index, if you use the second parameter), then it returns -1.

last_index_of

Returns the index within this string of the last (rightmost) occurrence of the specified substring. It returns the index of the first (leftmost) character of the substring. For example: `"abcabc"?last_index_of("ab")` will return 3. Also, you can specify the index to start the search from. For example,

```
"abcabc"?last_index_of("ab", 2)
```

will return 0. Note that the second parameter indicates the maximum index of the start of the substring. There is no restriction on the numerical value of the second parameter: if it is negative, it has the same effect as if it were zero, and if it is greater than the length of this string, it has the same effect as if it were equal to the length of this string. Decimal values are truncated to integers.

If the first parameter does not occur as a substring in this string (before the given index, if you use the second parameter), then it returns `-1`.

length

The number of characters in the string.

lower_case

The lower case version of the string. For example, `"GrEeN MoUsE"` becomes `"green mouse"`.

replace

It is used to replace all occurrences of a string in the original string with another string. It does not deal with word boundaries. For example:

```
${"this is a car acarus"?replace("car", "bulldozer")}
```

will print:

```
this is a bulldozer abulldozerus
```

The replacing occurs in left-to-right order. This means that this:

```
${"aaaaa"?replace("aaa", "X")}
```

will print:

```
Xaa
```

If the first parameter is an empty string, then all occurrences of the empty string are replaced, like `"foo"?replace("", "|")` will evaluate to `"|f|o|o|"`.

`replace` accepts an optional **flags parameter**, as its third parameter.

upper_case

The upper case version of the string. For example, `"GrEeN MoUsE"` becomes `"GREEN MOUSE"`.



CHAPTER 3

Service Link Adapter Development Kit

- [Overview, page 3-1](#)
- [Getting Started, page 3-1](#)
- [What is an Adapter?, page 3-4](#)
- [Example Adapter, page 3-6](#)
- [nsXML Format, page 3-13](#)
- [Sample Inbound and Outbound Documents, page 3-24](#)

Overview

This chapter describes how to use the Service Link Adapter Development Kit (ADK) to develop Service Link adapters. The ADK is the set of components that allow the production of adapters for the Service Link subsystem of Request Center. Service Link provides external communications for Request Center and provides for coordinated externalization of workflow tasks with other systems.

To achieve this communication, Service Link supports installable adapters. Service Link ships with standard adapters, but developers can create others. This chapter describes the process of writing adapters.

Intended Audience

This chapter is intended for:

- **Administrator.** The administrator has access to the product packages and can install Service Portal products in a customer system.
- **Adapter Developer.** The adapter developer is a person that is well versed in java technologies, including ANT, and it is a subject matter expert of the integration required.

In development environments, the same person may fill both these roles.

Getting Started

This section describes the installation of the ADK, its structure, compiling adapters, and adapter deployment.

Installing the JDK

Follow the instructions from Sun or IBM to install the Java Development Kit. Service Portal is certified with Sun JDK 6 for installation on WebLogic 10.3 or JBoss 7.1.1, and with IBM Java 1.6 for installation on WebSphere 7.0.0.17.

Installing the ADK

To install the ADK:

1. **Administrator:** Expand the context of the product packages, locate the adk.zip under the image/isee/dist folder, and inform the adapter developer of the location.
2. **Adapter developer:** Obtain the file adk.zip (or adk.tar.gz) from the administrator.
3. **Adapter developer:** Expand the ADK package in a local machine, in C:\ADK. It does not have to be the C drive, nor the ADK directory. However, the examples in this chapter use C:\ADK.

ADK Structure

After installing the ADK, the following subdirectories exist:

Directory	Description
<root>	Contains the build procedure files.
ant	Complete ANT build system. This ANT is the standard Apache ANT build system, with some added extensions.
doc	Contains the java doc subdirectory. You may place the ADK documentation here.
doc\javadoc	The help for the ADK in javadoc format.
example	Contains our example adapter.
example\src	Contains the source java files for the example adapters.
example\deploy	Contains adapter.xml, which describes this adapter.
lib	Contains the ADK libraries needed for compilation.

An adapter is a subdirectory in the main ADK structure. After installing, **example** is one such adapter. Adapter code has to be structured in the following way:

Directory	Description
<adapter>	The short name of the adapter. In the case of the example adapter, this is example .
<adapter>\src	Mandatory. The root for the source java files.
<adapter>\deploy	Mandatory. The deployment directory. This should contain a file called adapter.xml .
<adapter>\lib	Optional. Additional libraries to be added to the lib directory of ISEE.war .
<adapter>\config	Optional. Additional files to be copied to the classes directory of ISEE.war .

In the example provided, only **src** and **deploy** exist.

After compiling the files, create a staging directory (see the following sections for a description on how to build adapters). The staging directory can be deleted and recreated with the build procedure later.

Directory	Description
staging	The root of the production directory.
staging\classes	The compiled java classes for the adapters.
staging\adapters	Contains the built jars for each of the adapters. The adapters appear with the name adapter_<adaptershortname>.jar.
staging\config	The files from each config subdirectories for each adapter.
staging\deploy	The files from each of the deploy subdirectories, renamed as <adaptershortname>.xml.
staging\lib	The files from each of the lib subdirectories for each adapter.
staging\dist	The final isee.adapters deployable file.

Creating Adapter Source Structures

The procedure to create new adapters is:

-
- Step 1** Create the directory structure as defined above.
 - Step 2** Create the source and place it in the structure.
 - Step 3** Create adapter.xml and place it in the deploy directory. For more information, see the [“Understanding the adapter.xml Descriptor” section on page 3-10](#).
 - Step 4** Optionally add additional libraries and configuration files.
 - Step 5** Modify build.properties and add your adapter to the adapters line. This configures ANT to look for the created directories.
-

The compilation steps allow for adding the build to version control, and later compiled before installation.

Compiling Adapters

After creating the adapter, build it by executing:

build.cmd (or **./build.sh** for unix systems)

The final product appears under **staging/dist/isee.adapters**. This file needs to be provided to the administrator for deployment.

Deploying Adapters

To deploy an adapter, the administrator performs the following procedures:

-
- Step 1** Obtain Service Link custom adapters package from provider, or compile it according to company standards if it is a custom adapter developed internally.
 - Step 2** Unzip the adapter to a temporary directory (for example, c:\temp\adapter). This directory is hereinafter referred to as <AH>.
 - Step 3** Copy the <AH>/adapters/<ADAPTER_NAME>.jar to the deployed “ISEE.war/WEB-INF/lib” directory.
 - Step 4** Copy the <AH>/lib/* files (if any) to the deployed “ISEE.war/WEB-INF/lib” directory.
 - Step 5** Copy the <AH>/config/* files (if any) to the deployed “ISEE.war/WEB-INF/classes” directory.
 - Step 6** Copy the <AH>/udk/* files (if any) to the deployed “ISEE.war/WEB-INF/classes” directory.
 - Step 7** Open a command window and cd into the <ADK_HOME>/lib folder (created from extracting the adk.zip or adk.tar.gz). For each *.xml file in the <AH>/deploy directory (there should be one xml file per adapter), execute adapter_dbinstaller.sh or adapter_dbinstaller.cmd as appropriate to your environment. Use --help or -? as the argument to the adapter_dbinstaller.sh or adapter_dbinstaller.cmd to see the list of required input arguments.
 - Step 8** For each udk file that was installed (Step 6), add the file’s name to the “UDConfig” property inside the integrationserver.properties file. The UDConfig property is a comma-delimited list of all udconfig files. Append the adapters udconfig files to this list.
 - Step 9** Start the Request Center and Service Link servers. Verify the new adapter exists.
-

What is an Adapter?

Concepts

An adapter is the vehicle by which Service Link connects with external systems (often referred as third-party systems). Adapters are composed of three pieces:

- An inbound piece, referred to as the **inbound adapter**
- An outbound piece, referred to as the **outbound adapter**
- An error handler

The inbound adapter manages incoming communications into Request Center. It processes the XML messages coming into the system. There are two types of inbound adapters: pollers and listeners. A poller is a thread that periodically wakes up and looks for incoming messages, while a listener waits and is awakened by an external event. An example of a poller is the inbound file adapter, which needs to periodically check for messages. An example of a listener is the HTTP adapter which waits until an HTTP XML event is posted.

Outbound adapters manage the XML messages coming out of Request Center. There is only one type of outbound adapter.

An agent is a logical element that protects service designers from having to know all the complexities of adapter and connection properties. An agent defines an inbound adapter and an outbound adapter. The inbound adapter is optional and can be specified as “Auto complete”. “Auto complete” is a mode whereby the system does not need a reply from a third party for the workflow to proceed, and is mostly associated with unreliable, or shoot-and-forget protocols, such as an email-based system. The administrator configures agents and their associations with adapters for the service designers to use.

In addition, XML transformations can be applied to messages before they go to a third-party system, or after they are received from a third-party system and delivered to Service Link.

The message system uses a common XML dialect known as nsXML, which is a schema that defines the valid XML that Service Link can process or produce. nsXML currently consists of six operations:

- task-started – outgoing
- task-cancelled – outgoing
- take-action – incoming
- update-data – incoming
- send-parameters – incoming
- add-comment – incoming

When outgoing, Service Link can transform these operations to a destination. The same is true for incoming messages, and the XSL transformations can convert the external format into the nsXML dialect.

See the “[nsXML Format](#)” section on page 3-13 for more information about these nsXML operations.

Types of Adapters

The adapters are of two types:

- Transport Adapters
Transport adapters are specific to a given transport, such as HTTP, file, JMS, or some proprietary network socket implementation.
- Application Adapters
Application adapters have an element of transport but are better understood by the specific third-party application, such as Remedy and Siebel. In many cases native APIs are provided through jars. In this version of Service Link, transport adapters cannot yet be extended to create application adapters.

Agents may use different adapters for inbound and outbound messages.

Adapter Components

In addition to java code, an adapter is composed of:

- Jar libraries (for example, Remedy java API)
- Static configuration files. These are highly discouraged as most customers may not allow changing of text files once deployed.
- Deployment descriptor. An XML file that describes the adapter.

Properties

In order to connect to third-party systems, adapters may expose connection properties that the Service Link module exposes to administrators. They are described in the XML deployment descriptor, and their values can be retrieved by the java code to a well established API.

Example Adapter

This section illustrates how to implement a simple adapter. The example adapter is a file adapter that communicates with the external world.

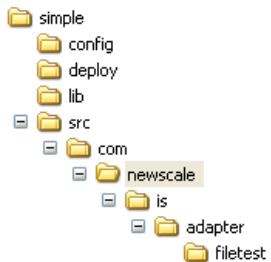
The simple file adapter contains:

- An outbound adapter that creates a file, whose file name is specified through adapter properties.
- An Inbound adapter that reads a file, whose file name is specified through adapter properties.
- A simple exception handler.

This adapter is not very useful in real life, because multiple calls override the outbound file, and similarly with the inbound file. However, it demonstrates the process that needs to be followed to create an adapter.

Directory Structure

First, create the adapter's directory structure. In the ADK directory structure, create a directory named **simple**, and create the following directory structure under it:



Under **src** notice the source package representing the java package **com.newscale.is.adapter.filetest**. Any other package can be used, but this example uses this one.

Outbound Adapter Class

Secondly, create the outbound adapter class. The name is **FileOutboundAdapter** and this class file should be placed in the package described in step one. Its skeleton is shown below, without the implementation of the methods.

```

import com.newscale.is.adk.AdapterContext;
import com.newscale.is.adk.base.OutboundAdapter;
import com.newscale.is.adk.exceptions.AdapterException;
public class FileOutboundAdapter extends OutboundAdapter {
    public FileOutboundAdapter (AdapterContext context) {
        super(context);
    }
}

```

```

public void initiate (AdapterContext context) throws AdapterException {
}
public void processMessage (String message, String channelId) throws AdapterException {
}
public void terminate () throws AdapterException {
}
public void commit() throws AdapterException {
}
public void rollback() throws AdapterException {
}
}

```

To create an outbound adapter, the class needs to extend the class **com.newscale.is.adk.base.OutboundAdapter** as shown above.

Implement a constructor that receives a **com.newscale.is.adk.AdapterContext** as a parameter. The recommended way to implement this constructor is also shown above: calling the super constructor.

Implement the **initiate** method as shown above. This method is called when an agent using the adapter is started. If this method is empty, you can omit it.

Implement the **processMessage** method. This method is called when a message is ready to be sent. If a transformer is specified in the agent, the transformer has transformed the message.

Implement the **terminate** method. Call this method is when the agent stops. If this method is empty, you can omit it.

Implement the **commit** method. Call this method when the agent is about to complete its transaction. If this method is empty, you can omit it. This method is used so that a transaction can be started in the **processMessage**, and later completed.

Implement the **rollback** method. This method is called when the agent is about to rollback its transaction. If this method is to be left empty, it can be omitted. This method is used so that a transaction can be started in the **processMessage**, and later recalled.

More information about transaction support can be found in the [“Transaction Support” section on page 3-9](#).

In our case, the file outbound class writes a file with the contents of the xml. To achieve that, first set up a variable that keeps the file name where the file is stored. For this purpose, use the agent properties.

```

String path = null;
public void initiate (AdapterContext context)
    throws AdapterException {
    Properties properties = context.getProperties();
    this.path = properties.getProperty("OB_FILE_DIR") + "/" +
        properties.getProperty("OB_FILE_NAME");
}

```

When the string is received, writing it to the file is trivial.

```

public void processMessage (String message, String channelId)
    throws AdapterException {
    try {
        Writer w = new FileWriter(path);
        w.write(message);
        w.close();
    } catch (Exception e) {
        e.printStackTrace();
        throw new AdapterException(1, "Problem while writing to a file: " +
            e.getMessage());
    }
}

```

Of course, this code has been oversimplified for the sake of explanation.

Poller Inbound Adapter Class

The skeleton for our inbound adapter is as follows:

```
public class FileInboundAdapter extends InboundAdapter {
    public FileInboundAdapter (AdapterContext context) {
        super(context);
    }
    public void initiate (AdapterContext context) throws AdapterException {
    }
    public String receiveMessage () throws AdapterException {
        return null;
    }
    public void terminate () throws AdapterException {
    }
    public void commit()
        throws AdapterException {
    }
    public void rollback()
        throws AdapterException {
    }
}
```

The semantics of the methods are just like the ones in the outbound adapter. The only exception is the **receiveMessage** method. The **receiveMessage** method is called periodically in the case of a poller adapter. If data is found, then the method returns a valid xml in third-party format. If no data is found, null is returned. The code for the inbound adapter is as follows (just like the outbound adapter, the initialization is done with the correct parameters):

```
String path = null;
public void initiate (AdapterContext context)
    throws AdapterException {
    Properties properties = context.getProperties();
    this.path = properties.getProperty("IB_FILE_DIR") + "/" +
        properties.getProperty("IB_FILE_NAME");
}
```

Processing of the file is done as follows:

```
public String receiveMessage () throws AdapterException {
    String receivedMessage = "";
    char data[] = {};
    try {
        StringBuffer buffer = new StringBuffer();
        FileInputStream fis = new FileInputStream(path);
        InputStreamReader isr = new InputStreamReader(fis, "UTF8");
        Reader in = new BufferedReader(isr);
        int ch;
        while ((ch = in.read()) > -1) {
            buffer.append((char) ch);
        }
        in.close();
        String requestString = buffer.toString();
        boolean success = (new File(path)).delete();
        return requestString;
    } catch (Exception e) {
        return null;
    }
}
```


Listener Inbound Adapter

A listener adapter is created by virtue of an ad-hoc process. Two classes are in play: the inbound adapter class, and an actual receiver class, like a servlet. The receiver class is required to obtain the channel ID. The receiver class locates the `InboundAdapter` class as follows:

```
ChannelInfoVO chVo = AgentDAO.getInstance().getChannelInfo(channelId);
if(chVo != null){
    Adapter adapter =
        AgentManager.getInstance().getAdapter(chVo.getAgentId());
    ((InboundAdapter).receiverProcess(xml);
}
```

The inbound adapter has a method called **receiverProcess** that should be called with the message, or an object whose **toString()** method returns the text of the message. The example does not provide a listener inbound adapter.

Exception Handler

Once the two adapters are done, the exception handler needs to be implemented. In our case it is a very simple class, where all we do is output the error. The complete class is shown here:

```
public class FileExceptionHandler implements ITransportExceptionHandler {
    public FileExceptionHandler () {
    }
    public void handleException (Map props, String message) {
        System.out.println("Outbound Message Failed to deliver: " + message);
    }
}
```

Transaction Support

Transaction support has been provided to the adapters so that agents get notified before they undo their own transactions. The methods **commit** and **rollback** have been added for that purpose.



Note

No logic code should be added to these methods, as the system is in the middle of committing or rolling back a transaction. These methods should only rollback or commit their resources.

To track resources to be committed or rolled back, an adapter can use this common technique:

Create a static map. Once the processing method is called (either `processMessage` or `receiveMessage`) the method can add:

```
private static Map resources = new HashMap();
public void processMessage (String message, String channelId)
    throws AdapterException {
    Connection con = ... // obtain a connection to external resource
    Map.put(Thread.currentThread(), con);
}

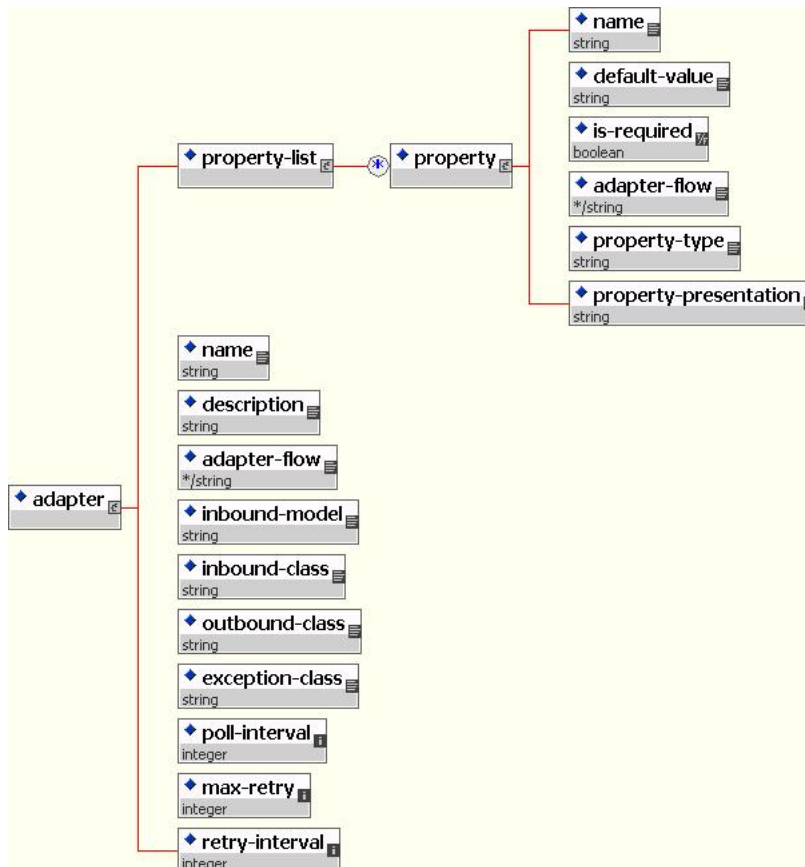
public void commit() throws AdapterException {
    con = (Connection) map.get(Thread.currentThread());
    map.remove(Thread.currentThread());
    con.commit();
}
```

Understanding the adapter.xml Descriptor

The adapter descriptor contains information for the deployment of the adapter and its properties.

The Adapter Schema

The adapter schema is as follows:



Description of “adapter” Element Fields

name: Name of the adapter

description: Description of the adapter

adapter-flow:

Valid values for this are:

- “inbound”
- “outbound”

inbound-model:

Valid values are:

- “listener”
- “poller”
- “extendedpoller”

inbound-class: Absolute class name of inbound adapter

outbound-class: Absolute class name of outbound adapter

exception-class: Absolute class name of exception handler for this adapter

poll-interval: Poll interval (applicable for “poller” type adapter) in milliseconds

max-retry: Max number of retries in case of message failure

retry-interval: Interval between retries in milliseconds

Description of “property” (Adapter Properties) Element Fields

name: Name of the adapter property

default-value: Default value for the property

is-required: Whether this is a mandatory or optional property. Valid values are “true” or “false”

property-type: The type of property. Valid values are “string” for now.

property-presentation: Valid values are “text” and “password”

adapter-flow:

Valid values are:

- “inbound”
- “outbound”
- “both”

Adapter.xml Example

```
<?xml version="1.0" encoding="UTF-8"?>
<adapter>
<property-list>
<property>
<name>InboundFinalResolution</name>
<default-value>Preserve</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundFileLocation</name>
<default-value>C://SL2//InboundFiles</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OnError</name>
<default-value>Preserve</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
```

```

<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundBackupLocation</name>
<default-value>c://SL2//InboundBackup</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>BackupSuffix</name>
<default-value>.bak</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>FileNameDateFormat</name>
<default-value>.yyyyMMddHHmmssSSS</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>InboundTempLocation</name>
<default-value>C://SL2//InboundTemp</default-value>
<is-required>true</is-required>
<adapter-flow>inbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundConflictResolution</name>
<default-value>Rename</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundFileLocation</name>
<default-value>C://SL2//InboundFiles</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundBackupLocation</name>
<default-value>c://SL2//InboundBackup</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
<property>
<name>OutboundTempLocation</name>
<default-value>C://SL2//InboundTemp</default-value>
<is-required>true</is-required>
<adapter-flow>outbound</adapter-flow>

```

```

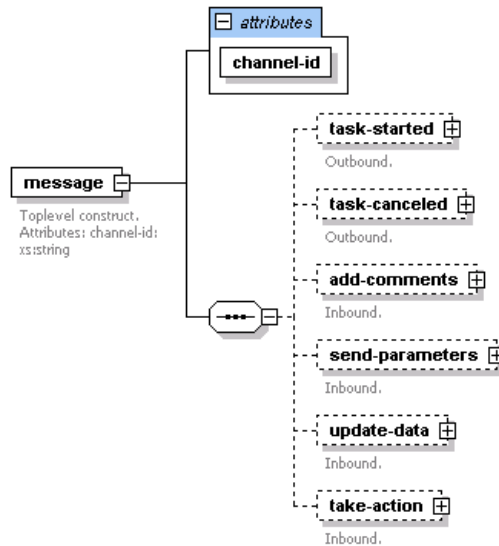
<property-type>string </property-type>
<property-presentation>text</property- presentation>
</property>
</property-list>
<name>File Adapter</name>
<description>Read/write the external data from/to external file system</description>
<adapter-flow>inbound</adapter-flow>
<inbound-model>poller</inbound-model>
<inbound-class>com.newscale.is.adapter.file.FileInboundAdapter</inbound-class>
<outbound-class>com.newscale.is.adapter.file.FileOutboundAdapter</outbound-class>
<exception-class>com.newscale.is.adapter.file.FileExceptionHandler</exception-class>
<poll-interval>10000</poll-interval>
<max-retry>0</max-retry>
<retry-interval>0</retry-interval>
</adapter>

```

nsXML Format

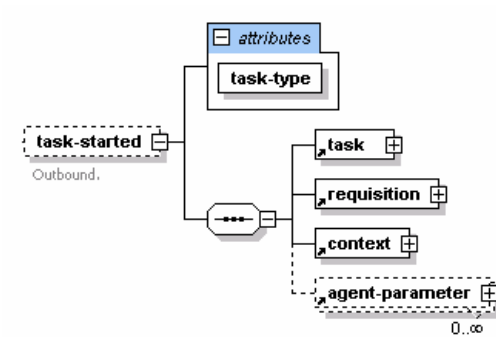
This section describes the details of the communication message content and structure. The message content is encapsulated in XML documents which are sent between Request Center and third-party systems over various carrier protocols such as HTTP, SOAP, or JMS. For easier understanding of the structures and substructures of messages, a graphical notation is used.

Message



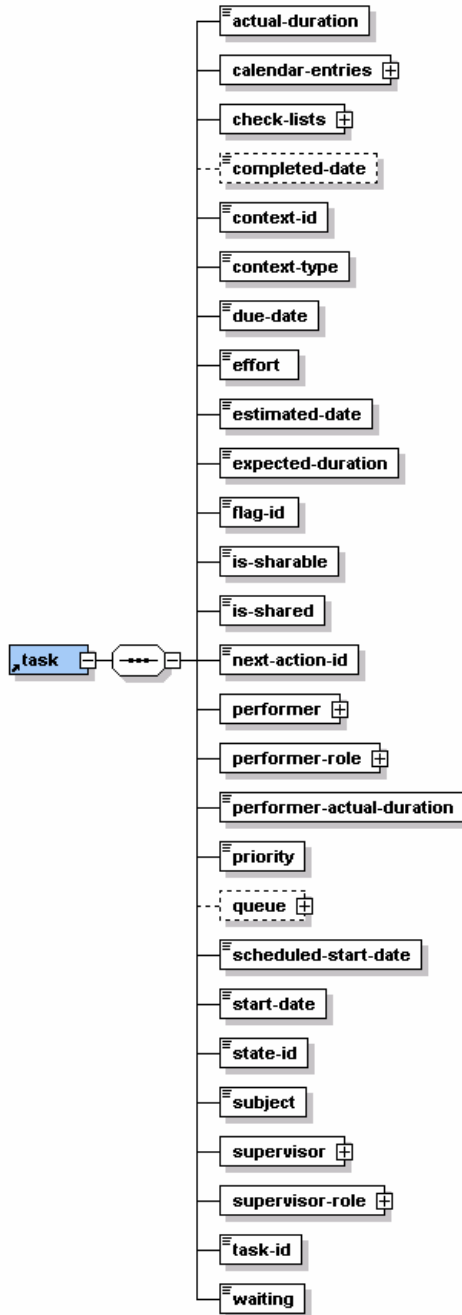
The outbound has a top level element message which contains the element “task-started” or “task-canceled”. The inbound document has a top level element message which contains one or many elements “add-comments,” “send-parameters,” “update-data,” or “take-action”. The message tag has a mandatory attribute which is called channel-id and is of type string. It is a unique string value created by Service Link for any outbound message created. The third-party system needs to reply back the message with the corresponding channel-id. This ID has to be carried on both the Service Portal and third-party system sides.

Task Started or Task Cancelled



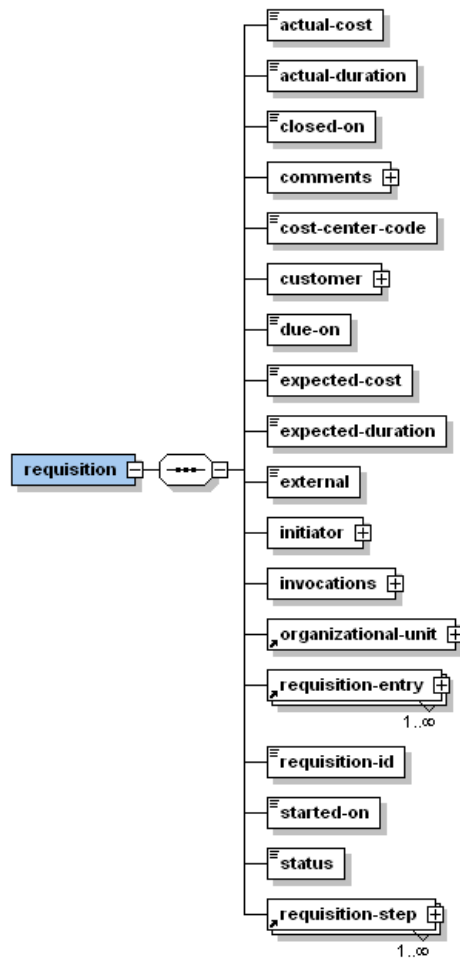
Task started kicks off an external activity in the third-party system. The design strategy followed for this operation is to incorporate all the information that may be required by the third-party system to execute the task. This element holds all the required details about the task and the requisition it belongs to. It may also contain one or more optional agent parameters. The context element describes the task in context of service delivery plan. This node does not contain values for Requisition-level reviews and approvals.

Task



Element	Description
actual-duration	Empty because the task has just become ongoing.
calendar-entries	The calendar entry of the person who requested a service.
check-lists	Task check list.
completed-date	Empty because the task has not yet been completed.
context-id	The object id of the context object in which the task gets initiated.
context-type	The object context, for example, Requisition Entry.
due-date	The due date for the task.
effort	Expected task effort in hours (how many working hours are expected to be required by one person to complete the task?).
estimated-date	Estimated completion date and time of the task.
expected-duration	Expected task duration in hours (how many working hours are expected to pass from the beginning of the task to the end?).
flag-id	Color indicator for the UI.
is-sharable	Boolean value indicating whether the task is sharable or not.
is-shared	Boolean value indicating whether the task is shared or not.
next-action-id	What is the next possible action for the task?
performer	The performer of the task. The performer element has an associated person object/.
performer-actual-duration	Empty because the task has not yet been completed/.
performer-role	What is the process role the performer is fulfilling?
priority	Task priority: 1, 2 or 3 for high, medium, or low, respectively/.
queue	Description of the queue to which this task has been assigned.
scheduled-start-date	Date on which the task is scheduled to be started/.
start-date	Date on which the task was started/.
state-id	Which state the task is in/.
subject	Subject of the task. The subject changes with the service definition, not with the requisition entry/.
supervisor	The supervisor of the task. The supervisor element has an associated person object.
supervisor-role	The process role the supervisor is fulfilling.
task-id	An integer used to uniquely identify this task instance. A new number is generated for each task of a requisition entry.
waiting	An indicator that represents the dependencies of this task including sub tasks and third-party systems.

Requisition

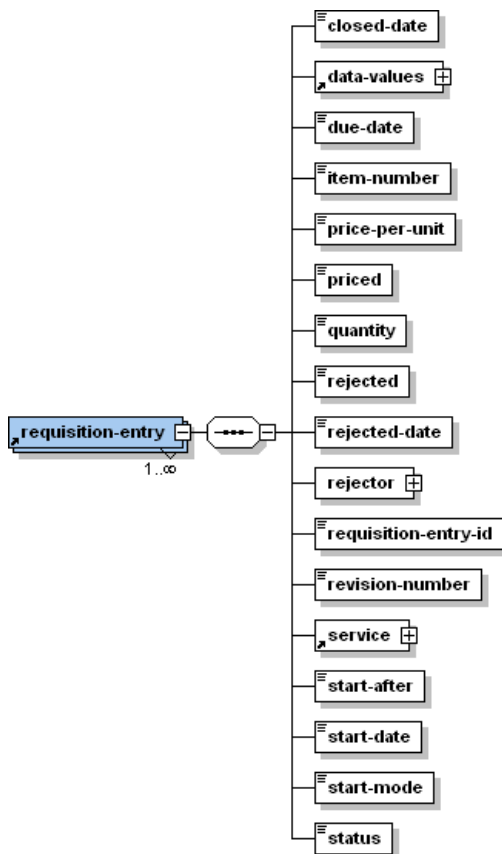


The requisition element encapsulates all requisition and requisition entry data that can be used for integration purposes when executing an external activity.

Element	Description
services	Number of services (or requisition entries) requested.
actual-cost	Actual cost of the requisition.
actual-duration	Actual duration of the requisition.
closed-on	Empty, as the requisition has not yet been completed.
comments	Comment on the requisition.
cost-center-code	Not used.
customer	The person for whom the requisition was ordered. It holds the person object data.
due-on	Date and time when the delivery of the requisition is due.
expected-cost	Expected cost of the requisition.
expected-duration	Expected duration in hours for handling the whole requisition.

Element	Description
external	Boolean value indicating whether the requisition has been initiated from an external system.
initiator	The person who ordered this requisition. It holds the person object data.
invocations	Attributes set up through RAPI (Requisition API).
organizational-unit	The organizational unit of the requestor (initiator).
requisition-entry	The requisition entry data.
requisition-id	Integer id of the submitted requisition. This is the same ID that can be seen in My Services and Service Manager after submitting a requisition manually.
requisition-step	The requisition authorization/delivery steps and status.
started-on	The date on which the requisition started.
status	State the requisition is currently in. While executing an external activity, this is ongoing.

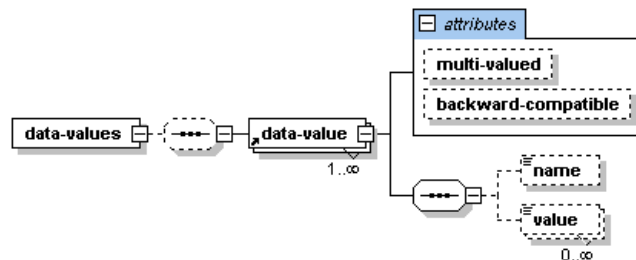
Requisition Entry



This tag encapsulates all the data of one requisition entry that can be used for integration purposes.

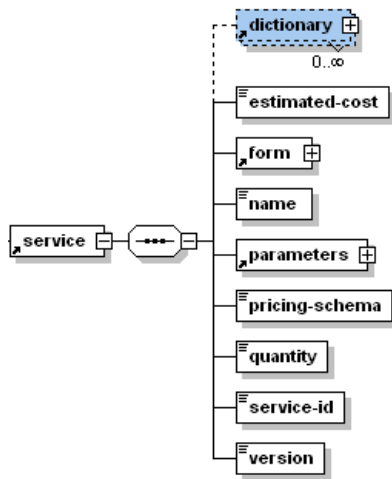
Element	Description
closed-date	Date and time when the requisition entry's status was changed from ongoing to completed. It is empty because the requisition is not closed when the task is ongoing.
data-values	Requisition entry data value.
due-date	Date on which this requisition is supposed to finish.
item-number	Item number of the requisition entry within the requisition.
price-per-unit	Unit price of the service requested.
priced	True if the price has been established and false if pricing is not done on the requisition.
quantity	Quantity ordered.
rejected	Indicates whether the requisition entry is approved or rejected.
rejected-date	If it is rejected, on what date.
rejector	Indicates the person who rejected the requisition.
requisition-entry-id	Entry ID.
revision-number	If the revision is made it indicates the revision number.
service	Element related to the service which this requisition entry belongs to.
start-after	Delayed start date.
start-date	The date on which the requisition entry got started.
start-mode	Specifies if the requisition entry starts immediately or late.
status	Status of the requisition entry: closed or ongoing.

Data Values



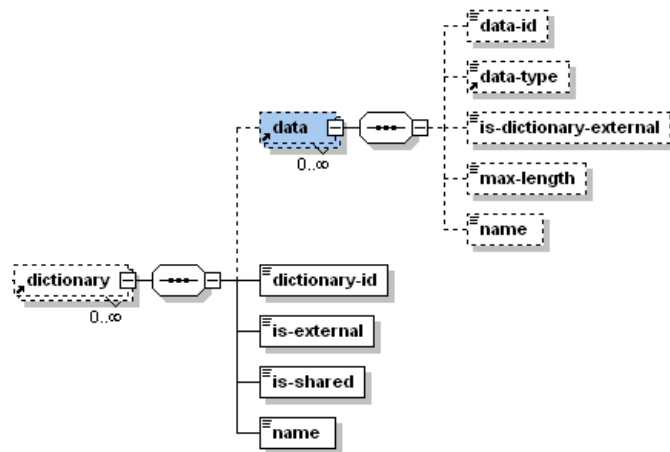
The data-values element can have one or more data values comprised of dictionary data. The data-value name indicates the “Dictionaryname.FieldName” and value is the value entered by the user while ordering the service. If the value is a multi-select drop down list, then one data-value element can have multiple values.

Service



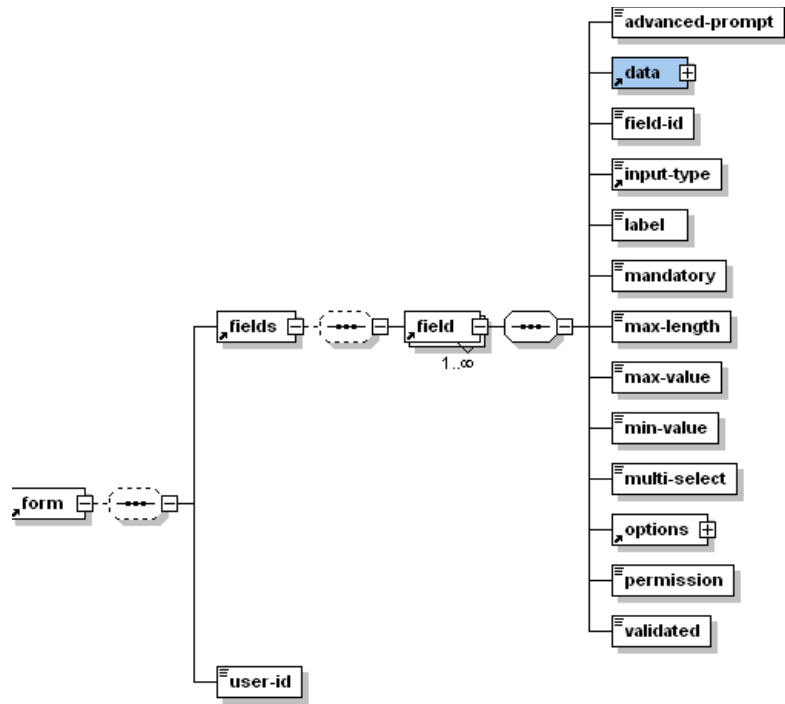
Element	Description
dictionary	A service element can have zero or more dictionaries.
estimated-cost	The estimated cost of the service.
form	Element which holds all the field elements of the service form.
name	Name of the service.
parameters	Parameters defined for this service.
pricing-schema	Specifies if the service is a bid, pricing task or fixed price.
quantity	How many quantities were ordered?
service-id	Id of the service in Request Center.
version	Last modified version number of the service.
standard-duration	Standard duration for the service.

Dictionary



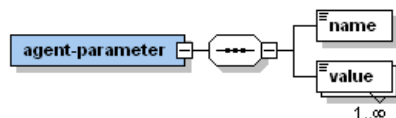
Element	Description
caption	A string value containing the caption data within the dictionary.
data	The data elements within the dictionary. The data element holds values for the data element name, maximum length, data type and other metadata.
dictionary-id	The dictionary id of a dictionary within Request Center.
dictionary-template-type-id	The template used for creating the dictionary (for example, 2 for person-based dictionaries).
classification-id	The dictionary classification (applicable to Service Item dictionaries only).
mdr-data-type-id	The dictionary service item type (applicable to Service Item dictionaries only).
display-order	An integer value containing the display order of the dictionary.
is-external	A Boolean value which indicates whether the dictionary is an internal Request Center dictionary or is external.
is-reportable	A Boolean value stating whether the dictionary has been marked as reportable for use with the Advanced Reporting module's Ad-Hoc reporting feature.
is-shared	A Boolean value which indicates whether the dictionary is a shared dictionary or not.
is-template	A Boolean value which indicates whether the dictionary is a template; the value is always false.
logic-name	Internal name of the dictionary (applicable to reserved dictionaries only).
name	Name of the dictionary.

Form



Element	Description
fields	<p>Fields have one or many field elements inside a requisition form.</p> <p>advanced-prompt – Rich html prompt.</p> <p>data – holds the data for the field which has data type, name, length, and so on.</p> <p>dictionary-display-order – The value for dictionary-display-order is the value of DefObjectDictionaries.DisplayOrder for the Dictionary associated with the DataElement associated with the Field.</p> <p>display-order – the value for display-order is the value of DefObjectDataHTML.DisplayOrder for the field.</p> <p>field-id – Field Id within the Request Center database.</p> <p>input-type – Input type of the field (for example, text, option, and so on).</p> <p>label – Label specified for the field.</p> <p>mandatory – The field data is mandatory in the service.</p> <p>max-length – Maximum length specified for the field.</p> <p>max-value – If it is a number range specified.</p> <p>min-value</p> <p>multi-select – Whether the input type is a multi-select box.</p> <p>options – The option list available for this data field.</p> <p>permission –</p> <p>validated – Should it be validated or not.</p>
user-id	

Agent Parameter



The agent parameter represents the external parameters specified for the agent. It has the Boolean attribute called multi-valued which is either true or false based on whether this parameter has multiple values selected by user. The name represents the name of the agent parameter and value represents its value.

Sample Inbound and Outbound Documents

task-started or task-cancelled (outgoing)

```

<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="18071221:1124919814742:-32752"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <task-started task-type="task">
    <task>
      <actual-duration>0.0</actual-duration>
      <calendar-entries>
        <calendar-entry>
          <calendar-entry-id>2</calendar-entry-id>
          <date>Thu Aug 25 17:00:00 PDT 2005</date>
          <end-time>Fri Aug 26 21:40:37 PDT 2005</end-time>
          <is-blocked>false</is-blocked>
          <is-break>false</is-break>
          <is-read>false</is-read>
          <person>
            <company-address/>
            <email>admin@company.com</email>
            <fax/>
            <first-name>admin</first-name>
            <home-ou>
              <name>&lt;s ID=&quot;847&quot; /&gt;</name>
              <organizational-unit-id>1</organizational-unit-id>
            </home-ou>
            <home-phone/>
            <last-name/>
            <login-name>admin</login-name>
            <person-id>1</person-id>
            <personal-address/>
            <supervisor-name/>
            <timezone>Pacific Standard Time</timezone>
            <work-phone/>
          </person>
          <sequence>0</sequence>
          <start-time>Thu Aug 25 21:40:37 PDT 2005</start-time>
          <subject>External Task</subject>
        </calendar-entry>
      </calendar-entries>
      <check-lists>
        <check-list-entry>
          <display-order>1</display-order>
          <is-mandatory>true</is-mandatory>
          <last-date/>
          <last-person/>
          <name>Make sure you wake up</name>
          <status>false</status>
        </check-list-entry>
        <check-list-entry>
          <display-order>2</display-order>
          <is-mandatory>true</is-mandatory>
          <last-date/>
          <last-person/>
          <name>Make sure you take a shower</name>
          <status>false</status>
        </check-list-entry>
        <check-list-entry>
          <display-order>3</display-order>

```



```

        <is-mandatory>true</is-mandatory>
        <last-date/>
        <last-person/>
        <name>Make sure you have breakfast</name>
        <status>>false</status>
    </check-list-entry>
</check-lists>
<completed-date/>
<context-id>1</context-id>
<context-type>Requisition Entry</context-type>
<due-date>Fri Aug 26 21:40:37 PDT 2005</due-date>
<effort>10.0</effort>
<estimated-date/>
<expected-duration>10.0</expected-duration>
<flag-id>0</flag-id>
<is-sharable>>false</is-sharable>
<is-shared>>false</is-shared>
<next-action-id>2</next-action-id>
<performer>
    <company-address/>
    <email>admin@company.com</email>
    <fax/>
    <first-name>admin</first-name>
    <home-ou>
        <name>&lt;s ID=&quot;847&quot;&/s>&/name>
        <organizational-unit-id>1</organizational-unit-id>
    </home-ou>
    <home-phone/>
    <last-name/>
    <login-name>admin</login-name>
    <person-id>1</person-id>
    <personal-address/>
    <supervisor-name/>
    <timezone>Pacific Standard Time</timezone>
    <work-phone/>
</performer>
<performer-actual-duration>0.0</performer-actual-duration>
<priority>2</priority>
<scheduled-start-date>Thu Aug 25 21:40:37 PDT 2005</scheduled-start-date>
<start-date>Wed Aug 24 21:42:15 PDT 2005</start-date>
<state-id>2</state-id>
<subject>External Task</subject>
<supervisor>
    <company-address>Foo Bar 25 Suite 300 Foo City CA 94404
USA</company-address>
    <email>internal@company.com</email>
    <fax/>
    <first-name>Monkey</first-name>
    <home-ou>
        <name>&lt;s ID=&quot;847&quot;&/s>&/name>
        <organizational-unit-id>1</organizational-unit-id>
    </home-ou>
    <home-phone/>
    <last-name>McBride</last-name>
    <login-name>monkey</login-name>
    <person-id>3</person-id>
    <personal-address>Fuchi Caca 16 Apartment C Fuchi Minn OR 78787
USA</personal-address>
    <supervisor-name/>
    <timezone>Pacific Standard Time</timezone>
    <work-phone/>
</supervisor>
<task-id>3</task-id>
<waiting>1</waiting>

```

```

</task>
<requisition>
  <actual-cost>0.0</actual-cost>
  <actual-duration>0.0</actual-duration>
  <closed-on/>
  <comments>
    <comment>
      <comment-date>Wed Aug 24 21:42:06 PDT 2005</comment-date>
      <comment-id>1</comment-id>
      <comment-text>I am adding a comment and I cannot think of a better
comment</comment-text>
      <component-id>3</component-id>
      <component-name>Request Center Component</component-name>
      <person>
        <company-address/>
        <email>admin@company.com</email>
        <fax/>
        <first-name>admin</first-name>
        <home-ou>
          <name>&lt;s ID=&quot;847&quot;&gt;&/name>
          <organizational-unit-id>1</organizational-unit-id>
        </home-ou>
        <home-phone/>
        <last-name/>
        <login-name>admin</login-name>
        <person-id>1</person-id>
        <personal-address/>
        <supervisor-name/>
        <timezone>Pacific Standard Time</timezone>
        <work-phone/>
      </person>
      <source-object-id>2</source-object-id>
      <source-object-inst-id>1</source-object-inst-id>
    </comment>
  </comments>
  <cost-center-code/>
  <customer>
    <company-address/>
    <email>admin@company.com</email>
    <fax/>
    <first-name>admin</first-name>
    <home-ou>
      <name>&lt;s ID=&quot;847&quot;&gt;&/name>
      <organizational-unit-id>1</organizational-unit-id>
    </home-ou>
    <home-phone/>
    <last-name/>
    <login-name>admin</login-name>
    <person-id>1</person-id>
    <personal-address/>
    <supervisor-name/>
    <timezone>Pacific Standard Time</timezone>
    <work-phone/>
  </customer>
  <due-on>Fri Aug 26 21:40:37 PDT 2005</due-on>
  <expected-cost>0.0</expected-cost>
  <expected-duration>0.0</expected-duration>
  <external>false</external>
  <initiator>
    <company-address/>
    <email>admin@company.com</email>
    <fax/>
    <first-name>admin</first-name>
    <home-ou>

```

```

        <name>&lt;s ID=&quot;847&quot;/&gt;</name>
        <organizational-unit-id>1</organizational-unit-id>
    </home-ou>
    <home-phone/>
    <last-name/>
    <login-name>admin</login-name>
    <person-id>1</person-id>
    <personal-address/>
    <supervisor-name/>
    <timezone>Pacific Standard Time</timezone>
    <work-phone/>
</initiator>
<invocations/>
<organizational-unit>
    <name>&lt;s ID=&quot;847&quot;/&gt;</name>
    <organizational-unit-id>1</organizational-unit-id>
</organizational-unit>
<requisition-entry>
    <closed-date/>
    <data-values>
        <data-value>
            <name>Requester</name>
            <value>John McGarzafi</value>
        </data-value>
        <data-value>
            <name>RemedyStuff.TicketID</name>
            <value>None yet</value>
        </data-value>
        <data-value>
            <name>RemedyStuff.AssetNumber</name>
            <value>123456789</value>
        </data-value>
    </data-values>
    <due-date>Fri Aug 26 21:40:37 PDT 2005</due-date>
    <item-number>1</item-number>
    <price-per-unit>0.0</price-per-unit>
    <priced>true</priced>
    <quantity>1</quantity>
    <rejected>false</rejected>
    <rejected-date/>
    <rejector>
        <company-address/>
        <email/>
        <fax/>
        <first-name/>
        <home-phone/>
        <last-name/>
        <login-name/>
        <person-id>0</person-id>
        <personal-address/>
        <supervisor-name/>
        <timezone/>
        <work-phone/>
    </rejector>
    <requisition-entry-id>1</requisition-entry-id>
    <revision-number>5</revision-number>
    <service>
        <dictionary>
            <data>
                <data-id>3</data-id>
                <data-type>Person</data-type>
                <is-dictionary-external>false</is-dictionary-external>
                <max-length>100</max-length>
                <name>Requester</name>
            </data>
        </dictionary>
    </service>

```

```

</data>
<dictionary-id>1</dictionary-id>
<is-external>>false</is-external>
<is-shared>>false</is-shared>
<name>Monkey Service (private)</name>
</dictionary>
<dictionary>
  <data>
    <data-id>1</data-id>
    <data-type>Text</data-type>
    <is-dictionary-external>>false</is-dictionary-external>
    <max-length>50</max-length>
    <name>TicketID</name>
  </data>
  <data>
    <data-id>2</data-id>
    <data-type>Text</data-type>
    <is-dictionary-external>>false</is-dictionary-external>
    <max-length>50</max-length>
    <name>AssetNumber</name>
  </data>
  <dictionary-id>2</dictionary-id>
  <is-external>>false</is-external>
  <is-shared>>true</is-shared>
  <name>RemedyStuff</name>
</dictionary>
<estimated-cost>0.0</estimated-cost>
<form>
  <fields>
    <field>
      <advanced-prompt/>
      <data>
        <data-id>2</data-id>
        <data-type>Text</data-type>
        <is-dictionary-external>>false</is-dictionary-external>
        <max-length>50</max-length>
        <name>AssetNumber</name>
      </data>
      <field-id>2</field-id>
      <input-type>text</input-type>
      <label>AssetNumber</label>
      <mandatory>>false</mandatory>
      <max-length>50</max-length>
      <max-value>0.0</max-value>
      <min-value>0.0</min-value>
      <multi-select>>false</multi-select>
      <options>
        <available-keys/>
        <available-labels/>
        <current-values/>
        <multivalued>>false</multivalued>
      </options>
      <permission>4</permission>
      <validated>>true</validated>
    </field>
    <field>
      <advanced-prompt/>
      <data>
        <data-id>1</data-id>
        <data-type>Text</data-type>
        <is-dictionary-external>>false</is-dictionary-external>
        <max-length>50</max-length>
        <name>TicketID</name>
      </data>

```

```

    <field-id>1</field-id>
    <input-type>text</input-type>
    <label>TicketID</label>
    <mandatory>>false</mandatory>
    <max-length>50</max-length>
    <max-value>0.0</max-value>
    <min-value>0.0</min-value>
    <multi-select>>false</multi-select>
    <options>
      <available-keys/>
      <available-labels/>
      <current-values/>
      <multivalued>>false</multivalued>
    </options>
    <permission>4</permission>
    <validated>>true</validated>
  </field>
  <field>
    <advanced-prompt>Give the name!</advanced-prompt>
    <data>
      <data-id>3</data-id>
      <data-type>Person</data-type>
      <is-dictionary-external>>false</is-dictionary-external>
      <max-length>100</max-length>
      <name>Requester</name>
    </data>
    <field-id>3</field-id>
    <input-type>text</input-type>
    <label>Requester Name</label>
    <mandatory>>false</mandatory>
    <max-length>100</max-length>
    <max-value>0.0</max-value>
    <min-value>0.0</min-value>
    <multi-select>>false</multi-select>
    <options>
      <available-keys/>
      <available-labels/>
      <current-values>
        <string>John McGarzafi</string>
      </current-values>
      <multivalued>>false</multivalued>
    </options>
    <permission>4</permission>
    <validated>>true</validated>
  </field>
</fields>
  <user-id>0</user-id>
</form>
<name>Monkey Service</name>
<parameters>
  <default-duration>0.0</default-duration>
  <priority>2</priority>
  <start-date/>
  <start-mode>0</start-mode>
</parameters>
<pricing-schema>0</pricing-schema>
<quantity>1</quantity>
<service-id>1</service-id>
<version>5</version>
</service>
<start-after/>
<start-date>Wed Aug 24 21:40:50 PDT 2005</start-date>
<start-mode>0</start-mode>
<status>1</status>

```

```

</requisition-entry>
<requisition-id>1</requisition-id>
<started-on>Wed Aug 24 21:40:32 PDT 2005</started-on>
<status>1</status>
<requisition-step>
  <completed-on/>
  <due-on>Fri Aug 26 21:40:37 PDT 2005</due-on>
  <estimated-on>Fri Aug 26 21:40:37 PDT 2005</estimated-on>
  <name>Delivery project for Monkey Service</name>
  <status>2</status>
</requisition-step>
</requisition>
<agent-parameter multi-valued="false">
  <name>Ticket</name>
  <value>None yet</value>
</agent-parameter>
<agent-parameter multi-valued="false">
  <name>Asset</name>
  <value>123456789</value>
</agent-parameter>
</task-started>
</message>

```

take-action (incoming)

```

<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="18071221:1124919814742:-32752">
  <take-action action="done"/>
</message>

```

send-parameters (incoming)

```

<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="18071221:1116468068789:-32360">
  <send-parameters>
    <agent-parameter>
      <name>Param1</name>
      <value>cat</value>
    </agent-parameter>
    <agent-parameter>
      <name>Param2</name>
      <value>catlitter</value>
    </agent-parameter>
  </send-parameters>
</message>

```

update-data (incoming)

```

<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="32580443:1116278520968:-32460">
  <update-data>
    <data-value multi-valued="false">
      <name>Name</name>
      <value>Rasesh Shah</value>
    </data-value>
    <data-value multi-valued="true">
      <name>TestDict1.foo</name>

```

```
        <value>value1</value>
        <value>value2</value>
    </data-value>
</update-data>
</message>
```

add-comments (incoming)

```
<?xml version="1.0" encoding="UTF-8"?>
<message channel-id="32580443:1116276793649:-32629">
    <add-comments>
        <comment>Test Comment</comment>
    </add-comments>
</message>
```




CHAPTER 4

Remedy Service Adapter

- [Overview, page 4-1](#)
- [Prerequisites, page 4-4](#)
- [BMC Remedy Configuration Steps \(Sample\), page 4-4](#)
- [Obtaining the Adapter, page 4-4](#)
- [Installing the Adapter, page 4-5](#)
- [Viewing the Adapter, page 4-5](#)
- [Configuring the Agent, page 4-5](#)
- [Configuring the Transformation, page 4-7](#)
- [Designing a Service for a Request Handled by the Remedy Adapter , page 4-11](#)
- [Test Scenario, page 4-12](#)
- [Log Messages, page 4-12](#)

Overview

The Cisco Service Adapter for BMC® Remedy® IT Service Management is a bidirectional integration between the two products. It allows you to provide a single customer portal within the Service Portal My Services module for both service requests and incident/change management. Prior to Release 9.3, it was licensed as a standard adapter for Service Portal. It is currently available from Advanced Services as a custom adapter.

With the Service Adapter for BMC Remedy IT Service Management, you can:

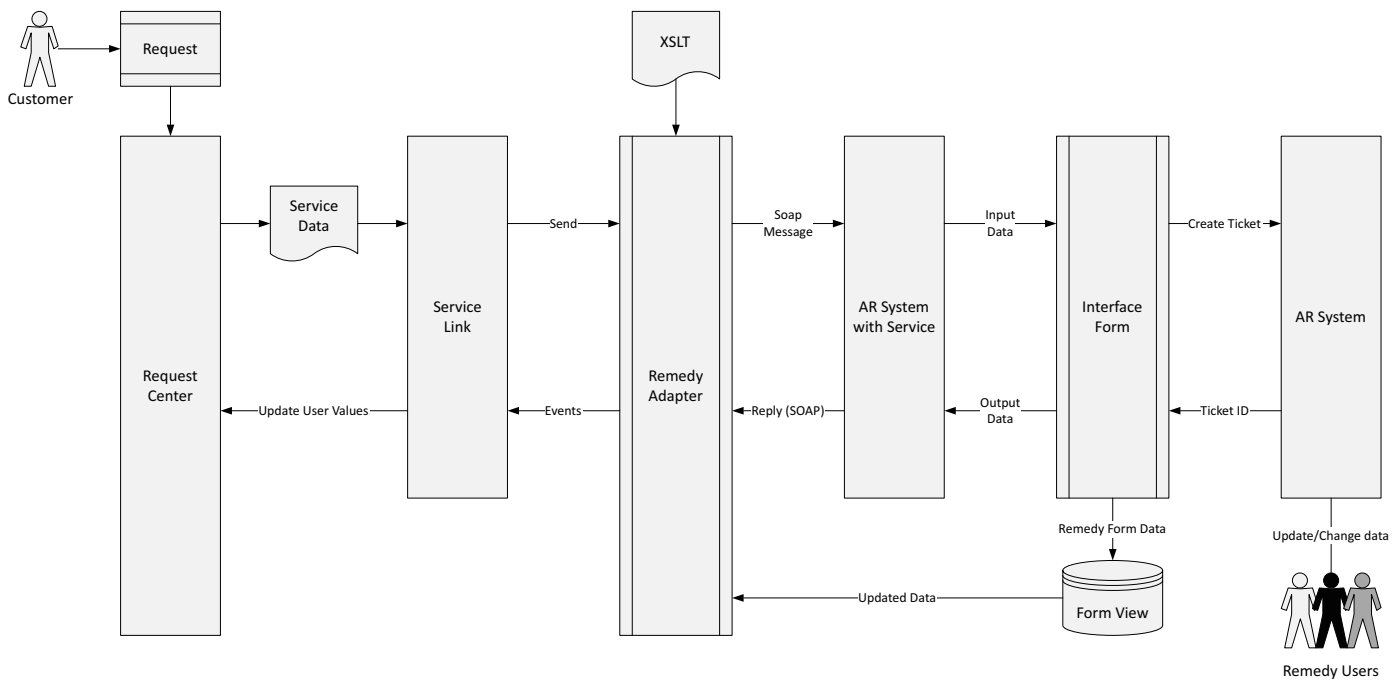
- Create one or more tickets at the task level in the Remedy system after customers create requests in My Services. One or more tasks within a service request process can be integrated with the Remedy system. As part of the fulfillment process for a requisition, a ticket would be created at the individual task level and sent to the Remedy system by the Remedy Adapter.
- Receive updates on the status of the Remedy ticket and other Remedy tasks. Once a ticket has been created in the AR System, corresponding workflow is executed. Upon successful completion of the workflow, the AR System sends back the status update to Service Portal. Alternately, Service Portal can poll the AR System for status updates.

The Service Link Adapter for BMC Remedy is completely noninvasive. No components must be installed into the AR System. A simple Remedy Interface form has to be created and a midtier configuration with Web Services needs to be enabled in the AR System for the Interface form.

Once the AR System midtier is configured for Web Services, Service Portal and Service Link communicate to the AR System with simple AR System client connections through the Web Services. Appropriate AR System licensing is required for access to the AR System Midtier Web Services. A workflow needs to be created in the Remedy System so that messages sent to the Interface Form through the Web Service create a ticket in the actual Remedy forms.

For the Service Link database access to the AR System RDBMS, a JDBC connection is required. Appropriate read permissions are required to the interface corresponding database view of the Interface form which are used to read information from the AR System to Service Portal.

The following diagram illustrates the system level integration between Service Portal and the AR System:



Integration Scenarios

Scenario 1: Request is created in Service Portal and submitted to the AR System for Fulfillment

- Step 1** The user submits a request in Service Portal.
- Step 2** Service Portal processes the workflow and identifies the target external application as the AR System, based on the request attributes.
- Service Portal posts a message on the agent-specific queue for Service Link to pick up and forward to the appropriate adapter. In this case, it is the Remedy adapter with SOAP Web Services capabilities built into it.
- Step 3** Service Link forwards the message to the Remedy adapter. The Remedy adapter takes the outbound document and applies an XSL transformation (XSLT) on the outbound document to create a SOAP-aware document. This adapter then uses the connection information contained in the Agent definition to make the appropriate communications calls to the Web Service as described in the WSDL document that was exported from the Remedy AR System using the Remedy Tools.

- The invocation of the web service is a single synchronous call. If the web service cannot be invoked, an error is returned to the Integration Server (Service Link).
 - The Integration Server will execute a preconfigured number of retries to call the web service. If all retries become exhausted, the task is pushed into a “Troubleshooting” state.
 - Once in a “Troubleshooting” state, the task can be sent again using manual processes provided by Service Link (“Send Manual Message”).
- Step 4** The server side of the web service is invoked by the client-side call executed by the SOAP message from the Remedy Adapter. This server side of the web service is within the Remedy AR System.
- Step 5** The Remedy AR System will process the request and return either a SOAP Fault or SOAP Success.
- Step 6** The Remedy Adapter has been waiting for the reply to the web service invoked, since it was configured with “ProcessResponse” set to true.
- If the response timeout has expired, an Error is returned to the Integration Server, which will retry the invocation, if retries remain. If the response timeout has not expired, the returning message is passed through another configured XSL transformation to create a Business Engine response message. The message updates information related to the task.
- Step 7** If the response contains a SOAP Success, there is at minimum a Ticket ID that has been returned by the AR System. This Ticket ID is used to update a corresponding field in the service data relevant to the task. If the response contains a SOAP Error, the Comment section of the task is updated with the Error information returned by the AR System.
-

Scenario 2: Receive status update from the AR System

- Step 1** An AR System user changes the status of an opened ticket, and the status change now needs to be forwarded to Service Portal.
- Step 2** The AR System workflow ensures that the changed data that is relevant to Service Portal is updated to the corresponding database table of the Interface form. The database view is created as part of form configuration in the Remedy system.
- Step 3** The Remedy Inbound Poller periodically polls the corresponding view in the AR database for new records that need to be processed into Service Portal. The newly updated records are fetched by comparing the last polled date and modified date of each record in the database view. The database view holds one row for each ticket created.
- Step 4** For each database record changed, the information from the database view is read and processed into a Business Engine message. The Business Engine will receive the message and update the task information appropriately.
-

Prerequisites

Service Portal Requirements

- Service Link module installed
- JDBC access to the BMC AR System database

BMC Requirements

- Remedy AR System 5.X or higher
- Remedy Mid Tier with Web Services enabled
- Supported Remedy database platforms: Oracle 10g or higher; SQL Server 2005 or higher

BMC Remedy Configuration Steps (Sample)

To configure BMC Remedy:

-
- Step 1** Create an Interface form using the out-of-box option which creates a corresponding table in the Remedy database. Add custom fields where required.
 - Step 2** Create a Web Service for that Interface form.
 - Step 3** Configure the mapping between the interface form fields and the Web Service fields for both input and output fields.
 - Step 4** Generate a WSDL for the Web Service.
 - Step 5** Create workflow which sends data from the interface form to the Remedy Ticketing form (Change Management form or Incident Management form). Use filters where required in the workflow for prefills (and so on) of the Remedy Ticketing form.
 - Step 6** Create workflow which passes the data back to the Interface form from the Remedy Ticketing System as defined in the Web Service outbound response. Usually it is status, priority or any other updates based on business requirements. Filters may also be used here.
 - Step 7** Test the Remedy workflow by making a request from the Interface form to check that the system is working before Service Portal can send out messages.
 - Step 8** Make sure the licensing of the midtier servlet engine supports multiple sessions.
-

Obtaining the Adapter

The Service Adapter for BMC Remedy was available as a packaged adapter prior to Release 9.3. Beginning from Release 9.3, Remedy Service Adapters are only available through Cisco Advanced Services.

Installing the Adapter

See the “[Deploying Adapters](#)” section on page 3-4 for instructions on how to install the custom adapters.

Viewing the Adapter

After installing the Service Adapter for BMC Remedy, to confirm that it has been successfully installed, check the following Service Link pages.

To view the installed adapter, start Service Portal, choose **Service Link**, choose **Manage Integrations > Adapters**, and then click **Remedy Adapter**. The adapter home page appears, as shown below.

The screenshot shows the Service Portal interface with the 'Adapters' tab selected. On the left, a tree view shows various adapters, with 'Remedy Adapter' highlighted. The main area displays the 'Manage Adapter' page for the 'Remedy Adapter'.

Manage Adapter	
Name	Remedy Adapter
Description	Remedy Adapter #version=10.1.0.002# #builddate=10-24-08#. Sends requests via HTTP or HTTPS connection to Remedy Webservice. Subsequent updates are fetched from a Remedy DB View.
Created On	11/17/2009
Created By	admin admin
Last Updated On	03/16/2010
Last Updated By	admin admin
Message Support	Outbound
Inbound Model	Poller
Inbound Class	com.newscale.is.adapter.remedy.RemedyInboundAdapter
Outbound Class	com.newscale.is.adapter.remedy.RemedyOutboundAdapter
Exception Class	
Polling Interval (in milliseconds)	<input type="text" value="30000"/>
Retry Interval (in milliseconds)	<input type="text" value="30000"/>
Maximum Attempts	<input type="text" value="0"/>

Configuring the Agent

As with any Service Link integration, an agent is the key component that configures the adapter and relates the service definitions and their data to the transformations and third-party system you wish to integrate with.

The Remedy Agent requires mandatory properties in order to function as well as meet the requirements for ticket creation within the Remedy system. The following table contains the set of required properties with default values for the Remedy Adapter. For more information on the usage of some of these properties, see [Chapter 2, “Service Link”](#).

Name	Description	Default Value
ContentType	Outbound SOAP message content type	text/xml
RoutingURL	SOAP URL where the message is routed to	Available from the Remedy WSDL
AcceptUntrustedURL	Indicates whether a security certificate is required for the target URL	false
TimeOut		18000
ProcessResponse	Indicates whether Service Link should accept the SOAP response from the web service	true
Username	Integrated Windows Authentication (IWA) username for the SOAP message to be authenticated by the web server where the web service is running	Only if IWA is enforced
Password	IWA user password	
Domain	IWA domain of the user	
Realm	NTLM Realm	If required when IWA is enforced
SOAPAction	SoapAction Header as defined in the WSDL with input map type as CreateInputMap	
JDBCUrl	JDBC URL to connect to the Remedy database to read updated data	for example, jdbc:newscale:sqlserver://localhost:1433;DatabaseName=ARSystem
JDBCDriverClass	Driver class to make the connection to the database	com.newscale.jdbc.UnifiedDriver
DBUserName	Database username	
DBPassword	Database password	
FormName	Interface form name or the corresponding DB view name	
FieldName_ModifiedDate	Column name in the view that holds the modified date of the Remedy ticket	
FieldName_CaseID	Column name in the view that holds the Remedy ticket ID	
FieldName_Status	Column name in the view that holds the status data	

In addition to the required properties, you should add additional agent parameters that you will collect from the service form and want to send to Remedy for display on the Remedy ticket. These are customer-specific values that are important for your business process and requirements.

The following figures show an example of a configured Remedy Agent containing the mandatory values.

Outbound Properties

Configure Outbound Properties	
Name	Value
RemedyOutboundAdapter.ContentType	text/xml
RemedyOutboundAdapter.RoutingURL	http://spartana.oakqas.celosis.com/arsys/services/ARService?server=s...
RemedyOutboundAdapter.AcceptUntrustedURL	true
RemedyOutboundAdapter.TimeOut	18000
RemedyOutboundAdapter.ProcessResponse	true
RemedyOutboundAdapter.Username	Demo
RemedyOutboundAdapter.Password	*****
RemedyOutboundAdapter.Domain	
RemedyOutboundAdapter.Realm	
RemedyOutboundAdapter.SOAPAction	OpCreate

Inbound Properties

Configure Inbound Properties	
Name	Value
RemedyInboundAdapter.JDBCUrl	jdbc:newscale:sqlserver://spartana.oakqas.celosis.com:1433;Database...
RemedyInboundAdapter.JDBCDriverClass	com.newscale.jdbc.UnifiedDriver
RemedyInboundAdapter.DBUserName	ARAdmin
RemedyInboundAdapter.DBPassword	*****
RemedyInboundAdapter.FormName	SimpleForm
RemedyInboundAdapter.FormView	
RemedyInboundAdapter.FieldName_ModifiedDate	Modified_Date
RemedyInboundAdapter.FieldName_CaseID	ns1:Request_ID
RemedyInboundAdapter.FieldName_Status	Status

The Service Designer Integration Wizard partially automates creating agent definitions for web services-based agents. In addition, Cisco Advanced Services has tools that can assist customers in creating Remedy Agent definitions. Please contact Cisco to inquire into the availability.

Configuring the Transformation

The Service Link outbound and inbound transformations define the XSLT that converts the raw Service Portal data into the data that the Remedy WSDL can consume, and the XSLT that converts the response from Remedy back to Cisco messages. After configuring the agent and understanding the properties passed between the two systems, the following steps are required to create the transformation:

- Step 1** Author the XLST in the editor of your choice.
- Step 2** Create the transformations in Service Link by choosing **Manage Integrations > Transformations**.
- Step 3** Add a transformation and set the direction to “Outbound”. Enter the XLST for converting nsXML to Remedy SOAP request on the **Request** subtab.

- Step 4** Add another transformation and set the direction to “Inbound”. Enter the XSLT for converting Remedy SOAP response to nsXML on the **Request** subtab.

The following sections describe sample XSLTs that transform Service Portal data into the basic required fields for Remedy ticket creation.

Inbound Transformation Details

A sample inbound transformation is shown below.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<xsl:template match="/">
<xsl:apply-templates select="inbound-results" />
<xsl:apply-templates select="soapenv:Envelope" />
</xsl:template>
<xsl:template match="inbound-results">
<message>
<xsl:attribute name="channel-id">
<xsl:value-of select="row/column[@name='sl_channelid']" />
</xsl:attribute>
<send-parameters>
<xsl:if test="row/column[@name='request_id']!= '' and
row/column[@name='request_id']!= 'null' ">
<agent-parameter>
<name>Request_ID</name>
<value><xsl:value-of select="row/column[@name='request_id']" /></value>
</agent-parameter>
</xsl:if>
<xsl:if test="row/column[@name='short_description']!= '' and
row/column[@name='short_description']!= 'null' ">
<agent-parameter>
<name>Short_Description</name>
<value><xsl:value-of select="row/column[@name='short_description']" /></value>
</agent-parameter>
</xsl:if>
</send-parameters>
<xsl:if test="row/column[@name='status']='4' or row/column[@name='status']='3'">
<take-action action="done" />
</xsl:if>
</message>
</xsl:template>
<xsl:template match="soapenv:Envelope">
<message>
<xsl:attribute name="channel-id">
<xsl:value-of select="//*[local-name()='SL_CHANNELID']" />
</xsl:attribute>
<send-parameters>
<xsl:if test="//*[local-name()='Request_ID']!= ''">
<agent-parameter>
<name>Request_ID</name>
<value><xsl:value-of select="//*[local-name()='Request_ID']" /></value>
</agent-parameter>
</xsl:if>
</send-parameters>
</message>
</xsl:template>
</xsl:stylesheet>
```


Outbound Transformation Details

A sample outbound request transformation is shown below.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template name="main" match="/message/task-started">
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Header soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<AuthenticationInfo>
<userName>Demo</userName>
<password></password>
</AuthenticationInfo>
</soap:Header>
<soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<OpCreate>
<Assigned_To><xsl:value-of
select="/message/task-started/agent-parameter[name='Assigned_To']/value" /></Assigned_To>
<Short_Description><xsl:value-of
select="/message/task-started/agent-parameter[name='Short_Description']/value"
/></Short_Description>
<Status><xsl:value-of select="/message/task-started/agent-parameter[name='Status']/value"
/></Status>
<Submitter><xsl:value-of
select="/message/task-started/agent-parameter[name='Submitter']/value" /></Submitter>
</OpCreate>
</soap:Body>
</soap:Envelope>
</xsl:template>
<xsl:template name="taskcancelled" match="/message/task-canceled">
</xsl:template>
<xsl:template name="date_format">
<xsl:param name="text"/>
<!-- Break up the string into components -->
<xsl:variable name="date" select="substring-before($text, ' ')" />
<xsl:variable name="time" select="substring-after($text, ' ')" />
<xsl:variable name="hour" select="substring-before($time, ':)" />
<xsl:variable name="minute" select="substring-after($time, ':)" />
<!-- Reassemble the components applying rules -->
<xsl:value-of select="$date" />
<!-- Add a T between date and time-->
<xsl:text>T</xsl:text>
<xsl:if test="not(string-length($hour)=2)">
<!-- Add leading 0 to hour if needed -->
<xsl:text>0</xsl:text>
</xsl:if>
<xsl:value-of select="$hour" />
<xsl:text>:</xsl:text>
<xsl:value-of select="$minute" />
<!-- Add seconds -->
<xsl:text>:00</xsl:text>
</xsl:template>
</xsl:stylesheet>
```

The Service Designer Integration Wizard can assist customers in creating Remedy transformations.

Outbound and Inbound Date Format Transformations

The date format has to be set to international date/UTC in the outbound and inbound transformations for both outbound and inbound messages. Further, the date format needs to be converted from seconds in the database view, as described in the [“Converting the Date in the Remedy Interface Form Database View”](#) section on page 4-11.

The following table describes the required international date formats for the Remedy and Service Portal systems.

System	International Date Format
Remedy	YYYY-MM-DDThh:mm:ssZ Any date that is sent to Remedy has to be in this format. The “Z” is appended at the end informs Remedy that this is a UTC date.
Service Portal	YYYY-MM-DDThh:mm:ss Any date sent to Service Portal from Service Link must be in this international format. Service Portal assumes the date to be in UTC, avoiding the need for the trailing “Z.”

Transformation XSL template

Outbound Transformation

The transformation may not already have a template that converts the date value to international date format. “Z” is appended to the new date value in the transformation as shown below:

```
<xsl:template name="intl_date_format">
<xsl:param name="remedy_date"/>
<!-- Break up the string into components -->
<xsl:variable name="date" select="substring-before($remedy_date, ' ')/>
<xsl:variable name="time" select="substring-after($remedy_date, ' ')/>
<xsl:variable name="hour" select="substring-before($time, ':')"/>
<xsl:variable name="minute" select="substring-after($time, ':')"/>
<!-- Reassemble the components applying rules -->
<xsl:value-of select="$date"/>
<!-- Add a T between date and time-->
<xsl:text>T</xsl:text>
<xsl:if test="not(string-length($hour)=2)">
<!-- Add leading 0 to hour if needed -->
<xsl:text>0</xsl:text>
</xsl:if>
<xsl:value-of select="$hour"/>
<xsl:text>:</xsl:text>
<xsl:value-of select="$minute"/>
<!-- Add seconds -->
<xsl:text>:00Z</xsl:text>
</xsl:template>
```

This template may be used as follows:

```
<xsl:call-template name="date_format"><xsl:with-param name="remedy_date"
select="/message/task-started/agent-parameter[name='Start_Date']/value" />
```

Inbound Transformation

The transformation may not already have a template that converts the date value to international date format. The template converts the date value that comes from Remedy inbound message.

```
<xsl:template name="intl_date_format">
<xsl:param name="remedy_date"/>
<xsl:value-of select="translate(substring-before($remedy_date, '.'), ' ', 'T')"/>
</xsl:template>
```

This template may be used as follows:

```
<xsl:call-template name="intl_date_format"><xsl:with-param name="remedy_date"
select="row/column[@name='end_date']" /></xsl:call-template>
```

Converting the Date in the Remedy Interface Form Database View

You must customize the Remedy DB View in the Remedy database to which the Service Link Remedy Agent connects. The column that holds the date value in the DB view is stored in seconds rather than as a date value. This has to be converted to a date value in the view's SQL, as described in the following table:

RDBMS	SQL Statement
Microsoft SQL Server	DATEADD(s, DBCOLUMN, '1970-01-01')
Oracle	(TO_DATE('01-JAN-1970') + DBCOLUMN / 86400)

Designing a Service for a Request Handled by the Remedy Adapter

Now that you have all of the Service Link components created, you are ready to build a service in Service Designer that uses the agent. These steps can be automated by using the Service Designer Integration Wizard. See [Chapter 2, "Service Link"](#) for detailed information about creating web services-based Service Link agents.

-
- Step 1** Create a dictionary in Service Designer that will be used to collect the information to be sent and received from Remedy.
 - Step 2** Create an Active Form Component (AFC) in Service Designer that includes the Remedy dictionary created in the previous step.
 - Step 3** Create a service in Service Designer that you will use for the Remedy adapter activity.
 - Step 4** Choose the service you just created, and then click the **Forms** tab, and add the Active Form Component to the service.
 - Step 5** Click the **Plan** tab, and then create an external task that will orchestrate the Remedy workflow.
 - Step 6** Change any of the agent value mappings as required in the Service Definition using the task agent mapping popup.
 - Step 7** Save the service.
-

Test Scenario

Now that all of your Remedy Adapter components have been created and you have bound them to a Service Definition, you are ready to test your integration. The following describes simple steps to confirm that your integration is functional:

-
- Step 1** Create a requisition in Service Portal for the service associated to the Remedy Agent.
 - Step 2** Verify the receipt of the Execute Task message in Service Link corresponding to the requisition. This is the outbound SOAP request from Service Portal to Remedy, containing the fields as defined in the outbound agent parameter mapping.
 - Step 3** Verify the receipt of the Send Parameters HTTP response message in Service Link. This is the SOAP response of the Remedy web service. The message is processed in Service Portal only when the ProcessResponse flag in the Remedy Agent is set to “True”.
 - Step 4** Verify the corresponding request in the Remedy form. The Remedy ticket should contain the matching information included in the SOAP request.
 - Step 5** Make changes to the Remedy form. Modify the form field values and the ticket status.
 - Step 6** Verify the receipt of the Send Parameters messages in Service Link. The corresponding requisition entry should have the form data updated based on the inbound agent parameter mapping.
 - Step 7** Verify the receipt of the Take Action (or Composite) message in Service Link when the status in Remedy is set to Resolved, Closed, or Rejected. The take-action operation generated through the inbound transformation sets the external task status to Completed.
-

Log Messages

Message	Description	Recommended Action
INFO Message	Information about the tasks being performed.	Follow the information in the log.
DEBUG Message	Information about the tasks being performed to enable debugging.	Follow the debug message for the details of a certain task.
Message Routing Exception	An error in routing of the message.	Verify the Routing URL is correct. Open the URL in a browser and a Remedy screen should appear.
Internal System Error	Error in the Remedy midtier or Remedy System.	Check for a SOAP response message which indicates the error messages from the midtier.



CHAPTER 5

Web Services

- [Overview, page 5-1](#)
- [Prerequisites for Web Services, page 5-2](#)
- [Web Services for Request Management, page 5-5](#)
- [Web Services for Task Management, page 5-16](#)
- [Web Services for Portfolio Management, page 5-19](#)
- [Sample Requests and Responses, page 5-21](#)
- [Web Services Error Messages, page 5-40](#)

Overview

This chapter documents the use of web services for Service Portal. These include services which implement the Requisition API (RAPI 2), an API which allows an external system to create and manage service requests within Request Center. The web services include additional requests, to allow the management of delivery and authorization tasks within a service request; and to review the contents of the Request Center service catalog or the Demand Center service portfolio.

Audience

The intended audience for this chapter comprises programmers and designers who are responsible for implementing a RAPI interface between Request Center and an external application. Knowledge of the following methodologies and technologies are helpful in understanding and making best use of this documentation:

- Service design and configuration in Request Center
- Web Services, including SOAP messages and Web Service Descriptive Language (WSDL)

Web Services

Web Services provide a means for an external application to create or update requisitions (service requests) and tasks which comprise those requisitions. The external application may perform all or part of the request fulfillment process, starting from the submission of the service request to a Service Portal

installation; incorporating any approval or review tasks; and performing any service delivery tasks. The external application does this by sending a SOAP message to Service Portal and processing the response received.

Prerequisites for Web Services

Detailed prerequisites for developing and implementing web services are given below. These include

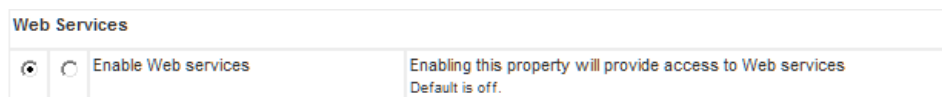
- A Service Portal installation, configured to support web services.
- An environment for developing a client to submit the web service requests and receive the responses.
- An environment for testing the code.

Service Portal Installation and Configuration

Service Portal must be configured to support web services.

Administration Settings

The global setting “Enable Web services” in **Administration > Settings** needs to be turned on. This is a global switch for all the web services in the system. If this global setting is off, no web service in Request Center is accessible. By default, this setting is turned off.



WSDLs

To validate any request developed, the web services WSDLs must be available. The WSDLs can be found at:

<http://<ServerName>/RequestCenter/webservices/wSDL/>

Available WSDLs are summarized in the table below.

WSDL	Contents
AuthenticationService.wsdl	A request to authenticate the specified user to Service Portal.
RequisitionService.wsdl	Requests to submit a requisition, cancel a requisition, or get its status.
ServiceCatalog.wsdl	For internal use only.
ServiceManagerTaskService.wsdl	Requests to approve or reject an authorization or to signify a review has been performed.
ServicePortfolio.wsdl	Requests to manage Demand Center service offerings and associated costs.

Roles and Capabilities

The web services can be accessed by users who have a role which includes appropriate capabilities for the Web Services module. No prebuilt roles include these capabilities, so administrators will need to use Organization Designer to create one or more custom roles. Once the role is created, you can add Web Services capabilities:

The screenshot shows the Organization Designer interface. At the top, there is a header with the user name [admin admin], Profile, Logout, and Organization Designer. Below the header are tabs for Functional Positions and Roles. The main content area is titled 'Capabilities' and includes a checkbox for 'Show inherited capabilities'. Below this is a table with columns for 'Module' and 'Capability'. The table lists five capabilities for the 'Service Manager' module: 'Cancel Ongoing Requisitions', 'Create Ad Hoc Tasks', 'Manage Work', 'Perform Global Delivery Search', and 'Perform Work'. There are 'Add' and 'Remove' buttons below the table. Below the table is a section titled 'Add System Capability' with a 'Choose Module' dropdown set to 'Web Services' and a 'Choose Capability' section with six checkboxes: 'Service Catalog Access', 'Financial Management Access', 'Requisition Access', 'Requisition System Account', 'Task Access', and 'Task System Account'. There are 'Add' and 'Cancel' buttons at the bottom of this section. On the right side, there is a sidebar with a navigation menu containing 'General', 'Members', 'Capabilities', 'Permissions', and 'Administration', with 'Capabilities' selected.

The web services capabilities are:

- **Requisition Access:** users having this capability alone can access the RequisitionService web service requests for themselves. The authenticated user and the initiator will have to be the same. If not, an appropriate fault response is thrown.
- **Requisition System Account:** users having this capability can access the RequisitionService web service requests for themselves as well as anybody else. The authenticated user and the initiator can be different.
- **Task Access:** users having this capability alone can access the ServiceManagerTaskService web service requests for themselves.

- **Task System Account:** users having this capability can access the ServiceManagerTaskService web service requests for themselves as well as anybody else. The authenticated user and the initiator can be different.
- **Service Catalog Access:** users having this capability can access the Service Catalog web service requests.
- **Financial Management Access:** users having this capability can access the Service Portfolio web service requests.

Testing and Development Environment

The examples in this chapter were developed using soapUI, a tool for developing and testing web services. A free version of this tool is available from the soapUI website. A professional version is also available. The steps outlined below may vary depending on the version of the tool you use.

Once soapUI is downloaded and installed, you may create a workspace for developing the web services. You can create a project to include the WSDLs that will be used to construct sample requests and to validate requests sent to Service Portal. An initial WSDL is required when a project is created. More WSDLs may be added to the same project later on. Keeping the **Create Requests** option checked during project creation will allow sample requests to be generated.

Generating Code

The client for the web services can be coded with tools like CXF or Axis from Apache.

Generating Client Code using Axis 2

Detailed instructions and user guide for generating web service client using Axis 2 can be found in the Apache website.

Here are the high-level steps for creating the axis2 client using soapUI:

-
- Step 1** Download the Axis 2 library.
 - Step 2** Set the Axis 2 library location in the soapUI Preferences menu.
 - Step 3** Generate the client code by going to **Tools > Axis 2 Artifacts**.
-

When generating the client code, you should choose **adb**, the Axis default binding, as the databinding method. You should also generate a test case option.

The client code is generated. Method stubs are created in the test case. You will need to populate the objects properly.

Generating Client Code using Apache CXF

Instructions for generating web service client using CXF can also be found in the Apache website.

Here are the steps needed to create a CXF client using soapUI:

-
- Step 1** Download the Apache CXF library.
- Step 2** Set the CXF library location in soapUI Preferences menu.
- Step 3** Generate the client code by going to **Tools > Apache CXF**.
-

The client code is generated. The class of interest is:

RequisitionServicePortType_RequisitionServiceHttpPort_Client.java

This class has a main method and all the operations defined in the WSDL can be invoked from here. The code for invoking these operations will already be present. All that needs to be done is to populate the various variables needed.

Method stubs are created. All that is needed is to populate the objects properly.

Web Services for Request Management

Overview

The operations that can be performed via RAPI 2 request management are summarized in the table below:

Request	Description
addComment	Add a comment to an open requisition.
cancelRequisition	Cancel an open requisition, including all service requests in the requisition.
cancelRequisitionEntry	Cancel a service request. If this is the last service request in the requisition, cancel the requisition.
getOpenRequisitions	Get a list of all open requisitions.
getRequisitions	Get a list of open requisitions, optionally restricting the contents of the list.
getRequisitionStatus	Get the status of the specified requisition.
getServiceDefinition	Get the definition of the service for which a requisition is to be entered.
submitRequisition	Submit a new requisition.

Sample Service Definition

The samples given in this chapter show the XML required for submitting a request for the New Standard Laptop Computer service. The service form for ordering this service looks like the following when ordered by an administrative user. (For a nonadministrative user, the New Laptop dictionary at the top of the form would not be visible.)

Laptop Details

*	Laptop Type	---	Select the type of laptop you need from the drop-down list.
*	Manufacturer	<input type="text"/>	
	Model	<input type="text"/>	
	Disk Space (GB)	<input type="text"/>	
	Memory (GB)	<input type="text"/>	
	Price	<input type="text"/>	

Customer

First Name	<input type="text" value="First Name"/>
Last Name	<input type="text" value="Last Name"/>
Personal_Identification	<input type="text"/>
Email Address	<input type="text" value="firstname.lastname@newscale.com"/>
Department Name	<input type="text" value="IT Services"/>
State	<input type="text" value="Virginia"/>

Other Work Site

Will work be performed at the customer location?	<input checked="" type="radio"/> Yes <input type="radio"/> No
--	---

Customer Work Site

Street	<input type="text" value="PO Box 101207"/>
OfficeCubeRoom	<input type="text"/>
City	<input type="text" value="Arlington"/>
State	<input type="text" value="Virginia"/>
Zip Code	<input type="text" value="22210"/>

Authentication

Any web service exposed by Request Center needs to be authenticated. Unauthenticated web service calls need to be intercepted and stopped.

Authentication via web services in Request Center can be done in the following ways:

- Authenticate per session
- Authenticate per request

Nonauthenticated users cannot make any successful web services call. If the global setting “Enable Web Services” is turned off, no web service in Request Center is accessible. By default, this setting is turned off.

Authenticate per Session

In this approach the user first makes an Authentication web service call and authenticates the user. The server then establishes a session for this user. As long as this session is valid, this user can make additional web service calls. The authenticate per session request is included in the AuthenticationService WSDL.

The authenticate request has the following format:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:aut="http://authentication.api.newscale.com">
  <soapenv:Header/>
  <soapenv:Body>
    <aut:authenticate>
      <aut:userName?</aut:userName>
      <aut:password?</aut:password>
    </aut:authenticate>
  </soapenv:Body>
</soapenv:Envelope>
```

Authenticate per Request

In this approach, there is no separate call to the authentication web service. Instead the user sends the authentication information in the SOAP header as part of each web service call. The Authentication handler for the web service in Request Center checks whether the user is authenticated. If no session has been established for this particular user, this handler retrieves the authentication information from the SOAP header. If the authentication information is present, this handler tries to authenticate the user. If the authentication information is missing or is not valid, this handler throws an exception to the client with the proper error code and error message.

Encryption

The password specified in the SOAP header cannot be encrypted. The password can be made secure by using SSL.

RBAC Check for Web Service Access

Each web service exposed in Request Center has an associated system capability. The authentication handler also checks to see whether the specified user can access (or execute) the web service. If the user has the appropriate system capability, the user is allowed to proceed further. Otherwise, an exception is thrown to the client with the proper error code and message.

Interaction of SOAP Authentication with Directory Integration

If Directory Integration is not enabled, the user specified must exist in the personnel directory before the SOAP request is issued.

If Directory Integration is enabled and the Login event includes an Import Person operation, an external directory is consulted to retrieve the person's profile, and that information is inserted into the personnel directory. In taking this approach, the directory information must include a role granting appropriate web services capabilities, or such a role must have been previously assigned to the business unit (or service teams) of which the user is a member. Consequently, it is recommended that prospective SOAP accounts be prepopulated in the database and assigned appropriate privileges before these accounts submit requests.

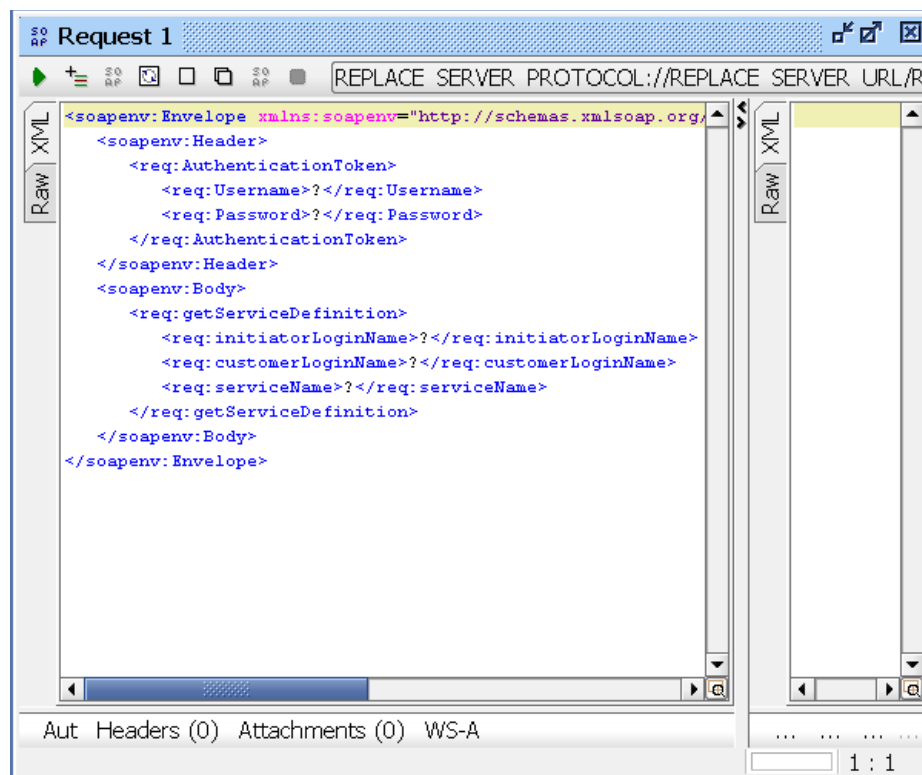
Getting the Service Definition

The `getServiceDefinition` request returns metadata describing the specified service. This metadata is required to submit a request. The use of this operation for services that include grid dictionaries is not supported in this release. An error is returned when the operation is invoked against such services.

getServiceDefinition Request

The request specifies the name of the service whose definition is needed.


In soapUI, right-click the sample request (Request1) under the `getServiceDefinition` node, then click **Show Request Editor**. The request appears, as it was generated. A question mark (?) indicates all XML elements where a value is expected.



You must supply an endpoint for the SOAP request. If you consult the properties of this request (and the menu bar above), you will see that the Endpoint has not yet been defined. Replace this with the endpoint for the RAPI 2 services:

`http://<ServerName>/RequestCenter/services/RequisitionService`

where **RequisitionService** is the wsdl name.

You can then copy the request, using the **Creates a copy of this request** icon () in the menu bar of the Request Editor, leaving the prototype request for reference. In your copy, replace the question marks, supplying authentication criteria, as well as the initiator and customer login names and the name of the service you are interested in:

```


<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:req="http://requisition.api.newscale.com">
  <soapenv:Header>
    <req:AuthenticationToken>
      <req:Username>
      <req:Password>
    </req:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
    <req:getServiceDefinition>
      <req:initiatorLoginName>
      <req:customerLoginName>
      <req:serviceName>
    </req:getServiceDefinition>
  </soapenv:Body>
</soapenv:Envelope>
  
```

```

<req:AuthenticationToken>
  <req:Username>admin</req:Username>
  <req>Password>admin</req>Password>
</req:AuthenticationToken>
</soapenv:Header>
<soapenv:Body>
  <req:getServiceDefinition>
    <req:initiatorLoginName>admin</req:initiatorLoginName>
    <req:customerLoginName>mathurston</req:customerLoginName>
    <req:serviceName>New Standard Laptop Computer</req:serviceName>
  </req:getServiceDefinition>
</soapenv:Body>
</soapenv:Envelope>

```

getServiceDefinition Response

To submit the getServiceDefinition request, click the Submit request to specified URL button () at the top left of the Request Editor window. The response appears within the Request Editor, to the right of the request.

The response to getServiceDefinition returns the metadata that describes the service, as summarized in the table below:

XML Element (with document hierarchy)	Description
Service	
name	Name of the service
pricingmodel	
quantity	Quantity of service to be ordered
version	The version number of the service
Dictionaries >	
Dictionary	Each service contains one or more dictionaries
name	Name of the dictionary
readable	True if the is dictionary readable as per the Access Control in the Service Designer Active form component for the ordering moment; false otherwise
writable	True if the dictionary editable as per the Access Control in the Service Designer Active form component for the ordering moment; false otherwise
Fields >	
DictionaryField	Each dictionary contains one or more fields
canSelectMultiple	Can multiple values be selected for this field?
defaultValue	The default value of the field
fieldDataType	The data type of the field (numeric, date, and so on)
fieldName	The name of the field
inputType	The html input type of the field
label	The label of the field
mandatory	True if the field is mandatory; false otherwise

XML Element (with document hierarchy)	Description
maxLength	Maximum length of the field
selectableValues	Selectable values for the field

Each dictionary is described, as well as each field within the dictionary. The access control specified for the dictionary in the ordering moment is critical for writing a well-formed submitRequisition request. Only those dictionaries which are readable or writable by the customer in the ordering moment are included in the response and need to be included in the submitRequisition request.

```
<name>Customer_Information</name>
<readable>true</readable>
<writable>true</writable>
</Dictionary>
```

The complete getServiceDefinitionResponse for the “New Standard Laptop Computer” is given in the [“Sample Requests and Responses”](#) section on page 5-21.

Submitting a Requisition

Technically speaking, it is not required to perform a getServiceDefinition request before sending a submitRequisition request. However, the getServiceDefinition returns information that is critical to formulating a valid submitRequisition message for the current version of the service.

- The current version of the service is required. The version number is incremented whenever the service definition itself or any of the included Active Form Components or dictionaries is updated.
- The getServiceDefinition request specifies which fields are mandatory; the submit request can be certain to include data for all mandatory fields.
- All mandatory dictionaries and fields must be listed in the submit request. The dictionaries, or the fields within the respective dictionaries, may appear in any order.
- The getServiceDefinition request also returns default values assigned to any fields, included resolved lightweight namespaces for Customer and Initiator information. These values are typically mandatory and need to be supplied in the submitRequisition request.
- The getServiceDefinition request can be used to submit a request for a service whose definition includes fields with options (single-select, multi-select, and radio buttons) when those options are defined using the Active Form Component's Display Options (HTML Representation) pages. When the options are specified via a data retrieval rule, the service request can be submitted; however, it is the responsibility of the submitting program to ensure that the value for the field is a valid option.

The submitRequisition request basically bypasses the ordering moment which occurs when a request is submitted via My Services. No conditional rules, data retrieval rules, or ISF is executed in conjunction with the submitted request. Therefore, if these facilities are used to provide values for dictionary fields or to perform validations, an alternate means must be found of providing these values. The use of this operation for services that include grid dictionaries is not supported in this release. An error is returned when the operation is invoked against such services.

submitRequisition Request

Any dictionary viewable or editable in the ordering moment must be included as a <section> node in the submitrequisition request. All mandatory fields and their values must be specified. No value need be included for the optional fields (but be sure to remove the question marks inserted by soapUI). The order of the dictionaries and the order of the fields does not have to match the order in the service definition but the fields have to appear under the correct dictionary node.

XML Element (and document hierarchy)	Description
initiatorLoginName	The initiator's login name
customerLoginName	The customer's login name
serviceRequests > ServiceRequest	There can be multiple service requests.
name	The name of the service
quantity	Quantity of services to be ordered
version	The version of the service
Sections > Section	Each service can have multiple dictionaries (sections).
name	The name of the dictionary
Fields > Field	Each dictionary can have multiple fields.
name	The name of the field
value > string	The value to be set for this field

For example, XML setting the value of the ZipCode field in the dictionary RC_ServiceLocation to be "07201" would look like this:

```
<req:Section>
.
.
.
  <req:fields>
.
.
.
    <req:Field>
      <req:name>ZipCode</req:name>
      <req:value>
        <req:string 07201/>
      </req:value>
    </req:Field>
  </req:fields>
  <req:name>RC_ServiceLocation</req:name>
</req:Section>
```

submitrequisition Response

If the request to submit the requisition succeeds, the response will include the requisition ID of the created request, as well as several other attributes of the request.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <ns1:submitRequisitionResponse xmlns:ns1="http://requisition.api.newscale.com">
      <ns1:submitRequisitionResult ns1:customer="admin admin"
        ns1:dueDate="2009-05-08T16:14:26.267-07:00"
        ns1:requisitionId="186"
      >
```

```

        ns1:initiator="admin admin"
        ns1:startDate="2009-04-30T18:14:26.110-07:00"
        ns1:status="Ongoing"/>
    </ns1:submitRequisitionResponse>
</soap:Body>
</soap:Envelope>

```

The attributes of the submitRequisitionResult response are summarized in the table below:

XML Element	Description
submitRequisitionResponse > submitRequisitionResult	The response will contain as many entries as there are services in the service request
customer	The customer name for the requisition
dueDate	The due date for the requisition
requisitionId	The requisition ID for the requisition
initiator	The initiator for the requisition
startDate	The date the requisition was started
status	The status of the requisition

If the request fails, an error message is returned. Possible errors are shown in Appendix B: RAPI Error Messages. The error message is always in the format of a “SOAP fault”, as shown in the sample below:

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>The version specified in the request does not match the version in the
database for service 'New Standard Laptop Computer'. Please get the latest service
definition.</faultstring>
      <detail>
        <RequisitionFault xmlns="http://requisition.api.newscale.com">
          <errorCode>REQ_0018</errorCode>
          <errorMessage>The version specified in the request does not match the version in
the database for service 'New Standard Laptop Computer'. Please get the latest service
definition.</errorMessage>
        </RequisitionFault>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>

```

Getting a List of Requisitions

The getRequisitions and getOpenRequisitions operations return information about open requisitions. They differ in the arguments that can be included in the request.

These operations might be useful in managing requisitions. For example, a list of open requisitions might be returned, and those of a particular type (for a particular service) whose past due date exceeds some user-defined threshold may be noted.

getOpenRequisitions Request

getOpenRequisitions returns all open requisitions, up to a specific maximum number of requisitions. The requisitions are returned in descending order by Requisition ID. This request is supported only for backward compatibility of certain retired Request Center integration points and should not be used in web services.

getRequisitions Request

getRequisitions returns all requisitions, up to a specific maximum number of requisitions. It also allows you to specify the view type and status of the requisitions to be returned. This request is supported only for backward compatibility of certain retired Request Center integration points and should not be used in web services.

Getting the Requisition Status

The getRequisitionStatus operation returns information on the authorizations and task plan status for the specified requisition. The level of detail is similar to that shown to the My Services user, when he/she views the delivery plan:

Delivery Process				
	Process Milestone	Due Date	Completed On	Status
✓	Service Group Review	12/07/2011 10:00 AM	12/07/2011 3:06 AM	Completed
✓	Service Group Authorization	12/07/2011 11:00 AM	12/07/2011 3:07 AM	Completed
	Delivery project for Testemail_Order_Mobile Device_Service_at_doorstep	12/08/2011 11:00 AM		In Progress

getRequisitionStatus Request

The request returns information on the current status of a requisition.

XML Element (and document hierarchy)	Description
loginUserName	The name of the user requesting the information. This user must have privileges to view the requisition.
requisitionid	The id of the requisition to be interrogated.

GetRequisitionStatus Response

If the request to get the requisition succeeds, the response will include information about the requisition.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    </soap:Body>
</soap:Envelope>
```

The attributes of the `get*RequisitionsResult` response are summarized in the table below:

XML Element	Description
<code>getRequisitionStatusResponse > get*RequisitionsResult</code>	
<code>RequisitionEntryStatuses > RequisitionEntryStatus</code>	One status block for each service in the request
<code>itemNumber</code>	Sequence assigned to the service within the requisition
<code>quantity</code>	Number of services order
<code>requisitionEntryId</code>	Requisition Entry ID for the service
<code>serviceName</code>	Name of the service
<code>status</code>	Current status of the requisition entry
<code>requisitionStepStatuses > RequisitionStepStatus</code>	One StepStatus for each moment configured in the delivery plan for the service
<code>dueDate</code>	Date the current authorization, review or task is due
<code>name</code>	Moment in the delivery plan; for example “Service Group Authorization” or “Delivery project for <service name>”
<code>stepStatus</code>	Status of the task; for example, “In Progress”, “Pending” or “Completed”

Adding Comments to a Requisition

The `addComments` operation adds a user comment to the specified requisition.

addComments Request

The request adds the specified comment to the specified requisition. The user specified must have permission to access the requisition.

XML Element (and document hierarchy)	Description
<code>loginUserName</code>	The name of the user adding the comment
<code>requisitionid</code>	The id of the requisition to be affected
<code>commentText</code>	The text of the user comment

Track Requisitions > Requisition Status

Canceling a Requisition

The `cancelRequisition` operation is used to cancel the specified service request. All services that comprise the requisition are canceled.

The `cancelrequisitionentry` operation cancels the specified service (requisition entry) within a service request. If this is the only (or last) service in the requisition, the requisition is canceled. Otherwise, its status remains unchanged.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:req=>
  <soapenv:Header>
    <req:AuthenticationToken>
      <req:Username>admin</req:Username>
      <req>Password>admin</req>Password>
    </req:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
    <req:cancelRequisition>
      <req:loginUserName>ltierstein</req:loginUserName>
      <req:requisitionId>99</req:requisitionId>
    </req:cancelRequisition>
  </soapenv:Body>
</soapenv:Envelope>
```

The response is shown below (with formatting added for clarity):

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <ns1:cancelRequisitionResponse xmlns:ns1="http://requisition.api.newscale.com">
      <ns1:cancelRequisitionResult>
        ns1:closedDate="2009-06-02T12:04:32.837-07:00"
        ns1:customer="Leslie Tierstein"
        ns1:dueDate="2009-04-03T15:00:00-07:00"
        ns1:id="99"
        ns1:initiator="Leslie Tierstein"
```

```

        ns1:startDate="2009-04-03T09:55:54.843-07:00"
        ns1:status="Cancelled"/>
    </ns1:cancelRequisitionResponse>
</soap:Body>
</soap:Envelope>

```

Web Services for Task Management

Overview

The operations that can be performed via task management web services are summarized in the table below:

Request	Description
approveTask	Approve an authorization/approval.
getAuthorizations	Retrieve authorizations
getAuthorizationsForUser	Retrieve authorizations for a specified user
getMyAuthorizations	Retrieve authorizations assigned to the specified person
rejectSelectedReqEntry	Reject the specified service (requisition entry)
rejectTask	Reject an authorization/approval
reviewTask	Mark a “review” task as reviewed

Getting a List of Authorizations

The getAuthorizations and getMyAuthorizations operations return information about authorizations that are “In Progress”. They differ in the arguments that can be included in the request.

Unlike the Requisition Service operations, which have provisions for separate Web Services Administrative users (specified in the SOAP Header) and the Request Center user to which the operation applies, these Task Service operations allow the specification of only one user, in the SOAP header. Therefore, the user whose authorizations are to be retrieved or processed must have the Task Access capability of the Web Services module. To do this:

- Create a role which includes that capability. Since the ability to perform authorizations is included in the My Services Professional role, create a child of that role:

- Assign that role (either in addition to or instead of My Services Professional) to people whose authorizations need to be reviewed or processed via web services:

getMyAuthorizations Request

getMyAuthorizations returns all open requisitions, up to a specific maximum number of requisitions for the person whose Request Center credentials are specified in the SOAP header. The requisitions are returned in descending order by Requisition ID. This request is supported only for use in the JSR168-compliant Authorizations portlet and should not be used in web services.

getAuthorizations Request

getAuthorizations returns all authorizations, up to a specific maximum number of authorizations, starting with a specified authorization in the list. It also allows you to specify the view type and status of the requisitions to be returned. This request is supported only for use in the JSR168-compliant Authorizations portlet and should not be used in web services.

getAuthorizationsForUser Request (internal only and unsupported)

Very similar to getAuthorizations described above, but adds a userLoginName parameter you can use to specify the user for whom you want to get authorizations.

Sample getAuthorizationsForUser SOAP Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://smtask.api.newscale.com"> <soapenv:Header>
  <smt:AuthenticationToken>
    <smt:Username>admin</smt:Username>
    <smt:Password>admin</smt:Password>
  </smt:AuthenticationToken>
</soapenv:Header>
<soapenv:Body>
  <smt:getAuthorizationsForUser>
    <smt:userLoginName>qreviewer</smt:userLoginName>
    <smt:startRow>0</smt:startRow>
    <smt:numberOfRows>5</smt:numberOfRows>
    <smt:status>1</smt:status>
    <smt:viewType>2</smt:viewType>
  </smt:getAuthorizationsForUser>
</soapenv:Body>
</soapenv:Envelope>
```

Useful Parameters for getAuthorizations and getAuthorizationsForUser Requests

Parameter	Values
Status	Ongoing – 1 Cancelled – 2 Approved – 3 Rejected – 4 Reviewed – 5 All – 6
ViewType	My Authorizations – 1 My Assigned and Unassigned – 2

Approving or Rejecting an Authorization

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:smt="http://smtask.api.newscale.com">
  <soapenv:Header>
    <smt:AuthenticationToken>
      <!--Optional:-->
      <smt:Username>admin</smt:Username>
      <!--Optional:-->
      <smt:Password>admin</smt:Password>
    </smt:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
    <smt:approveTask>
      <smt:approverLoginName>maria</smt:approverLoginName>
      <smt:taskID>281</smt:taskID>
    </smt:approveTask>
  </soapenv:Body>
</soapenv:Envelope>
```

```

    </smt:approveTask>
  </soapenv:Body>
</soapenv:Envelope>

```

The response is shown below (with formatting added for clarity):

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <ns1:approveTaskResponse xmlns:ns1="http://smtask.api.newscale.com">
      <ns1:approveTaskResult
        ns1:actionID="5"
        ns1:requisitionId="103"
        ns1:status="approved"
        ns1:taskName="Computer Memory - Upgrade - APPROVAL NEEDED"/>
    </ns1:approveTaskResponse>
  </soap:Body>
</soap:Envelope>

```

Web Services for Portfolio Management

Overview

Request	Description
exportOfferingCostData	Export the costing information on one or more offerings
getAllServiceOfferingStatus	Get the name and current status of all service offerings
getServiceOfferingStatus	Get the name and current status of all service offerings whose names match the specified search string
importOfferingCostData	Import the costing information

Exporting Offering Cost Data

The exportOfferingCostData operation exports the costing information on one or more services.

Retrieving Service Offerings and their Status

The getAllServiceOfferingStatus operation retrieves the name of all service offerings and their current status.

The getServiceOfferingStatus operation retrieves the name and status of all services whose name meets the search criteria specified. Search criteria may consist of any string; the search is not case sensitive.

getServiceOfferingStatus Request

A sample getServiceOfferingStatus request is shown below.

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:fin="http://financialmanagement.api.newscale.com">

```

```

<soapenv:Header>
  <fin:AuthenticationToken>
    <fin:Username>admin</fin:Username>
    <fin:Password>admin</fin:Password>
  </fin:AuthenticationToken>
</soapenv:Header>
<soapenv:Body>
  <fin:getServiceOfferingStatus>
    <fin:searchString>EMAIL</fin:searchString>
  </fin:getServiceOfferingStatus>
</soapenv:Body>
</soapenv:Envelope>

```

getAllServiceOfferingStatus Request

No user-supplied parameters are required in the message body.

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:fin="http://financialmanagement.api.newscale.com">
  <soapenv:Header>
    <fin:AuthenticationToken>
      <fin:Username>admin</fin:Username>
      <fin:Password>admin</fin:Password>
    </fin:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
    <fin:getAllServiceOfferingStatus/>
  </soapenv:Body>
</soapenv:Envelope>

```

getServiceOfferingStatus Response

For each service offering, the response includes the offering name and its status.

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <ns1:getServiceOfferingStatusResponse
xmlns:ns1="http://financialmanagement.api.newscale.com">
      <ns1:getServiceOfferingsResult>
        <ns1:ServiceOfferingStatus>
          <serviceOfferingName xmlns="http://financialmanagement.api.newscale.com">Email and
Calendar - FY08</serviceOfferingName>
          <status xmlns="http://financialmanagement.api.newscale.com">draft</status>
        </ns1:ServiceOfferingStatus>
      </ns1:getServiceOfferingsResult>
    </ns1:getServiceOfferingStatusResponse>
  </soap:Body>
</soap:Envelope>

```


Sample Requests and Responses

getServiceDefinition Response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <ns1:getServiceDefinitionResponse xmlns:ns1=>
      <ns1:getServiceDefinitionResult>
        <ictionaries>
          <Dictionary>
            <fields>
              <DictionaryField>
                <canSelectMultiple>>false</canSelectMultiple>
                <defaultValue>
                  <string/>
                </defaultValue>
                <fieldDataType>Text</fieldDataType>
                <fieldName>ModelNumber</fieldName>
                <inputType>text</inputType>
                <label>Model Number</label>
                <mandatory>>false</mandatory>
                <maxLength>50</maxLength>
                <selectableValues>
                  <string/>
                </selectableValues>
              </DictionaryField>
              <DictionaryField>
                <canSelectMultiple>>false</canSelectMultiple>
                <defaultValue>
                  <string/>
                </defaultValue>
                <fieldDataType>Text</fieldDataType>
                <fieldName>AssetTag</fieldName>
                <inputType>text</inputType>
                <label>Asset Tag</label>
                <mandatory>>false</mandatory>
                <maxLength>50</maxLength>
                <selectableValues>
                  <string/>
                </selectableValues>
              </DictionaryField>
            </fields>
            <name>NewLaptop</name>
            <readable>>true</readable>
            <writable>>true</writable>
          </Dictionary>
          <Dictionary>
            <fields>
              <DictionaryField>
                <canSelectMultiple>>false</canSelectMultiple>
                <defaultValue>
                  <string>admin</string>
                </defaultValue>
                <fieldDataType>Text</fieldDataType>
                <fieldName>First_Name</fieldName>
                <inputType>text</inputType>
                <label>First Name</label>
                <mandatory>>false</mandatory>

```

```

    <maxLength>100</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
  </DictionaryField>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string>admin</string>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Last_Name</fieldName>
  <inputType>text</inputType>
  <label>Last Name</label>
  <mandatory>>false</mandatory>
  <maxLength>100</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string>admin</string>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Login_ID</fieldName>
  <inputType>hidden</inputType>
  <label>Login ID</label>
  <mandatory>>false</mandatory>
  <maxLength>200</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Personal_Identification</fieldName>
  <inputType>text</inputType>
  <label>Personal_Identification</label>
  <mandatory>>false</mandatory>
  <maxLength>510</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string>ed @cisco.com</string>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Email_Address</fieldName>
  <inputType>text</inputType>
  <label>Email Address</label>
  <mandatory>>false</mandatory>
  <maxLength>1024</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>

```

```

</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string>Site Administration</string>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Home_Organizational_Unit</fieldName>
  <inputType>text</inputType>
  <label>Department</label>
  <mandatory>>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Company_State</fieldName>
  <inputType>text</inputType>
  <label>State</label>
  <mandatory>>false</mandatory>
  <maxLength>100</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Supervisor</fieldName>
  <inputType>hidden</inputType>
  <label>Supervisor</label>
  <mandatory>>false</mandatory>
  <maxLength>100</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Supervisor_Email</fieldName>
  <inputType>hidden</inputType>
  <label>Supervisor Email</label>
  <mandatory>>false</mandatory>
  <maxLength>1024</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>

```

```

    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Custom_1</fieldName>
  <inputType>text</inputType>
  <label>Custom_1</label>
  <mandatory>>false</mandatory>
  <maxLength>200</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Custom_2</fieldName>
  <inputType>text</inputType>
  <label>Custom_2</label>
  <mandatory>>false</mandatory>
  <maxLength>200</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
</fields>
<name>Customer_Information</name>
<readable>>true</readable>
<writable>>true</writable>
</Dictionary>
<Dictionary>
  <fields>
    <DictionaryField>
      <canSelectMultiple>>false</canSelectMultiple>
      <defaultValue>
        <string>admin</string>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>First_Name</fieldName>
      <inputType>text</inputType>
      <label>First Name</label>
      <mandatory>>false</mandatory>
      <maxLength>100</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
    </DictionaryField>
    <DictionaryField>
      <canSelectMultiple>>false</canSelectMultiple>
      <defaultValue>
        <string>admin</string>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Last_Name</fieldName>
      <inputType>text</inputType>
      <label>Last Name</label>
      <mandatory>>false</mandatory>
      <maxLength>100</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
    </DictionaryField>
  </fields>
</Dictionary>

```

```

<DictionaryField>
  <canSelectMultiple>false</canSelectMultiple>
  <defaultValue>
    <string>admin</string>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Login_ID</fieldName>
  <inputType>text</inputType>
  <label>Login ID</label>
  <mandatory>false</mandatory>
  <maxLength>200</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Personal_Identification</fieldName>
  <inputType>hidden</inputType>
  <label>Personal Identification</label>
  <mandatory>false</mandatory>
  <maxLength>510</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>false</canSelectMultiple>
  <defaultValue>
    <string>ed @cisco.com</string>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Email_Address</fieldName>
  <inputType>text</inputType>
  <label>Email Address</label>
  <mandatory>false</mandatory>
  <maxLength>1024</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>false</canSelectMultiple>
  <defaultValue>
    <string>Site Administration</string>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Home_Organizational_Unit</fieldName>
  <inputType>text</inputType>
  <label>Department</label>
  <mandatory>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
</fields>
<name>Initiator_Information</name>
<readable>>false</readable>
<writable>>false</writable>

```

```

</Dictionary>
<Dictionary>
  <fields>
    <DictionaryField>
      <canSelectMultiple>>false</canSelectMultiple>
      <defaultValue>
        <string>Yes</string>
      </defaultValue>
      <fieldDataType>Boolean</fieldDataType>
      <fieldName>PerformWork</fieldName>
      <inputType>radio</inputType>
      <label>Will work be performed at the customer location?</label>
      <mandatory>>false</mandatory>
      <maxLength>0</maxLength>
      <selectableValues>
        <string>Yes</string>
        <string>No</string>
      </selectableValues>
    </DictionaryField>
  </fields>
  <name>RC_PerformWork</name>
  <readable>>true</readable>
  <writable>>true</writable>
</Dictionary>
<Dictionary>
  <fields>
    <DictionaryField>
      <canSelectMultiple>>false</canSelectMultiple>
      <defaultValue>
        <string/>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Street1</fieldName>
      <inputType>text</inputType>
      <label>Street</label>
      <mandatory>>false</mandatory>
      <maxLength>50</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
    </DictionaryField>
    <DictionaryField>
      <canSelectMultiple>>false</canSelectMultiple>
      <defaultValue>
        <string/>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Street2</fieldName>
      <inputType>hidden</inputType>
      <label>Street2</label>
      <mandatory>>false</mandatory>
      <maxLength>50</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
    </DictionaryField>
    <DictionaryField>
      <canSelectMultiple>>false</canSelectMultiple>
      <defaultValue>
        <string/>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Floor</fieldName>
      <inputType>hidden</inputType>

```

```

<label>Floor</label>
<mandatory>>false</mandatory>
<maxLength>50</maxLength>
<selectableValues>
  <string/>
</selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>OfficeCubeRoom</fieldName>
  <inputType>text</inputType>
  <label>OfficeCubeRoom</label>
  <mandatory>>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>Building</fieldName>
  <inputType>hidden</inputType>
  <label>Building</label>
  <mandatory>>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>City</fieldName>
  <inputType>text</inputType>
  <label>City</label>
  <mandatory>>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>State</fieldName>
  <inputType>text</inputType>
  <label>State</label>
  <mandatory>>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>

```

```

        <string/>
    </selectableValues>
</DictionaryField>
<DictionaryField>
    <canSelectMultiple>>false</canSelectMultiple>
    <defaultValue>
        <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>PostalCode</fieldName>
    <inputType>text</inputType>
    <label>Zip Code</label>
    <mandatory>>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
        <string/>
    </selectableValues>
</DictionaryField>
<DictionaryField>
    <canSelectMultiple>>false</canSelectMultiple>
    <defaultValue>
        <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Country</fieldName>
    <inputType>hidden</inputType>
    <label>Country</label>
    <mandatory>>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
        <string/>
    </selectableValues>
</DictionaryField>
<DictionaryField>
    <canSelectMultiple>>false</canSelectMultiple>
    <defaultValue>
        <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>MailStop</fieldName>
    <inputType>hidden</inputType>
    <label>MailStop</label>
    <mandatory>>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
        <string/>
    </selectableValues>
</DictionaryField>
<DictionaryField>
    <canSelectMultiple>>false</canSelectMultiple>
    <defaultValue>
        <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>Region</fieldName>
    <inputType>hidden</inputType>
    <label>Region</label>
    <mandatory>>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
        <string/>
    </selectableValues>
</DictionaryField>
<DictionaryField>

```



```

    <canSelectMultiple>>false</canSelectMultiple>
    <defaultValue>
      <string/>
    </defaultValue>
    <fieldDataType>Text</fieldDataType>
    <fieldName>District</fieldName>
    <inputType>hidden</inputType>
    <label>District</label>
    <mandatory>>false</mandatory>
    <maxLength>50</maxLength>
    <selectableValues>
      <string/>
    </selectableValues>
  </DictionaryField>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>LocationName</fieldName>
  <inputType>hidden</inputType>
  <label>LocationName</label>
  <mandatory>>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>LocationCode</fieldName>
  <inputType>hidden</inputType>
  <label>LocationCode</label>
  <mandatory>>false</mandatory>
  <maxLength>50</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
</fields>
<name>RC_RequestorLocation</name>
<readable>>true</readable>
<writable>>true</writable>
</Dictionary>
<Dictionary>
  <fields>
    <DictionaryField>
      <canSelectMultiple>>false</canSelectMultiple>
      <defaultValue>
        <string/>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>Street</fieldName>
      <inputType>text</inputType>
      <label>Street</label>
      <mandatory>>false</mandatory>
      <maxLength>40</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
    </DictionaryField>
  </fields>
</Dictionary>

```

```

    </selectableValues>
  </DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>OfficeCubeRoom</fieldName>
  <inputType>text</inputType>
  <label>OfficeCubeRoom</label>
  <mandatory>>false</mandatory>
  <maxLength>40</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>State</fieldName>
  <inputType>text</inputType>
  <label>State</label>
  <mandatory>>false</mandatory>
  <maxLength>40</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>City</fieldName>
  <inputType>text</inputType>
  <label>City</label>
  <mandatory>>false</mandatory>
  <maxLength>40</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>
  <defaultValue>
    <string/>
  </defaultValue>
  <fieldDataType>Text</fieldDataType>
  <fieldName>BuildingName</fieldName>
  <inputType>hidden</inputType>
  <label>BuildingName</label>
  <mandatory>>false</mandatory>
  <maxLength>40</maxLength>
  <selectableValues>
    <string/>
  </selectableValues>
</DictionaryField>
<DictionaryField>
  <canSelectMultiple>>false</canSelectMultiple>

```

```

        <defaultValue>
          <string/>
        </defaultValue>
        <fieldDataType>Text</fieldDataType>
        <fieldName>Floor</fieldName>
        <inputType>hidden</inputType>
        <label>Floor</label>
        <mandatory>>false</mandatory>
        <maxLength>40</maxLength>
        <selectableValues>
          <string/>
        </selectableValues>
      </DictionaryField>
    <DictionaryField>
      <canSelectMultiple>>false</canSelectMultiple>
      <defaultValue>
        <string/>
      </defaultValue>
      <fieldDataType>Text</fieldDataType>
      <fieldName>ZipCode</fieldName>
      <inputType>text</inputType>
      <label>Zip Code</label>
      <mandatory>>false</mandatory>
      <maxLength>15</maxLength>
      <selectableValues>
        <string/>
      </selectableValues>
    </DictionaryField>
  </fields>
  <name>RC_ServiceLocation</name>
  <readable>>true</readable>
  <writable>>true</writable>
</Dictionary>
</dictionaries>
<estimatedpriceperunit>1500.0</estimatedpriceperunit>
<name>New Standard Laptop Computer</name>
<pricingmodel>0</pricingmodel>
<quantity>0</quantity>
<serviceId>5</serviceId>
<version>32</version>
</ns1:getServiceDefinitionResult>
</ns1:getServiceDefinitionResponse>
</soap:Body>
</soap:Envelope>

```

Sample submitRequisition Request

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:req=>
  <soapenv:Header>
    <req:AuthenticationToken>
      <req:Username>admin</req:Username>
      <req>Password>admin</req>Password>
    </req:AuthenticationToken>
  </soapenv:Header>
  <soapenv:Body>
    <req:submitRequisition>
      <req:initiatorLoginName>admin</req:initiatorLoginName>
      <req:customerLoginName>admin</req:customerLoginName>
      <req:serviceRequests>
        <req:ServiceRequest>
          <req:name>New Standard Laptop Computer</req:name>
          <req:quantity>1</req:quantity>
        </req:ServiceRequest>
      </req:serviceRequests>
    </req:submitRequisition>
  </soapenv:Body>
</soapenv:Envelope>

```

```

<req:sections>
  <req:Section>
    <req:fields>
      <req:Field>
        <req:name>ModelNumber</req:name>
        <req:value>
          <req:string>T60</req:string>
        </req:value>
      </req:Field>
      <req:Field>
        <req:name>AssetTag</req:name>
        <req:value>
          <req:string>ABC123</req:string>
        </req:value>
      </req:Field>
    </req:fields>
    <req:name>NewLaptop</req:name>
  </req:Section>
  <req:Section>
    <req:fields>
      <req:Field>
        <req:name>First_Name</req:name>
        <req:value>
          <req:string>admin</req:string>
        </req:value>
      </req:Field>
      <req:Field>
        <req:name>Last_Name</req:name>
        <req:value>
          <req:string>admin</req:string>
        </req:value>
      </req:Field>
      <req:Field>
        <req:name>Login_ID</req:name>
        <req:value>
          <req:string>admin</req:string>
        </req:value>
      </req:Field>
      <req:Field>
        <req:name>Personal_Identification</req:name>
        <req:value>
          <req:string />
        </req:value>
      </req:Field>
      <req:Field>
        <req:name>Email_Address</req:name>
        <req:value>
          <req:string>training3@cisco.com</req:string>
        </req:value>
      </req:Field>
      <req:Field>
        <req:name>Home_Organizational_Unit</req:name>
        <req:value>
          <req:string>Site Administration</req:string>
        </req:value>
      </req:Field>
      <req:Field>
        <req:name>Company_State</req:name>
        <req:value>
          <req:string />
        </req:value>
      </req:Field>
      <req:Field>
        <req:name>Supervisor</req:name>

```

```

        <req:value>
          <req:string />
        </req:value>
      </req:Field>
    <req:Field>
      <req:name>Supervisor_Email</req:name>
      <req:value>
        <req:string />
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Custom_1</req:name>
      <req:value>
        <req:string />
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Custom_2</req:name>
      <req:value>
        <req:string />
      </req:value>
    </req:Field>
  </req:fields>
</req:Section>
<req:Section>
  <req:fields>
    <req:Field>
      <req:name>First_Name</req:name>
      <req:value>
        <req:string>admin</req:string>
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Last_Name</req:name>
      <req:value>
        <req:string>admin</req:string>
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Login_ID</req:name>
      <req:value>
        <req:string>admin</req:string>
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Personal_Identification</req:name>
      <req:value>
        <req:string />
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Email_Address</req:name>
      <req:value>
        <req:string>training3@cisco.com</req:string>
      </req:value>
    </req:Field>
    <req:Field>
      <req:name>Home_Organizational_Unit</req:name>
      <req:value>
        <req:string>Site Administration</req:string>
      </req:value>
    </req:Field>
  </req:fields>

```

```

        <req:name>Initiator_Information</req:name>
      </req:Section>
    </req:sections>
    <req:version>32</req:version>
  </req:ServiceRequest>
</req:serviceRequests>
</req:submitRequisition>
</soapenv:Body>
</soapenv:Envelope>

```

Sample getMyAuthorizations Response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <ns1:getMyAuthorizationsResponse xmlns:ns1="http://smtask.api.newscale.com">
      <ns1:getMyAuthorizationsResult>
        <ns1:Activity>
          <activityFormId xmlns="http://smtask.api.newscale.com">2</activityFormId>
          <activityTypeId xmlns="http://smtask.api.newscale.com">2</activityTypeId>
          <actualDuration xmlns="http://smtask.api.newscale.com">0.0</actualDuration>
          <agentId xmlns="http://smtask.api.newscale.com">0</agentId>
          <clientOrganizationalUnit xmlns="http://smtask.api.newscale.com">
            <authorizationStructure>0</authorizationStructure>
            <billable>true</billable>
            <costCenterCode xsi:nil="true"/>
            <description xsi:nil="true"/>
            <GUID>3C921968-6474-45B2-8D65-A1822E52782F</GUID>
            <id>6</id>
            <localeId>1</localeId>
            <managerId>0</managerId>
            <managerName xsi:nil="true"/>
            <name>Field Sales</name>
            <organizationalUnitTypeId>2</organizationalUnitTypeId>
            <parentId>0</parentId>
            <parentName xsi:nil="true"/>
            <parentOrganizationalUnitGuid xsi:nil="true"/>
            <placeId>0</placeId>
            <placeName xsi:nil="true"/>
            <recordStateId>1</recordStateId>
            <tenantId>1</tenantId>
          </clientOrganizationalUnit>
          <clientOuId xmlns="http://smtask.api.newscale.com">6</clientOuId>
          <creatorObjectId xmlns="http://smtask.api.newscale.com">57</creatorObjectId>
          <creatorObjectInstId
xmlns="http://smtask.api.newscale.com">9</creatorObjectInstId>
          <customer xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <customerId xmlns="http://smtask.api.newscale.com">12</customerId>
          <customerName xmlns="http://smtask.api.newscale.com">Terry Training</customerName>
          <customerRoleId xmlns="http://smtask.api.newscale.com">0</customerRoleId>
          <customerRoleName xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>
          <defActivityId xmlns="http://smtask.api.newscale.com">0</defActivityId>
          <depth xmlns="http://smtask.api.newscale.com">0</depth>
          <displayOrder xmlns="http://smtask.api.newscale.com">0</displayOrder>
          <dueOn xmlns="http://smtask.api.newscale.com">2009-06-03T23:00:00</dueOn>
          <dueOnTz xmlns="http://smtask.api.newscale.com">149</dueOnTz>
          <effort xmlns="http://smtask.api.newscale.com">0.5</effort>
          <escalationLevel xmlns="http://smtask.api.newscale.com">0</escalationLevel>
          <expectedDuration xmlns="http://smtask.api.newscale.com">8.0</expectedDuration>
          <flagId xmlns="http://smtask.api.newscale.com">0</flagId>
          <formURL xsi:nil="true" xmlns="http://smtask.api.newscale.com"/>

```

```

<group xmlns="http://smtask.api.newscale.com">0</group>
<hasChildren xmlns="http://smtask.api.newscale.com">false</hasChildren>
<icon xsi:nil="true" xmlns="http://smtask.api.newscale.com" />
<id xmlns="http://smtask.api.newscale.com">422</id>
<instructions xmlns="http://smtask.api.newscale.com" />
<isBusy xmlns="http://smtask.api.newscale.com">0</isBusy>
<isLast xmlns="http://smtask.api.newscale.com">false</isLast>
<lastChannelId xsi:nil="true" xmlns="http://smtask.api.newscale.com" />
<listCount xmlns="http://smtask.api.newscale.com">-1</listCount>
<nextActionId xmlns="http://smtask.api.newscale.com">5</nextActionId>
<overbookTime xmlns="http://smtask.api.newscale.com">0</overbookTime>
<parentId xmlns="http://smtask.api.newscale.com">0</parentId>
<performerActualDuration
xmlns="http://smtask.api.newscale.com">0.0</performerActualDuration>
  <performerId xmlns="http://smtask.api.newscale.com">11</performerId>
  <performerName xmlns="http://smtask.api.newscale.com">Jared
Roberts</performerName>
  <performerOfficeId xmlns="http://smtask.api.newscale.com">0</performerOfficeId>
  <performerRoleId xmlns="http://smtask.api.newscale.com">331</performerRoleId>
  <performerRoleName xsi:nil="true" xmlns="http://smtask.api.newscale.com" />
  <performerSharable
xmlns="http://smtask.api.newscale.com">false</performerSharable>
  <performerShared xmlns="http://smtask.api.newscale.com">false</performerShared>
  <priority xmlns="http://smtask.api.newscale.com">0</priority>
  <priorityName xsi:nil="true" xmlns="http://smtask.api.newscale.com" />
  <processId xmlns="http://smtask.api.newscale.com">0</processId>
  <projectActivityId xmlns="http://smtask.api.newscale.com">0</projectActivityId>
  <reqId xmlns="http://smtask.api.newscale.com">170</reqId>
  <retryCount xmlns="http://smtask.api.newscale.com">0</retryCount>
  <scheduledStart
xmlns="http://smtask.api.newscale.com">2009-06-03T15:00:00-07:00</scheduledStart>
  <startedOn
xmlns="http://smtask.api.newscale.com">2009-06-03T12:09:41.443-07:00</startedOn>
  <startedOnTz xmlns="http://smtask.api.newscale.com">149</startedOnTz>
  <stateId xmlns="http://smtask.api.newscale.com">6</stateId>
  <stateName xsi:nil="true" xmlns="http://smtask.api.newscale.com" />
  <stepId xmlns="http://smtask.api.newscale.com">4</stepId>
  <stepLogicName xsi:nil="true" xmlns="http://smtask.api.newscale.com" />
  <subject xmlns="http://smtask.api.newscale.com">Computer Memory - Upgrade -
APPROVAL NEEDED</subject>
  <taskUrl xsi:nil="true" xmlns="http://smtask.api.newscale.com" />
  <ticketId xmlns="http://smtask.api.newscale.com">175</ticketId>
  <ticketObjectId xmlns="http://smtask.api.newscale.com">37</ticketObjectId>
  <totalCost xmlns="http://smtask.api.newscale.com">0.0</totalCost>
  <waiting xmlns="http://smtask.api.newscale.com">0</waiting>
  <WDDXCheckList xsi:nil="true" xmlns="http://smtask.api.newscale.com" />
</ns1:Activity>
</ns1:getMyAuthorizationsResult>
</ns1:getMyAuthorizationsResponse>
</soap:Body>
</soap:Envelope>

```

Sample exportOfferingCostData Response

```

<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <ns1:exportOfferingCostDataResponse
xmlns:ns1="http://financialmanagement.api.newscale.com">
      <ns1:exportOfferingCostDataResult>
        <ns1:ServiceOffering>

```

```

<componentServices xmlns="http://financialmanagement.api.newscafe.com">
  <ComponentService>
    <category>Infrastructure</category>
    <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
    <name>Internet</name>
    <objectiveMultiplierList>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Bronze Support</objectivename>
      </ObjectiveMultiplier>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Gold Support</objectivename>
      </ObjectiveMultiplier>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Silver Support</objectivename>
      </ObjectiveMultiplier>
    </objectiveMultiplierList>
    <quantityIncludedPerAgreementPeriod/>
    <unitPrice>0.0</unitPrice>
  </ComponentService>
  <ComponentService>
    <category>Hardware</category>
    <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
    <name>CD Burner - Add</name>
    <objectiveMultiplierList>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Bronze Support</objectivename>
      </ObjectiveMultiplier>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Gold Support</objectivename>
      </ObjectiveMultiplier>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Silver Support</objectivename>
      </ObjectiveMultiplier>
    </objectiveMultiplierList>

<quantityIncludedPerAgreementPeriod>Users.units*0.9</quantityIncludedPerAgreementPeriod>
  <unitPrice>87.45</unitPrice>
</ComponentService>
  <ComponentService>
    <category>Hardware</category>
    <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
    <name>Computer Memory - Upgrade</name>
    <objectiveMultiplierList>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Bronze Support</objectivename>
      </ObjectiveMultiplier>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Gold Support</objectivename>
      </ObjectiveMultiplier>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Silver Support</objectivename>
      </ObjectiveMultiplier>
    </objectiveMultiplierList>

<quantityIncludedPerAgreementPeriod>Users.units*0.15</quantityIncludedPerAgreementPeriod>

```



```

    <unitPrice>300.0</unitPrice>
  </ComponentService>
<ComponentService>
  <category>Hardware</category>
  <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
  <name>Desktop Printer - Black and White Inkjet</name>
  <objectiveMultiplierList>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Bronze Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Gold Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Silver Support</objectivename>
    </ObjectiveMultiplier>
  </objectiveMultiplierList>

<quantityIncludedPerAgreementPeriod>Users.units*0.05</quantityIncludedPerAgreementPeriod>
  <unitPrice>87.0</unitPrice>
</ComponentService>
<ComponentService>
  <category>Hardware</category>
  <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
  <name>New Desktop Computer - Add</name>
  <objectiveMultiplierList>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Bronze Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Gold Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Silver Support</objectivename>
    </ObjectiveMultiplier>
  </objectiveMultiplierList>

<quantityIncludedPerAgreementPeriod>Users.units*0.05</quantityIncludedPerAgreementPeriod>
  <unitPrice>1260.0</unitPrice>
</ComponentService>
<ComponentService>
  <category>Hardware</category>
  <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
  <name>USB Flash Drive</name>
  <objectiveMultiplierList>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Bronze Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Gold Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Silver Support</objectivename>
    </ObjectiveMultiplier>
  </objectiveMultiplierList>

```

```

<quantityIncludedPerAgreementPeriod>Users.units*1</quantityIncludedPerAgreementPeriod>
  <unitPrice>41.0</unitPrice>
</ComponentService>
<ComponentService>
  <category>Software</category>
  <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
  <name>Adobe Photoshop - Add</name>
  <objectiveMultiplierList>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Bronze Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Gold Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Silver Support</objectivename>
    </ObjectiveMultiplier>
  </objectiveMultiplierList>

<quantityIncludedPerAgreementPeriod>Users.units*0.25</quantityIncludedPerAgreementPeriod>
  <unitPrice>239.0</unitPrice>
</ComponentService>
<ComponentService>
  <category>Software</category>
  <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
  <name>Microsoft Office - Add</name>
  <objectiveMultiplierList>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Bronze Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Gold Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Silver Support</objectivename>
    </ObjectiveMultiplier>
  </objectiveMultiplierList>

<quantityIncludedPerAgreementPeriod>Users.units*1</quantityIncludedPerAgreementPeriod>
  <unitPrice>133.0</unitPrice>
</ComponentService>
<ComponentService>
  <category>Support</category>
  <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
  <name>Service Desk Level 1 Support</name>
  <objectiveMultiplierList>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Bronze Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Gold Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Silver Support</objectivename>
    </ObjectiveMultiplier>
  </objectiveMultiplierList>

```

```

        </ObjectiveMultiplier>
      </objectiveMultiplierList>

    <quantityIncludedPerAgreementPeriod>Users.units*5</quantityIncludedPerAgreementPeriod>
    <unitPrice>15.0</unitPrice>
  </ComponentService>
  <ComponentService>
    <category>Support</category>
    <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
    <name>Service Desk Level 2 Support</name>
    <objectiveMultiplierList>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Bronze Support</objectivename>
      </ObjectiveMultiplier>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Gold Support</objectivename>
      </ObjectiveMultiplier>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Silver Support</objectivename>
      </ObjectiveMultiplier>
    </objectiveMultiplierList>

  <quantityIncludedPerAgreementPeriod>Users.units*2</quantityIncludedPerAgreementPeriod>
  <unitPrice>50.0</unitPrice>
</ComponentService>
<ComponentService>
  <category>Support</category>
  <estimatedQuantityPerPeriod>0.0</estimatedQuantityPerPeriod>
  <name>Virus Removal</name>
  <objectiveMultiplierList>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Bronze Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Gold Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
      <multiplier>1.0</multiplier>
      <objectivename>Silver Support</objectivename>
    </ObjectiveMultiplier>
  </objectiveMultiplierList>

  <quantityIncludedPerAgreementPeriod>Users.units*0.9</quantityIncludedPerAgreementPeriod>
  <unitPrice>42.0</unitPrice>
</ComponentService>
</componentServices>
<costDrivers xmlns="http://financialmanagement.api.newscale.com">
  <CostDriver>
    <benchmarkUnitPrice>850.0</benchmarkUnitPrice>
    <estimatedTotalCost>69.46</estimatedTotalCost>
    <estimatedUnitsPerPeriod>1.00</estimatedUnitsPerPeriod>
    <margin>0.0</margin>
    <name>Users</name>
    <objectiveMultiplierList>
      <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Bronze Support</objectivename>
      </ObjectiveMultiplier>
      <ObjectiveMultiplier>

```

```

        <multiplier>1.0</multiplier>
        <objectivename>Gold Support</objectivename>
    </ObjectiveMultiplier>
    <ObjectiveMultiplier>
        <multiplier>1.0</multiplier>
        <objectivename>Silver Support</objectivename>
    </ObjectiveMultiplier>
</objectiveMultiplierList>
</CostDriver>
<CostDriver>
    <benchmarkUnitPrice>0.0</benchmarkUnitPrice>
    <estimatedTotalCost>0.83</estimatedTotalCost>
    <estimatedUnitsPerPeriod>1.00</estimatedUnitsPerPeriod>
    <margin>0.0</margin>
    <name>Gigabytes</name>
    <objectiveMultiplierList>
        <ObjectiveMultiplier>
            <multiplier>1.0</multiplier>
            <objectivename>Bronze Support</objectivename>
        </ObjectiveMultiplier>
        <ObjectiveMultiplier>
            <multiplier>1.0</multiplier>
            <objectivename>Gold Support</objectivename>
        </ObjectiveMultiplier>
        <ObjectiveMultiplier>
            <multiplier>1.0</multiplier>
            <objectivename>Silver Support</objectivename>
        </ObjectiveMultiplier>
    </objectiveMultiplierList>
</CostDriver>
</costDrivers>
<name xmlns="http://financialmanagement.api.newscale.com">Employee Desktop
Computing</name>
    <status xsi:nil="true" xmlns="http://financialmanagement.api.newscale.com"/>
</ns1:ServiceOffering>
</ns1:exportOfferingCostDataResult>
</ns1:exportOfferingCostDataResponse>
</soap:Body>
</soap:Envelope>

```

Web Services Error Messages

In the error messages below, the symbol of a number enclosed in curly brackets (for example, '{0}') is replaced in the actual error message with the name or identifier of the object that caused the error.

AUTH_0001	The user has not been authenticated yet or the session has timed out.
AUTH_0002	Authentication failed for user '{0}'.
AUTH_0003	Request Center is configured for SSO but some configuration problems are preventing the SSO from working correctly.
AUTH_0005	The user name header is invalid. It is either not present or empty. Please send a valid header.
AUTH_0006	Access to web services has been turned off.
AUTH_0007	User does not have access to this web service.

INFRA_0001	Cannot submit the requisition as the Request Center Business Engine queue is not available and the Administration setting for asynchronous submission has been turned on. Please try later or contact your administrator.
REQ_0001	Initiator '{0}' is not found in the system.
REQ_0002	Customer '{0}' is not found in the system.
REQ_0003	Service '{0}' is not found in the system.
REQ_0004	User '{0}' is not found in the system.
REQ_0005	RequisitionID '{0}' is not found in the system.
REQ_0006	Cannot cancel requisition entry '{0}' as the specified requisition id '{1}' does not match.
REQ_0007	Cannot cancel requisition entry '{0}' as the specified service '{1}' does not match.
REQ_0008	User does not have permission to cancel this requisition. Only the requisition owner can cancel a requisition.
REQ_0009	This requisition has already been cancelled.
REQ_0010	The customer '{0}' does not have permission to order the service '{1}'.
REQ_0011	The service '{0}' is not orderable.
REQ_0012	User does not have permission to add comments to this requisition.
REQ_0013	Service Form Mandatory Field '{0}' is not filled.
REQ_0014	Service Form Field '{0}' exceeds maximum length allowed.
REQ_0015	Please enter a valid Number in the Field '{0}'.
REQ_0016	Service Form Field '{0}' should have only one value.
REQ_0017	User does not have permission to access this requisition.
REQ_0018	The version specified in the request does not match the version in the database for service '{0}'. Please use the latest service definition.
REQ_0019	The initiator '{0}' does not have Order on Behalf permission for this customer '{1}'.
REQ_0020	The authenticated user '{0}' does not have web service Requisition System Account capability.
REQ_0021	The value you passed for this Field '{0}' does not exist in the option list.
REQ_0022	The values for this Field '{0}' have more data than designed values.
REQ_0023	The value you passed for this Field '{0}' does not exist in the option list.
REQ_0024	Service Form Field '{0}' has a Date format issue.
REQ_0025	Service Form Field '{0}' has a DateTime format issue.
REQ_0026	Service Form Field '{0}', you provided the login as '{1}' does not exist in the system. Please input the login name.
REQ_0027	You cannot cancel this Requisition ID '{0}'. The requisition is closed.
REQ_0028	The Requisition ID '{0}' is in "point of no return" status. You cannot cancel this requisition.
REQ_0029	You cannot cancel this Requisition Entry Id '{0}'. The Requisition Entry is closed.
REQ_0030	The status of service '{0}' is Inactive.

REQ_0031	You cannot cancel this Requisition Entry Id '{0}'. The Requisition Entry is already cancelled.
REQ_0032	The value you passed for this Field '{0}' does not exist in the option list.
REQ_0033	The value you passed for this Field '{0}' does not exist in the option list.
REQ_0034	The values for this Field '{0}' have more data than designed values.
REQ_0035	The dictionary name '{0}' does not exist for this service. Please correct the field name.
REQ_0036	The dictionary field '{0}' does not exist for this dictionary '{0}'. Please correct the field name.
REQ_0037	User does not have permission to cancel this requisition entry.
REQ_0038	Some of the fields are missing in this dictionary '{0}'.
REQ_0100	Runtime Exception occurred. This can be caused by a legitimate Business Engine workflow exception (for example, the task is not allowed to be cancelled as defined in Service Designer). User should check the task definition.
REQ_0101	Runtime error occurred while reading the service form data.
REQ_0101	Runtime error occurred while reading the service form data.
TASK_0005	The task '{0}' cannot be rejected.
TASK_0008	The task '{0}' doesn't exist in the system.
TASK_0001	The user '{0}' doesn't have permission to approve the task '{1}'.
TASK_0002	The user '{0}' doesn't have permission to reject the task '{1}'.
TASK_0003	The user '{0}' doesn't have permission to review the task '{1}'.
TASK_0004	The task '{0}' is not an approval task.
TASK_0005	The task '{0}' cannot be rejected.
TASK_0006	The task '{0}' is not a review task.
TASK_0008	The task '{0}' does not exist in the system.
TASK_0009	The requisition entry id '{0}' for the specified task id '{1}' does not match.
TASK_0010	The task '{0}' does not contain more than one requisition entries.
TASK_0011	The task '{0}' does not have financial or OU authorization.
TASK_0012	The user '{0}' does not have permission to reject partial requisition entry for this task '{1}'.
TASK_0013	The task '{0}' has already been rejected.



CHAPTER 6

REST API

- [Overview, page 6-1](#)
- [Invoking REST API, page 6-9](#)
- [Detailed API Reference, page 6-16](#)
- [Error Messages, page 6-48](#)
- [Quick Reference, page 6-49](#)

Overview

Cisco offers a set of standard REST (Representational State Transfer) APIs and Java stubs for accessing entities defined in Service Portal. They are collectively known as nsAPI.

Authentication is enforced in the nsAPI with session support provided. Access permissions to the entities are governed by the Role-Based Access Control (RBAC) object-level permissions defined for the user in the Service Portal application.

For integrating with external applications, as well as the Portal Manager solution, nsAPI can be used. The portal features support the design and rendering of portlets created using Java, JavaScript, HTML, or Ext JS—the UI framework for the portal. Within such portlets, nsAPI can be invoked to retrieve the required entity information, and allow users to update the data for certain types of entities. More information about the portal module can be found in the *Cisco Service Portal Designer Guide*.

Supported Entities

The entities supported by nsAPI come under the following categories:

Entity Group	Entity Type
Definitional Data	Categories Services Service Offerings Agents
Directory Data	Organizational Units Persons Groups Accounts

Entity Group	Entity Type
Transactional Data	Agreements Requisitions Requisition Entries Authorizations Tasks
Lifecycle Center Data	Service Item Details All Service Items Standards
Portal Designer Data	Custom content tables

Operations

The following types of operations are supported by the nsAPI:

- HTTP GET operations for core entities predefined in the Service Portal application
- HTTP GET operations for user-defined entities in Lifecycle Center and Portal Designer
- HTTP POST operations for specific entities:
 - Person – Create person, update person details
 - Tasks – Perform task actions (done/approve/reject/review)

The GET operations fetch data for the entity specified whereas PUT operations allow modifications to be made on the entity details or status. The user invoking the operations must have the necessary read/write permissions to the affected entity instances.

Conventions and Syntax

- The REST URL follows this convention:

```
http(s)://<serverURL>/RequestCenter/nsapi/<entityGroup>/<entityType>/<filters>?sortBy=
<columnName>&sortDir=<sortOrder>?startRow=<x>&recordSize=<y>
```

where elements enclosed in angle brackets (<>) indicate that an appropriate parameter or parameter value must be substituted.

- Filters, sorting and paging controls and actions are passed as optional parameters in the REST URLs. Filters may include one or multiple expressions as explained in the “Filters” section below. When filters are absent, all instances found for the entity are returned.
- Multiple filters can sometimes be combined. In this case, second and subsequent instances of the parameter are optional, and the syntax diagram indicates this by enclosing the parameter syntax within square brackets ([]). The optional parameters must be enclosed in separators (!).
- Unless otherwise noted, all URLs are case-sensitive.
- Versioning is built into the nsAPI. For the base version 1.0, the REST URLs do not have the version number as a parameter. However, this may change in the future releases.
- HTTP status codes and error messages are returned in the REST responses to show whether the requested operation succeeded or failed.
- nsAPI Java binding packages are named com.newscale.nsapi.*.

Filters

nsAPI supports a variety of filters for GET operations based on the entity type:

Entity Type	Available Filters/Syntax	REST URL Examples
All Entities	Id /id/<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs/id/16
	Name – exact match /name/<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs/name/Create%20Custom%20VM
	Name – wildcard search ?name=<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs?name=Create%20Custom*
Standards, Service Items, Custom Content	<p>Any table column, with up to three filter expressions comprised of the following elements:</p> <ul style="list-style-type: none"> • Comparison Operators: =, >, <, >=, <= • Relational Operators: AND, OR (case-insensitive, order precedence is not supported) • Separator: <p>/<columnName1><operator1><value1>[<AND OR>]<columnName2><operator2><value2>][<AND OR> <columnName3><operator3><value3>]</p> <p><i>Date field values should be in mm-dd-yyyy format.</i></p>	<p>http://serverURL/RequestCenter/nsapi/standard/StOperatingSystem/Custom1=Linux AND Custom2=64</p> <p><i>which can also be written as:</i></p> <p>http://serverURL/RequestCenter/nsapi/standard/StOperatingSystem/Custom1=Linux Custom2=64</p> <p>http://serverURL/RequestCenter/nsapi/standard/StOperatingSystem/Custom1=Linux OR Custom2=64</p> <p>http://serverURL/RequestCenter/nsapi/serviceitems/serviceitemssubscription/SubmitDate=<03-01-2011</p> <p>http://serverURL/RequestCenter/nsapi/customcontent/UcAnnouncementObj/Category=Corporate</p>
Service Items	<p>View Name ?ViewName=<value></p> <p><i>where possible View Names are:</i></p> <ul style="list-style-type: none"> • <i>My ServiceItems (as seen in My Services)</i> • <i>Manage ServiceItems (as seen in Service Item Manager)</i> 	<p>http://serverURL/RequestCenter/nsapi/serviceitem/SiDesktop?ViewName=My%20ServiceItems</p> <p>http://serverURL/RequestCenter/nsapi/serviceitem/SiDesktop?ViewName=Manage%20ServiceItems</p>
Organizational Units	<p>OU Type ?type=<all/businessUnit/serviceTeam></p>	http://serverURL/RequestCenter/nsapi/directory/organizationalunits?type=serviceTeam

Entity Type	Available Filters/Syntax	REST URL Examples
Person	Login Name ?loginname=<value>	http://serverURL/RequestCenter/nsapi/directory/people?loginname/dsmith
	OU Name ?ouname=<value>	http://serverURL/RequestCenter/nsapi/directory/people?ouname=Operations
	Group Name ?groupname=<value>	http://serverURL/RequestCenter/nsapi/directory/people?groupname=Approvers
	Role Name ?rolename=<value> <i>Inherited roles are not available for filtering.</i>	http://serverURL/RequestCenter/nsapi/directory/people?rolename=Service%20Performer
Categories	Category Type ?catalogType=<serviceCatalog/offeringCatalog>	http://serverURL/RequestCenter/nsapi/definition/categories?catalogType=serviceCatalog
Services	Category Name ?categoryName=<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs?categoryName=Manage%20Physical%20Servers
	Keyword ?keywordName=<value>	http://serverURL/RequestCenter/nsapi/definition/servicedefs?keywordName=server
Requisitions and Authorizations	View Name, Status /ViewName=<value1>[Status=<value2>] <i>The view names available for filtering correspond to the list of views in the My Services Requisitions and Authorizations tabs.</i> <i>The status filter must be used in conjunction with the view name filter and cannot be used on its own.</i>	http://serverURL/RequestCenter/nsapi/transaction/requisitions/ViewName=Ordered%20for%20Self http://serverURL/RequestCenter/nsapi/transaction/authorizations/ViewName=Authorizations%20for%20Self http://serverURL/RequestCenter/nsapi/transaction/authorizations/ViewName=Authorizations%20for%20Self Status=Approved

Entity Type	Available Filters/Syntax	REST URL Examples
Tasks	View Name ?viewName=<value> <i>The view names available for filtering correspond to the list of system-defined views in Service Manager. User-defined views are not available for filtering.</i>	http://serverURL/RequestCenter/nsapi/transaction/tasks?viewName=AvailableWork
Tasks – for a specific requisition entry	Task Type ?taskType=<value> <i>where possible Task Types are: all, delivery, authorization</i> Task Status (Skipped) ?showSkippedTasks=<false true>	http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionEntryNumber=1234?taskType=delivery&showSkippedTasks=true

For POST operations, the following filters are supported:

Entity Type	Available Filters	REST URL Examples
Person	Login Name or Id <i>The filter is implicit in the operation and the person identifier is obtained from the request XML.</i>	http://serverURL/RequestCenter/nsapi/directory/people/update <i>The same URL is used for both create and update actions. No person attribute needs to be passed in the URL.</i>
Tasks	Id /<value>/<done approve reject review> <i>The “ ” indicates that exactly one of the possible options must be chosen.</i>	http://serverURL/RequestCenter/nsapi/transaction/tasks/215/approve

About Name Search

For wildcard search in entity name filters (for example, ?name=<value>), leading wildcard characters such as “*” or “%” in the search string are ignored. If these characters are located in the middle or at the end of the string, they are applied in wildcard matching.

To enable the use of a leading wildcard character in name search, locate the property ContainsQueryInFnS in the newscale.properties file (under RequestCenter.war/WEB-INF/classes/config) and set the value to true:

```
ContainsQueryInFnS=true
```

This property also controls full wildcard search support in the Service Manager and Service Link modules. **Such search operations may negatively impact system performance and are generally not recommended in production environments.**

Associated entities are nested entities that are fetched in conjunction with the primary entity—for example, the OU of which a person is a member. Associated entity name filters do not support wildcard search. Searches on those filters return only results with an exact match, and are case-sensitive. In other words, wildcard characters in the search string are treated as literals during matching. For example,

```
/nsapi/directory/people?ouname=star*OU
```

returns people in the organization unit literally named “star*OU”.

```
/nsapi/directory/people?groupname=starGroup*
```

returns people in the group literally named “starGroup*”.

Sorting

Sorting controls are available for operations that return more than one row of data. As with filters, sorting controls are specified as parameters in the REST URL:

```
?sortBy=<columnName>&sortDir=<sortOrder>
```

- **sortBy** – The field name by which sorting is to be done.
- **sortDir** – The direction of sorting. Possible values are **asc** (ascending) and **desc** (descending).

When no sorting parameter is passed in the REST URL, the default sort field and sort order are assumed, as shown in the table below.

Entity Type	Default Sort Field and Sort Order
Categories	Name (Asc)
Services	Name (Asc)
Offerings	Name (Asc)
Agents	Name (Asc)
People	First Name (Asc)
Organizational Units	Name (Asc)
Groups	Name (Asc)
Accounts	Name (Asc)
All Service Items	Row Id* (Asc)
Service Item Details	Row Id* (Asc)
Standards	Row Id* (Asc)
Custom Content	Row Id* (Asc)
Requisitions	Submit Date (Asc), Requisition Id (Asc)
Requisition Entries	Requisition Entry ID (Asc)
Tasks	Task Name (Asc), Task Id (Asc)
Authorizations	Requisition Id (Asc), Task Id (Asc)
Agreements	Agreement Name (Asc)

Row Id is the physical order in which the records are inserted into the service item, standard and custom content tables. This corresponds to the column named PrimaryID in those tables.

Service Items, Standards and Custom Content entities support sorting for all columns. Other entities support sorting for specific fields only. See the [“Detailed API Reference” section on page 6-16](#) for more information.

Examples

```
/nsapi/directory/people?sortBy=lastName&sortDir=asc
```

returns person records listed in ascending order by their last name.

```
/nsapi/directory/people?sortBy=login&sortDir=desc
```

returns person records listed in descending order by their login name.

Paging

Paging controls are available for operations that return more than one row of data. They are also specified as parameters in the REST URL:

```
?startRow=<x>&recordSize=<y>
```

- **startRow** – The starting row from which to fetch the records. The default value is 1.
- **recordSize** – Number of records to be fetched at a time. The default value is set in the Portal Designer common settings. The maximum number of records allowed is 50.

The above parameters and the total number of records returned are shown as attributes in the root tag of the REST XML response, as shown in the example below:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<categories totalCount="11" recordSize="10" startRow="1">
  <category>
    <categoryId>1</categoryId>
    <categoryName><b>Consumer Services</b></categoryName>
    <description />
    . . .
  </category>
  . . .
</categories>
```

If you specify a **startRow** greater than the number of records which would be retrieved, an HTTP 500 error is returned. If you specify a **recordSize** greater than the number of records which would be retrieved, all rows are returned.

Examples

In the following examples, assume there are 60 services defined in Service Designer and the maximum number of records specified for the nsAPI setting in Portal Designer is 30.

Example 1:

```
/nsapi/definition/servicedefs
```

returns the first 30 services. Here is the how the response would look like for such a request:

```
<services totalCount="60" recordSize="30" startRow="1">
  . . .
</services>
```

Example 2:

```
/nsapi/definition/servicedefs?startRow=4
```

returns 30 services, starting from the fourth one. Here is the how the response would look like for such a request:

```
<services totalCount="60" recordSize="30" startRow="4">
  .
  .
  .
</services>
```

Example 3:

```
/nsapi/definition/servicedefs?startRow=4&recordSize=5
```

returns 5 services, starting from the fourth one. Here is the how the response would look like for such a request:

```
<services totalCount="60" recordSize="5" startRow="4">
  .
  .
  .
</services>
```

Example 4:

```
/nsapi/definition/servicedefs?startRow=4&recordSize=35
```

returns 30 services, starting from the fourth one. Here is the how the response would look like for such a request:

```
<services totalCount="60" recordSize="30" startRow="4">
  .
  .
  .
</services>
```

Nested Entities

Certain entities contain a nested structure. The retrieval of only the first-level children or associated entities are supported by nsAPI. Parent entities and their child entities are summarized in the table below.

Parent Entity	Child Entity (Entities)
Categories	Subcategories, Included Services, and Offerings for a category
Services	Categories, Keywords, and Bundled Services associated with a service
Service Offerings	Categories, Keywords, Objectives, Cost Drivers, and Component Services associated with a service offering
Person	OUs, Groups, Roles, Addresses, Contacts, Preferences, and Delegates associated with a person

For category, service or person searches that return more than one row of data in the result set, the associated entities are not fetched for performance reasons. The associated entities are available only in GET by Id or Name operations on the individual entities.

Examples

```
/nsapi/directory/people/loginname/<value>
```

returns the groups of which the person is a member:

```
<person>
  .
  .
  .
  <associatedGroups>
    <associatedGroup>
```

```

        <id>2</id>
        <name>group2</name>
    </associatedGroup>
    <associatedGroup>
        <id>5</id>
        <name>group5</name>
    </associatedGroup>
</associatedGroups>
</person>

/nsapi/directory/people/id/<value>

```

returns the groups to which the person belongs.

```
/nsapi/directory/people
```

does **not** return the groups the person belongs to.

```
/nsapi/directory/people?ouname=<value>
```

does **not** return the groups the person belongs to.

Invoking REST API

Using nsAPI with HTTP Clients

Application authentication is required for invoking nsAPI through browsers or other HTTP clients.

Upon successful login to Service Portal, enter a valid nsAPI REST URL in the browser address bar; for example:

```
http://<serverURL>/RequestCenter/nsapi/definition/categories/id/3
```

The response XML, like the one below, is shown in the browser:

```

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<category>
  <categoryId>3</categoryId>
  <categoryName>Workplace Services</categoryName>
  <description>Services for voice and data communications, desktop, mobile devices, and
application access.</description>
  <topDescriptionEnabled>>false</topDescriptionEnabled>
  <topDescription />
  <middleDescriptionEnabled>>false</middleDescriptionEnabled>
  <middleDescription />
  <bottomDescriptionEnabled>>false</bottomDescriptionEnabled>
  <bottomDescription />
  <catalogTypeId>1</catalogTypeId>
  <catalogType>Consumer Services Catalog</catalogType>
  <isRoot>>false</isRoot>
  <associatedServices>
    <associatedService>
      <description>Order a new or refurbished laptop. Manager approval
required.</description>
      <id>20</id>
      <name>New Laptop</name>
      <status>Active</status>
    </associatedService>
  </associatedService>

```

```

    <description>Order a new iPhone or Blackberry, configured and maintained under
corporate policy.</description>
    <id>22</id>
    <name>New Mobile Device</name>
    <status>Active</status>
  </associatedService>
</associatedServices>
<includedCategories>
  <includedCategory>
    <id>8</id>
    <name>Email</name>
  </includedCategory>
  <includedCategory>
    <id>9</id>
    <name>Laptops</name>
  </includedCategory>
</includedCategories>
<categoryURLSc>
  <a
href='/RequestCenter/myservices/navigate.do?categoryid=3&query=catalog&layout=popu
p_p' onclick="return GB_showFullScreen('Category', this.href)">Workplace Services</a>
  </categoryURLSc>
  <categoryURLOnlySc>/RequestCenter/myservices/navigate.do?categoryid
=3&query=catalog</categoryURLOnlySc>
</category>

```

If a REST API request was executed before logging into the application, the URL would return the following error:

```
HTTP Error 401 Unauthorized
```

For HTTP requests made through other clients, the authentication credentials can be passed as HTTP header parameters to the nsAPI Login URL:

```
http://<serverURL>/RequestCenter/nsapi/authentication/login
```

The **HTTP Header** must include the following parameters:

```
username=<username>
```

```
password=<password>
```

Upon successful authentication, a JSessionID cookie is returned in the HTTP response. Subsequent invocations of nsAPI should include the same JSessionID cookie in the request to retain the session without having to authenticate again.

Using nsAPI with JavaScript Portlets

nsAPI calls can be invoked in JavaScript portlets developed on the Portal Designer module. Using JavaScript, REST URL, AJAX, or Ext JS components, a portlet can be created to retrieve the data for the desired entities and rendered in a grid format.

Render Data in Ext JS Grid

Here is an example of using Ext JS to render a list of categories:

```

/* create the Data Store */
var store = new Ext.data.Store(<
  /* load using HTTP */

```



```

URL :
'http://<serverURL>/RequestCenter/nsapi/definition/servicecatalog/categories',

/* the return is XML, so let's set up a reader */
reader : new Ext.data.XmlReader(<
    root : "categorys",
    record : "category"
>, [<
    name : "id"
>, <
    name : "name"
>, <
    name : "description"
>])
>);

/* create the grid */
var grid = new Ext.grid.GridPanel(<
    store : store,
    columns : [<
        header : "Category ID",
        dataIndex : 'id',
        sortable : true
    >, <
        header : "Category Name",
        dataIndex : "name",
        sortable : true
    >, <
        header : "Description",
        dataIndex : "description",
        sortable : true
    >],
    renderTo : '#divName#',
    width : "100%",
    autoHeight : true,
    layout : 'fit',
    viewConfig : <
        forceFit : true
    >,
    bbar : new Ext.PagingToolbar(<
        pageSize : 25,
        store : store,
        displayInfo : true,
        displayMsg : 'Displaying topics <0> - <1> of <2>',
        emptyMsg : "No topics to display"
    >)
>);

store.load();

```

Get Logged-In User

Namespace variables for the currently logged-in user are available to be used in the JavaScript portlet; for example:

```

Ext.onReady(function() {
    /* Demonstrate JavaScript to get Logged-In user details */
    alert('PersonId: ' + nsAPP_CurrentUserId);
    alert('Login name: ' + nsAPP_CurrentUserLoginName);
    alert('First name: ' + nsAPP_CurrentUserFirstName);
    alert('Last name: ' + nsAPP_CurrentUserLastName);

```

```

        alert('HomeOUId: ` +nsAPP_CurrentUserHomeOuId);
    }

```

Using nsAPI with JSR Portlets

Authentication

When accessed through JSR portlets, the nsAPI Java client can be used to invoke the login and logout operations.

```

import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
. . .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter", "username", "password" ); // Login
by username, password.
. . .
nsApiClient.logout(); // Logout

. . .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter",
sessionId); // Login using current session id.
. . .
nsApiClient.logout();

```

Get Logged-In User

Here is an example of Spring-based JSR Portlet Controller to get the details of the logged-in user:

```

import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;
import org.springframework.ui.Model;
import org.springframework.stereotype.Controller;
import com.newscale.nsapi.directory.person.Person;
...
@Controller
public class MyJSRController {
    private NSApiClient nsApiClient = getNSApiClient();
    public NSApiClient getNsApiClient() {
        return nsApiClient;
    }
    public void setNsApiClient(NSApiClient nsApiClient) {
        this.nsApiClient = nsApiClient;
    }
    @RequestMapping("VIEW")
    @RenderMapping("NORMAL")
    public String viewNormal(RenderRequest request, RenderResponse response, Model model) {
        nsApiClient.login("http://<AppServer host>:<port>/RequestCenter",
request.getPortletSession().getId());
        // Get Currently Logged-in user from nsAPI client
        Person persons = nsApiClient.getDirectory().getCurrentUser();
        // Get user info
        long personId = persons.getPersonId();
        long homeOUId = persons.getHomeOrganizationalUnitId();
        String firstName =persons.getFirstName();
    }
}

```

```
String lastName =persons.getLastName());
String username =persons.getLogin();
    }
}
```

Get Operations

Here are some sample code snippets that illustrate the methods for fetching entities. For further details on these methods, see the Javadoc for the individual entity classes.

- Get person by Id

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapiclient.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
. . .

NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter", "username", "password");

Person person = nsApiClient.getDirectory().getPersonById(123);

/* This is the equivalent of REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people/id/123
*/
. . .
nsApiClient.logout();
```

- Get person by Name

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapiclient.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
. . .

NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter", "username", "password");
Person person = nsApiClient.getDirectory().getPersonByLoginName("jsmith");
/*
This is the equivalent of REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people/loginname/jsmith
*/
. . .
nsApiClient.logout();
```

- Get all people

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapiclient.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
import java.util.List;
import org.apache.commons.collections.map.MultiValueMap;
. . .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
```

```

nsApiClient.login("http://<serverURL>/RequestCenter", "username", "password");

PersonList persons = nsApiClient.getDirectory().getPeople(null);

/*
  This is the equivalent of REST URL
  http://<serverURL>/RequestCenter/nsapi/directory/people
*/
. . .
nsApiClient.logout();

```

- Get person with query filter

```

package com.newscale.nsapiclient.directory;
import com.newscale.nsapiclient.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClientConstants;
import com.newscale.nsapiclient.NSApiClient;
import org.apache.commons.collections.map.MultiValueMap;

import java.util.List;

NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter", "username", "password");

MultiValueMap filterMap = new MultiValueMap();//this can be used to specify multiple
filter criteria
filterMap.put(NSApiClientConstants.QUERYPARAM_NAME, "John");
PersonList persons = nsApiClient.getDirectory().getPeople(filterMap);

/*
  This is the equivalent of REST URL
  http://<serverURL>/RequestCenter/nsapi/directory/people?name=John
*/
. . .
nsApiClient.logout();

```

- Get person with multiple query filters

```

package com.newscale.nsapiclient.directory;
import com.newscale.nsapiclient.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClientConstants;
import com.newscale.nsapiclient.NSApiClient;
import org.apache.commons.collections.map.MultiValueMap;

import java.util.List;
. . .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter", "username", "password");

MultiValueMap filterMap = new MultiValueMap();
filterMap.put(NSApiClientConstants.QUERYPARAM_NAME, "John");
filterMap.put(NSApiClientConstants.QUERYPARAM_SORTBY, "lastName");
filterMap.put(NSApiClientConstants.QUERYPARAM_SORTDIR, "asc");
PersonList persons = nsApiClient.getDirectory().getPeople(filterMap);

// This is the equivalent of REST URL
// http://<serverURL>/RequestCenter/nsapi/directory/people?name
// =John&sortBy=lastName&sortDir=asc

```

```
// search for people by the name John,
// sort the result set by Last Name in ascending order
. . .
nsApiClient.logout();
```

Post Operations

Here are some sample code snippets that illustrate the methods for creating/updating a person and taking actions on tasks. For further details on these methods, see the Javadoc for the individual entity classes.

- Update person

```
package com.newscale.nsapiclient.directory;
import com.newscale.nsapiclient.directory.person.Person;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.directory.Directory;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClient;
. . .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter", "username", "password");
Person person = NSApiClient.getDirectory().getPersonById(123);
person.setLastName("Smith");
Person persons = NSApiClient.getDirectory().updatePerson(person);
/*
This is the equivalent of posting XML of person 123 with last name changed to "Smith" to
the REST URL
http://<serverURL>/RequestCenter/nsapi/directory/people/update
*/
. . .
nsApiClient.logout();
```

- Complete delivery task

```
package com.newscale.nsapiclient.transaction;
import com.newscale.nsapiclient.transaction.task.TaskAction;
import com.newscale.nsapi.NSApiConstants;
import com.newscale.nsapiclient.transaction.Transaction;
import com.newscale.nsapiclient.NSApiClientFactory;
import com.newscale.nsapiclient.NSApiClientConstants;
import com.newscale.nsapiclient.NSApiClient;
import org.apache.commons.collections.map.MultiValueMap;

import java.util.List;
. . .
NSApiClient nsApiClient = NSApiClientFactory.getInstance();
nsApiClient.login("http://<serverURL>/RequestCenter", "username", "password");

// This is the equivalent of REST URL
// http://<serverURL>/RequestCenter/nsapi/tasks/10/done

NSApiClient.getTransaction().completeTask(10);
. . .
nsApiClient.logout();
```

Detailed API Reference

The javadoc for nsAPI can be located in the Image folder of the product installer. Examples for invoking the nsAPI through REST URL and Java client are provided below for each of the supported entities.

Definitional Data

Categories

Area	Examples
Core API	<p>Get all categories</p> <p>By default only categories of type “Service Catalog” are returned.</p> <p>REST URL: <a href="http://<ServerURL>/RequestCenter/nsapi/definition/categories">http://<ServerURL>/RequestCenter/nsapi/definition/categories</p> <p>Java Example: <pre>CategoryList categories = NSApiClient.getDefinition().getCategories(null);</pre></p> <hr/> <p>Gets all consumer service categories</p> <p>Returns all categories of “Consumer Services” type.</p> <p>REST URL: <a href="http://<ServerURL>/RequestCenter/nsapi/definition/categories">http://<ServerURL>/RequestCenter/nsapi/definition/categories</p> <p>Java Example: <pre>CategoryList categories = NSApiClient.getDefinition().getCategories("serviceCatalog");</pre></p>
	<p>Gets all service offering categories</p> <p>Returns all categories of “Service Offerings” type</p> <p>REST URL: <a href="http://<ServerURL>/RequestCenter/nsapi/definition/categories?catalogType=offeringCatalog">http://<ServerURL>/RequestCenter/nsapi/definition/categories?catalogType=offeringCatalog</p> <p>Java Example: <pre>CategoryList categories = NSApiClient.getDefinition().getCategories("offeringCatalog");</pre></p>
	<p>Get consumer service category by Name</p> <p>Returns a service catalog category with the name specified. Nested entities (subcategories and included services) are fetched.</p> <p>REST URL: <a href="http://<ServerURL>/RequestCenter/nsapi/definition/categories/name/<categoryName>">http://<ServerURL>/RequestCenter/nsapi/definition/categories/name/<categoryName></p> <p>Java Example: <pre>Category categories = NSApiClient.getDefinition().getCategoryByName("<categoryName>");</pre></p>

Area	Examples
	<p>Gets consumer service category by Id</p> <p>Returns a service catalog category with the Id specified. Nested entities (subcategories and included services) are fetched.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/categories/id/<categoryId></code></p> <p>Java Example:</p> <pre>Category categories = NSApiClient. getDefinition().getCategoriesById(<categoryId>);</pre> <hr/> <p>Gets service offering category by Name</p> <p>Nested entities (subcategories and included offerings) are included. Returns a service offering category with the name specified.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/categories/name/<categoryName>?catalogType=offeringCatalog</code></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put(QUERYPARAM_CATALOG_TYPE, "offeringCatalog"); Category categories = NSApiClient.getDefinition().getCategoryByName("<categoryName>");</pre>
Filters	<p>Category Name Filters</p> <p>Search is case sensitive.</p> <p>StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored.</p> <p>Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard is applied.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/categories?name=<wildcardValue>&catalogType=serviceCatalog</code></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put(QUERYPARAM_CATALOG_TYPE, "serviceCatalog"); paramsMap.put(QUERYPARAM_NAME, "<wildcardValue>"); CategoryList categories = NSApiClient.getDefinition().getCategories(paramsMap);</pre>

Area	Examples
Sort Column(s)	Category Name
Response XML	<pre><categories totalCount="x" recordSize="y" startRow="z"> <category> . . . <associatedServices> . . . <associatedService> <description> </description> <id></id> <imageUrl></imageUrl> <name> </name> <status> </status> </associatedService> . . . </associatedServices> </category> </categories></pre>

Services

Area	Examples
CoreAPI	<p>Get all services</p> <p>Returns all services.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs</code></p> <p>Java Example:</p> <pre>ServiceList services = NSApiClient.getDefinition().getServices(null);</pre> <hr/> <p>Get all services by service catalog wildcard search</p> <p>Returns all services with service name, service description, category, or keyword that matches the search string. Leading wildcard is not supported.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?search={ wildcardValue * }</code></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put("search", "<wildcardValue*>"); ServiceList services = NSApiClient.getDefinition().getServices(paramsMap);</pre>

Area	Examples
	<p>Get service by Id</p> <p>Nested entities (associated categories and keywords) are fetched for getById and getName only.</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/id/<serviceId></p> <p>Java Example:</p> <pre>Service services = NSApiClient.getDefinition().getServiceById(<serviceId>);</pre>
	<p>Get service by Name</p> <p>Nested entities (associated categories and keywords) are fetched for getById and getName only.</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs/name/<serviceName></p> <p>Java Example:</p> <pre>Service services = NSApiClient.getDefinition().getServiceByName("<serviceName>");</pre>
	<p>Get all services in a category</p> <p>Returns all services associated with the category</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?CategoryName=<categoryName></p> <p>Java Example:</p> <pre>ServiceList services =NSApiClient.getDefinition().getServiceByCategoryName("<categoryName>");</pre>
	<p>Gets all services by keyword</p> <p>Returns all services associated with the keyword</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?KeywordName=<keyword></p> <p>Java Example:</p> <pre>ServiceList services = NSApiClient.getDefinition().getServiceByKeyword("<keyword>");</pre>

Area	Examples
Filters	<p>Services Name Filters</p> <p>Search is case sensitive.</p> <p>StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored.</p> <p>Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard should be supplied if needed.</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/definition/servicedefs?name=<wildcardValue></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put("name", "<wildcardValue*>"); ServiceList servicesList = NSApiClient.getDefinition().getServices(paramsMap)</pre>
Sort Column(s)	Service Name
Response XML	<pre><services totalCount="x" recordSize="y" startRow="z"> <service> . . . <includedServices> . . . <includedService> <id></id> <name> </name> </includedService> . . . </includedServices> . . . </service> </services></pre>

Service Offerings

Area	Examples
CoreAPI	<p>Get all service offerings</p> <p>Returns all service offerings.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/serviceofferings</code></p> <p>Java Example:</p> <pre>BusinessServiceList offerings = NSApiClient.getDefinition().getBusinessServices(null);</pre> <hr/> <p>Get service offering by Id</p> <p>Nested entities (associated categories and keywords) are fetched for <code>getById</code> and <code>getByName</code> only.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/serviceofferings/id/<offeringId></code></p> <p>Java Example:</p> <pre>BusinessService offerings = NSApiClient.getDefinition().getBusinessServiceById(<offeringId>);</pre> <hr/> <p>Get service offering by Name</p> <p>Nested entities (associated categories and keywords) are fetched for <code>getById</code> and <code>getByName</code> only.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/serviceofferings/name/<offeringName></code></p> <p>Java Example:</p> <pre>BusinessService of= NSApiClient.getDefinition().getBusinessServiceByName("<offeringName>");</pre>
Filters	<p>Offering Name Filters</p> <p>Search is case sensitive.</p> <p><code>StartsWith</code> (<code>ContainsQueryInFnS=false</code> in <code>newscale.properties</code>): leading wildcard is ignored.</p> <p><code>Contains</code> (<code>ContainsQueryInFnS=true</code> in <code>newscale.properties</code>): leading wildcard should be supplied if needed.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/serviceofferings?name=<wildcardValue></code></p> <p>Java Example:</p> <pre>BusinessServiceList offer = NSApiClient.getDefinition().getBusinessServicesByFilter("<wildcardValue>");</pre>

Area	Examples
Sort Column(s)	Offering Name
Response XML	<pre><serviceOfferings totalCount="x" recordSize="y" startRow="z"> <serviceOffering> . . . </serviceOffering> </serviceOfferings></pre>

Agents

Area	Examples
CoreAPI	<p>Get all agents</p> <p>Returns all agents.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/agents</code></p> <p>Java Example:</p> <pre>AgentList agents = NSApiClient.getDefinition().getAgents(null);</pre> <hr/> <p>Get agent by Id</p> <p>Outbound and inbound properties are fetched for <code>getById</code> and <code>getByName</code> only.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/agents/id/<agentId></code></p> <p>Java Example:</p> <pre>Agent agents = NSApiClient.getDefinition().getAgentById(<agentId>);</pre> <hr/> <p>Get agent by Name</p> <p>Outbound and inbound properties are fetched for <code>getById</code> and <code>getByName</code> only.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/agents/name/<agentName></code></p> <p>Java Example:</p> <pre>Agent Agents = NSApiClient.getDefinition().getAgentByName("<agentName>");</pre>
Filters	<p>Agent Name Filters</p> <p>Search is case sensitive.</p> <p><code>StartsWith</code> (<code>ContainsQueryInFnS=false</code> in <code>newscale.properties</code>): leading wildcard is ignored.</p> <p><code>Contains</code> (<code>ContainsQueryInFnS=true</code> in <code>newscale.properties</code>): leading wildcard should be supplied if needed.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/agents?name=<wild cardValue></code></p> <p>Java Example:</p> <pre>AgentList agents = NSApiClient.getDefinition().getAgentsByFilter("<wildcardValue>");</pre>

Area	Examples
Sort Column(s)	Agent Name
Response XML	<pre><agents totalCount="x" recordSize="y" startRow="z"> <agent> . </agent> </agents></pre>

Agreements

Area	Examples
CoreAPI	<p>Gets all agreements</p> <p>Returns all agreements.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/agreements</code></p> <p>Java Example:</p> <pre>AgreementList agreements = NSApiClient.getDefinition().getAgreements(null);</pre> <hr/> <p>Get agreement by Id</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/agreements/id/<agreementId></code></p> <p>Java Example:</p> <pre>Agreement agreements = NSApiClient.getDefinition().getAgreementById(<agreementId>);</pre> <hr/> <p>Get agreement by Name</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/agreements/name/<agreementName></code></p> <p>Java Example:</p> <pre>Agreement agreements = nsclient.getDefinition().getAgreementByName("<agreementName>");</pre>
Filters	<p>Agreement Name Filters</p> <p>Search is case sensitive.</p> <p>StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored.</p> <p>Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard should be supplied if needed.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/definition/agreements?name=<wildcardValue></code></p> <p>Java Example:</p> <pre>AgreementList agreements = NSApiClient.getDefinition().getAgreementsByFilter ("<wildcardValue>");</pre>

Area	Examples
Sort Column (s)	Agreement Name
Response XML	<pre><agreements totalCount="x" recordSize="y" startRow="z"> <agreement> . . . </agreement> </agreements></pre>

Directory Data

Person

Area	Examples
CoreAPI	<p>Get all people</p> <p>Returns all people.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/people</code></p> <p>Java Example:</p> <pre>PersonList person = NSApiClient.getDirectory().getPeople(null);</pre>
	<p>Get person by Id</p> <p>Returns the person with the specified Person Id.</p> <p>Nested entities (OUs, Groups, Roles, Addresses, Contact, and Preferences) are fetched.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/people/id/<personId></code></p> <p>Java Example:</p> <pre>Person persons = NSApiClient.getDirectory().getPersonById(<personId>);</pre>
	<p>Get person by LoginName</p> <p>Returns the person with the LoginName specified.</p> <p>Nested entities (OUs, Groups, Roles, Addresses, Contact, and Preferences) are fetched.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/people/loginname/<loginName></code></p> <p>Java Example:</p> <pre>Person persons = NSApiClient.getDirectory().getPersonByLoginName("<loginName>");</pre>

Area	Examples
	<p>Get logged-in user</p> <p>Returns the person who is currently logged in.</p> <p>REST URL:</p> <p><code>http://<serverURL>/RequestCenter/nsapi/directory/people/currentuser</code></p> <p>Java Example:</p> <pre>Person person = NSApiClient.getDirectory().getCurrentUser();</pre>
Special Conditions	<p>Create/Update Person</p> <p>Obtain person XML from the response of any of the GET person REST URLs mentioned above.</p> <p>POST person XML to the update person REST URL to create or modify a person.</p> <p>If the person exists (identified by login or personId in the XML), an update operation is performed; otherwise a new person is created.</p> <p>In the create operation, the following five elements are required:</p> <ul style="list-style-type: none"> • firstName • lastName • homeOrganizationalUnitName • email • login <p>In the create operation, the password of the person is the set to the login name.</p> <p>Changes to the Home OU in update operation replace the Home OU of the Person. Other associated OUs, Groups, and Roles are ignored in both the create and update operations.</p> <p>For Home OU, TimeZone, Locale, Supervisor, Authorization Delegate, Login Module, Contact, and Address attributes, the following rules apply:</p> <ul style="list-style-type: none"> • If an id element is sent in the XML but not found in the database, an exception with proper message is thrown. • If the id element is not found, the Name element sent in the XML is used instead. An exception is thrown if the name is not found in the database. • If neither id nor name element is in the XML and the attribute is optional, the attribute is ignored in the create/update operation (HomeOU is mandatory). • If the create/update operation fails due to incorrect or missing data sent in the XML, the HTTP status code returns “422 Unprocessable Entity”. <p>POST REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/people/update</code></p> <p>Java Example:</p> <pre>Person person = NSApiClient.getDirectory().getPersonById(<personId>); person.setLastName("<lastName>"); Person persons = NSApiClient.getDirectory().updatePerson(person);</pre>

Area	Examples
Filters	<p>OU Name Filters</p> <p>Search is case sensitive and uses exact match .</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/directory/people?ouname=<ouName></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put("ouname", "<ouName>"); PersonList person = NSApiClient.getDirectory().getPeople(paramsMap);</pre>
	<p>Group Name Filters</p> <p>Search is case sensitive and uses exact match.</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/directory/people?groupname=<groupName></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put("groupname", "<groupName>"); PersonList person = NSApiClient.getDirectory().getPeople(paramsMap);</pre>
	<p>Role Name Filters</p> <p>Search is case sensitive and uses exact match.</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/directory/people?rolename=<roleName></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put("rolename", "<roleName>"); PersonList person = NSApiClient.getDirectory().getPeople(paramsMap);</pre>
Sort Column(s)	First Name, Last Name, Login Name
XML Response	<pre><people totalCount="x" recordSize="y" startRow="z"> <person> . . . <contacts> . . . <contact> <contactId></contactId> <contactType></contactType> <contactTypeId></contactTypeId> <value></value> </contact> . . . </contacts> . . . </person> </people></pre>

Organizational Unit

Area	Examples
CoreAPI	<p>Get all organizational units</p> <p>Returns all organizational units.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits</code></p> <p>Java Example:</p> <pre>OrganizationalUnitList Ou = NSApiClient.getDirectory().getOrganizationalUnits(null);</pre> <hr/> <p>Get organizational unit by Id</p> <p>Nested entities (suborganizational units) are fetched for <code>getById</code> and <code>getByName</code> only.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/id/<ouId></code></p> <p>Java Example:</p> <pre>OrganizationalUnit Ou = NSApiClient.getDirectory().getOrganizationalUnitById(ouId);</pre> <hr/> <p>Get organizational unit by Name</p> <p>Nested entities (suborganizational units) are fetched for <code>getById</code> and <code>getByName</code> only.</p> <p>The Service Team OU is returned if two OUs with the same name but of different types are found.</p> <p>The optional parameter <code>?type=<businessUnit/serviceTeam></code> may also be specified (“all” is not allowed in the type parameter value).</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits/name/<ouName></code></p> <p>Java Example:</p> <pre>OrganizationalUnit Ou = NSApiClient.getDirectory().getOrganizationalUnitByName("<ouName>");</pre> <hr/> <p>Get OUs by Type</p> <p>Returns all organizational units of the OU type specified.</p> <p>Possible values: “all”, “businessUnit”, “serviceTeam”.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits?type=<ouType></code></p> <p>Java Example:</p> <pre>OrganizationalUnitList Ou = NSApiClient.getDirectory().getOrganizationalUnitByType("<ouType>");</pre>

Area	Examples
Filters	<p>Organizational Unit Name Filters</p> <p>Search is case sensitive.</p> <p>StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored.</p> <p>Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard should be supplied if needed.</p> <p>The option parameter ?type=<businessUnit/serviceTeam> may also be specified with wildcard name search.</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/directory/organizationalunits?name=<wildcard Value></p> <p>Java Example:</p> <pre>OrganizationalUnitList Ou = NSApiClient.getDirectory().getOrganizationalUnitsByFilter("<wildcardValue>");</pre>
Sort Column(s)	Organizational Unit Name
Response XML	<pre><organizationalunits totalCount="x" recordSize="y" startRow="z"> <organizationalunit> . . . </organizationalunit> </organizationalunits></pre>

Groups

Area	Examples
CoreAPI	<p>Get all groups</p> <p>Returns all groups.</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/directory/groups</p> <p>Java Example:</p> <pre>GroupList groups = NSApiClient.getDirectory().getGroups(null);</pre> <p>Get group by Id</p> <p>Nested entities (subgroups) are fetched for getById and getName only.</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/directory/groups/id/<groupId></p> <p>Java Example:</p> <pre>Group groups = NSApiClient.getDirectory().getGroupsById(<groupId>);</pre>

Area	Examples
	<p>Get group by Name</p> <p>Nested entities (subgroups) are fetched for getById and getName only.</p> <p>REST URL: <a href="http://<ServerURL>/RequestCenter/nsapi/directory/groups/name/<groupName>">http://<ServerURL>/RequestCenter/nsapi/directory/groups/name/<groupName></p> <p>Java Example: <pre>Group groups = NSApiClient.getDirectory().getGroupsByName("<groupName>");</pre></p>
Filters	<p>Group Name Filters</p> <p>Search is case sensitive.</p> <p>StartsWith (ContainsQueryInFnS=false in newscale.properties): leading wildcard is ignored.</p> <p>Contains (ContainsQueryInFnS=true in newscale.properties): leading wildcard should be supplied if needed.</p> <p>REST URL: <a href="http://<ServerURL>/RequestCenter/nsapi/directory/groups?name=<wild cardValue>">http://<ServerURL>/RequestCenter/nsapi/directory/groups?name=<wild cardValue></p> <p>Java Example: <pre>GroupList groups = NSApiClient.getDirectory().getGroupsByFilter("<wildcardValue>");</pre></p>
Sort Column(s)	Group Name
Response XML	<pre><groups totalCount="x" startRow="y" recordSize="z"> <group> . . </group> </groups></pre>

Accounts

Area	Examples
CoreAPI	<p>Get all Accounts</p> <p>Returns all accounts.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/accounts</code></p> <p>Java Example:</p> <pre>AccountList accounts = NSApiClient.getDirectory().getAccounts(null);</pre>
	<p>Get account by Id</p> <p>Nested entities (associated organizational units) are fetched for <code>getById</code> and <code>getByName</code> only.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/accounts/id/<accountId></code></p> <p>Java Example:</p> <pre>Account accounts = NSApiClient.getDirectory().getAccountsById(<accountId>);</pre>
	<p>Get account by Name</p> <p>Nested entities (associated organizational units) are fetched for <code>getById</code> and <code>getByName</code> only.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/accounts/name/<accountName></code></p> <p>Java Example:</p> <pre>Account accounts = NSApiClient.getDirectory().getAccountsByName("<accountName>");</pre>
Filters	<p>Group Name Filters</p> <p>Search is case sensitive.</p> <p><code>StartsWith</code> (<code>ContainsQueryInFnS=false</code> in <code>newscale.properties</code>): leading wildcard is ignored.</p> <p><code>Contains</code> (<code>ContainsQueryInFnS=true</code> in <code>newscale.properties</code>): leading wildcard should be supplied if needed.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/directory/accounts?name=<wild cardValue></code></p> <p>Java Example:</p> <pre>AccountList accounts = NSApiClient.getDirectory().getAccountsByFilter("<wildcardName>");</pre>
Sort Column(s)	Account Name
Response XML	<pre><accounts totalCount="x" startRow="y" recordSize="z"> <account> . . . </account> </accounts></pre>

Transactional Data

Requisitions

Area	Examples
CoreAPI	<p>Get requisitions for the current user</p> <p>Get requisitions with the default filter, that is:</p> <p>ViewName = Ordered for Self</p> <p>Status = Ongoing</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions</code></p> <p>Java Example:</p> <pre>RequisitionList requisitions = NSApiClient.getTransaction().getRequisitions(null);</pre> <hr/> <p>Get requisition by Id</p> <p>RBAC checking is applied against logged-in user.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/id/<requisitionId></code></p> <p>Java Example:</p> <pre>Requisition requisitions = NSApiClient.getTransaction().getRequisitionsById(<requisitionId>);</pre>

Area	Examples
Filters	<p>View and Status Filters</p> <p>The views and statuses available for filtering correspond to those in the My Services Requisitions tabs.</p> <p>If Status is not specified, “Ongoing” is used.</p> <p>If ViewName is not specified, “Ordered for Self” is used.</p> <p>Possible values for ViewName:</p> <ul style="list-style-type: none"> • Ordered for Self – Requisitions for the current user. • Ordered for Others – Requisitions submitted by the current user for others (via Order on Behalf). • Ordered for my unit – Requisitions for people in the OUs the current user belongs to (the view returns data for other people only if the user has the “See Requisitions for My Business Units” capability). <p>Possible values for Status: Ongoing, Preparation, Ordered, Closed, Cancelled, Rejected, All.</p> <p>If an incorrect value is given, the default values (“Ongoing” and “Ordered for Self”) are used.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/requisitions/ViewName=<viewName>[AND Status=<status>]</code></p> <p>Java Example:</p> <pre>String filterString = "ViewName=<viewName> AND Status=<status>"; RequisitionList requisitions = NSApiClient.getTransaction().getRequisitionsByFilter(filterString);</pre>
Sort Column(s)	Customer Name, Owner (Initiator) Name, Requisition ID, Service Name, Started Date, Status, Submit Date
Response XML	<pre><requisitions totalCount="x" recordSize="y" startRow="z"> <requisition> . . . </requisition> </requisitions></pre>

Requisitions Entries

Area	Examples
CoreAPI	<p>Get Requisition Entry By Id</p> <p>RBAC checking is applied against the current user.</p> <p>REST URL: <a href="http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/id/<reqEntryId>">http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/id/<reqEntryId></p> <p>Java Example:</p> <pre>RequisitionEntry requisitonEntry = NSApiClient.getTransaction().getRequisitionEntryById(<reqEntryId>);</pre>
	<p>Get Requisition Entries By Requisition Id</p> <p>RBAC checking is applied against the current user.</p> <p>REST URL: <a href="http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/RequisitionNumber=<reqId>">http://<ServerURL>/RequestCenter/nsapi/transaction/requisitionentries/RequisitionNumber=<reqId></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put("startRow", "1"); //optional paramsMap.put("recordSize", "10"); //optional String filterString = "RequisitionNumber=" + <reqId>; RequisitionEntryList requisitonEntries = NSApiClient.getTransaction().getRequisitionEntries(paramsMap, filterString);</pre>
Filters	Not applicable
Sort Column(s)	Due On, Requisition Entry ID
Response XML	<pre><requisitionEntries totalCount="x" recordSize="y" startRow="z"> <requisitionEntry> . . . </requisitionEntry> </requisitionEntries></pre>

Authorizations

Area	Examples
CoreAPI	<p>Get authorizations for the current user</p> <p>Gets authorizations with the default filter, that is:</p> <p>ViewName = Authorizations for Self</p> <p>Status = Ongoing</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations</code></p> <p>Java Example:</p> <pre>AuthorizationList authorizations = NSApiClient.getTransaction().getAuthorizations(null);</pre> <hr/> <p>Get authorization by Id</p> <p>RBAC checking is applied against the current user.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations/id/<taskId></code></p> <p>Java Example:</p> <pre>Authorization authorizations = NSApiClient.getTransaction().getAuthorizationsById(<taskId>);</pre> <hr/> <p>Get related authorization tasks (approval chain) by Id</p> <p>RBAC checking is applied against the current user.</p> <p>If the Id passed is for an authorization task at the requisition level (departmental authorization, departmental review, financial authorization), all requisition-level authorization tasks of that particular requisition are returned.</p> <p>If the Id passed is for an authorization task at the requisition entry level (service group authorization, service group review), all requisition entry-level authorization tasks of that particular requisition entry are returned.</p> <p>The default sorting is in descending order of Due On.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations?taskId=<taskId></code></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put("startRow", "" + 1); //optional paramsMap.put("recordSize", "" +10); //optional paramsMap.put("sortBy", "dueOn"); //optional paramsMap.put("sortDir", "asc"); //optional paramsMap.put("taskId", "<taskId>"); Authorization authorizations = NSApiClient.getTransaction().getAuthorizations (paramsMap);</pre>

Area	Examples
Filters	<p>View and Status Filters</p> <p>The views and statuses available for filtering correspond to those in the My Services Authorizations tabs.</p> <p>If Status is not specified, it is set to “Ongoing”.</p> <p>If ViewName is not specified, it is set to “Authorizations for Self”.</p> <p>Possible values for ViewName:</p> <ul style="list-style-type: none"> • Authorizations for Self – Authorizations assigned to the current user. • Assigned and Unassigned Authorizations for Self – Authorizations assigned to the current user or the queues which the user has access to. • Authorizations for Others – Authorizations assigned to people in the OUs the current user belongs to (the view returns data only if the user has the “See Authorizations for My Business Units” capability). <p>Possible values for Status: Ongoing, Approved, Rejected, Canceled, Reviewed, All.</p> <p>If an incorrect value is given the default values (“Ongoing” and “Authorizations for Self”) are used.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/authorizations/ViewName=<viewName>[AND Status=<status>]</code></p> <p>Java Example:</p> <pre>String filterString = "ViewName=<viewName> AND Status=<status>"; AuthorizationList authorizations = NSApiClient.getTransaction().getAuthorizationsWithFilters(filterString);</pre>
Sort Column(s)	Customer Name, Due On, Performer Name, Priority, Requisition ID
Response XML	<pre><authorizations totalCount="x" recordSize="y" startRow="z" /> <authorization> . . . </authorization> </authorizations></pre>

Tasks

Area	Examples
CoreAPI	<p>Get tasks for in the current user</p> <p>Gets tasks with the default filter, that is:</p> <p>ViewName = AvailableWork</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/tasks</code></p> <p>Java Example:</p> <pre>TaskList tasks = NSApiClient.getTransaction().getDeliveryTasks(null);</pre> <hr/> <p>Get task by Id</p> <p>RBAC checking is applied against the current user.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/id/<taskId></code></p> <p>Java Example:</p> <pre>TaskFull tasks = NSApiClient.getTransaction().getDeliveryTaskById(<taskId>);</pre>
	<p>Get tasks by Requisition Entry Id</p> <p>RBAC checking is applied against the current user.</p> <p>The default sorting is in ascending order of Activity Id.</p> <p>Parameters and possible values:</p> <ul style="list-style-type: none"> • showSkippedTasks: false (default), true • taskType: all (default), delivery, authorization • showNestedTasks: false, true (default) <p>When the parameter is set to true, delivery tasks are returned with the nested parent-child hierarchy maintained in the XML structure.</p> <ul style="list-style-type: none"> • showChildDeliveryPlan: false (default), true <p>This applies to bundle services only. When the parameter is set to true, the delivery tasks for all included services are also returned.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionEntryNumber=<reqEntryId>?taskType=<taskType>&showSkippedTasks=<false true>&showNestedTasks=<false true>&showChildDeliveryPlan=<false true></code></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put("taskType", "delivery"); //optional paramsMap.put("showSkippedTasks", "true"); //optional paramsMap.put("showNestedTasks", "true"); //optional paramsMap.put("sortBy", "dueOn"); //optional paramsMap.put("sortDir", "desc"); //optional String filterString = "RequisitionEntryNumber=" + <reqEntryId>; TaskList tasks = NSApiClient.getTransaction().getAuthAndDeliveryTasks (paramsMap, filterString);</pre>

Area	Examples
	<p>Get milestones by Requisition Id</p> <p>RBAC checking is applied against the current user.</p> <p>Delivery process milestones (reviews, authorizations, delivery projects) are returned in the chronological order. No sorting and paging is supported.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/RequisitionNumber=<reqId></code></p> <p>Java Example:</p> <pre>String filterString = "RequisitionNumber=" + <reqId>; MilestoneList milestones = NSApiClient.getTransaction().getDeliveryProcessForMilestone (filterString);</pre>
Special Conditions	<p>Approve Tasks</p> <p>Perform an HTTP POST with the action and task Id in the REST URL.</p> <p>POST REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/tasks /<taskId>/approve</code></p> <p>Java Example:</p> <pre>TaskAction approve = NSApiClient.getTransaction().approveTask(<taskId>;</pre> <hr/> <p>Reject Tasks</p> <p>Perform an HTTP POST with the action and task Id in the REST URL.</p> <p>POST REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/<taskId>/reject</code></p> <p>Java Example:</p> <pre>TaskAction Reject = NSApiClient.getTransaction().rejectTask(<taskId>;</pre>
	<p>Complete Tasks</p> <p>Perform an HTTP POST with the action and task Id in the REST URL.</p> <p>POST REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/<taskId>/done</code></p> <p>Java Example:</p> <pre>TaskAction Complete = NSApiClient.getTransaction().completeTask(<taskId>;</pre>
	<p>Review Tasks</p> <p>Perform an HTTP POST with the action and task Id in the REST URL.</p> <p>POST REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/tasks/<taskId>/review</code></p> <p>Java Example:</p> <pre>TaskAction Review = NSApiClient.getTransaction().reviewTask(<taskId>;</pre>

Area	Examples
Filters	<p>View Filters</p> <p>The views available for filtering correspond to those in the Service Manager module. User-defined views may not be used.</p> <p>If the ViewName filter is not specified, it is set to AvailableWork.</p> <p>Possible values for ViewName: AvailableWork, MyWork, MyLateWork, WorkForeCast</p> <p>If an incorrect value is given the default value is used.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/transaction/tasks?viewName=<viewName></code></p> <p>Java Example:</p> <pre>MultiValueMap paramsMap = new MultiValueMap(); paramsMap.put("ViewName", "<viewName>"); TaskList tasks = NSApiClient.getTransaction().getDeliveryTasks(paramsMap);</pre>
Sort Column(s)	<p>Activity ID, Completed On, Customer Name, Customer OU Name, Due On, Effort, Initiator Name, Performer Name, Priority, Requisition ID, Scheduled Start Date, Service Name, Task Name, Task Type</p>
Response XML	<pre><tasks totalCount="x" recordSize="y" startRow="z"> <task> . . </task> </tasks></pre>

Lifecycle Center Data

Service Item Details

Area	Examples
CoreAPI	<p>Get by Name</p> <p>Returns the details and subscription data of service item instances assigned to the current user for the service item name specified.</p> <p>If the user also has the capability “View Service Items for My Business Units”, the services items for people in all the OUs to which the user belongs are returned.</p> <p>The service item name parameter accepts the internal table name of the service item as shown in the Name field on the Design Service Item page (for example, SiVirtualMachine).</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceName></code></p> <p>Java Example:</p> <pre>ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemData("<serviceName>", null);</pre> <hr/> <p>View Filters</p> <p>The views available for filtering correspond to what the user sees in My Services and Service Item Manager modules.</p> <p>Possible values for ViewName:</p> <ul style="list-style-type: none"> • My ServiceItems – All instances of the service item the current user owns. Users who have the “View Service Items for My Business Units” capability also may view the items owned by other people in the OUs they belong to. • Manage ServiceItems – All instances of the service item (the view returns the data only if the user has the “Manage Service Item Instances” capability). <p>The ViewName argument should be placed at the end if other arguments such as sort order or page size are also specified.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceName>?ViewName=<viewName></code></p> <p>Java Example:</p> <pre>ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceName> ", "<viewName>");</pre>

Area	Examples
Filters	<p>Service item attribute filters</p> <p>Supports up to 3 filters.</p> <p>Supports all service item and subscription columns.</p> <p>Does not support filter by Service Item Classification Name (group) or Service Item Type (Cisco reserved or user-defined).</p> <p>REST URL:</p> <p>Comparison Operators:</p> <p>Number/Date columns: =, >, <, >=, <=</p> <p>String columns: = (case-sensitive; like, contains and starts-with operators are explained below)</p> <p>Relational Operators: AND, OR (case-insensitive, order precedence is not supported)</p> <p>Separator = </p> <p>Allowed Columns: All columns in service item and subscription</p> <p>http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceName>/<columnName1><operator1><value1>[<AND OR> <columnName2><operator2><value2>][<AND OR> <columnName3><operator3><value3>]</p> <p>Java Example:</p> <pre>String filter = "<columnName1><operator1><value1> <and or> <columnName2><operator2><value2> <and or> <columnName3><operator3><value3>"; ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceName>", filter);</pre> <hr/> <p>String column filters</p> <p>Filter for % (starts with) operator.</p> <p>Filter for * (Contains) operator.</p> <p>Does not support (ends with) operator.</p> <p>Examples:</p> <p>Name=service*</p> <p>Name=*g*</p> <p>Name=*g – not allowed</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceName>/<columnName>=<wildcardValue></p> <p>Java Example:</p> <pre>ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceName>",</pre>

Area	Examples
	<p>Date column attributes</p> <p>Date field values should be in mm-dd-yyyy format.</p> <p>Comparison Operators: =, >, <, <=, >=</p> <p>Example:</p> <p>SubmittedDate=12-10-2010</p> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/serviceitem/<serviceName>/<columnName><operator><mm-dd-yyyy></p> <p>Java Example:</p> <pre>ServiceItemDTO serviceItems = NSApiClient.getServiceItem().getServiceItemDataWithFilters("<serviceName> ", "<columnName><operator><mm-dd-yyyy>");</pre>
Sort Column(s)	<p>Service item attributes: All table columns.</p> <p>Subscription: Assigned Date, Display Name, ID (internal ID), Requisition ID, Submit Date.</p>
Response XML	<pre><serviceitem totalCount="x" recordSize="3" startRow="1" id="62"> <logicName></logicName> <name></name> <subscription> . . . <assignedDate> </assignedDate> <assignedDateRaw></assignedDateRaw> <customerID></customerID> <customerName> </customerName> <displayName> </displayName> <id></id> <organizationalUnitID></organizationalUnitID> <organizationalUnitName> </organizationalUnitName> <requisitionEntryID></requisitionEntryID> <requisitionID></requisitionID> <serviceItemClassificationID></serviceItemClassificationID> <serviceItemTypeID></serviceItemTypeID> <serviceItemTypeName> </serviceItemTypeName> <submittedDate> </submittedDate> <submittedDateRaw></submittedDateRaw> . . . </subscription> </serviceitem></pre>

All Service Items

Area	Examples
CoreAPI	<p>Get all items</p> <p>Shows all Subscription and Service Item data.</p> <p>By default, filters only the Service Items assigned to the current user. If the user also has the “View Service Items for My Business Units” capability, the service items for other people in all the OUs the user belongs to are also returned.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription</code></p> <p>Java Example:</p> <pre>ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionData (null) ;</pre>
Filters	<p>View Filters</p> <p>The views available for filtering correspond to what user sees in My Services and Service Item Manager modules respectively.</p> <p>Possible values for ViewName:</p> <ul style="list-style-type: none"> • My ServiceItems – All service items the current user owns. Users who have the “View Service Items for My Business Units” get also the items owned by other people in the OUs they belong to. • Manage ServiceItems – All service items (the view returns the data only if the user has the “Manage Service Item Instances” capability). <p>The ViewName argument should be placed at the end if other arguments such as sort order or page size are also specified.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription?ViewName=<viewName></code></p> <p>Java Example:</p> <pre>ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData ("<viewName>");</pre>

Area	Examples
	<p>Subscription filters</p> <p>Filters for all Service Item Subscription table columns.</p> <p>Supports up to 3 filters.</p> <p>REST URL:</p> <p>Comparison Operators:</p> <ul style="list-style-type: none"> • Number/Date columns: =, >, <, >=, <= • String columns: = (case-sensitive; like, contains and starts-with operators are explained below) <p>Relational Operators: AND, OR (case-insensitive, order precedence is not supported)</p> <p>Filter Separator: </p> <p>Supported Columns: All columns</p> <p>http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription/<columnName1><operator1><value1>[<AND OR> <columnName2><operator2><value2>][<AND OR> <columnName3><operator3><value3>]</p> <p>Java Example:</p> <pre>String filter = " <columnName1><operator1><value1> <and or> <columnName2><operator2><value2> <and or> <columnName3><operator3><value3>"; ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData (filter);</pre>
	<p>String column filters</p> <p>Support % (starts with) operator.</p> <p>Support * (Contains) operator.</p> <p>Does not support (ends with) operator.</p> <p>Examples:</p> <pre>Name=service* Name=*g* Name=*g -- not allowed</pre> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription/<columnName>=<wildcardValue></p> <p>Java Example:</p> <pre>String filter = "<columnName>=<wildcardValue>"; ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData (filter);</pre>

Area	Examples
	<p>Date column filters</p> <p>Date field values should be in mm-dd-yyyy format.</p> <p>Comparison Operators: =, >, <, <=, >=</p> <p>Example:</p> <pre>SubmittedDate=12-10-2010</pre> <p>REST URL:</p> <pre>http://<ServerURL>/RequestCenter/nsapi/serviceitems/serviceitemsubscription /<columnName><operator><mm-dd-yyyy></pre> <p>Java Example:</p> <pre>String filter = "<columnName><operator><mm-dd-yyyy>"; ServiceItemSubscriptionList AllserviceItems = NSApiClient.getServiceItemSubscription().getServiceItemSubscriptionFilterData (filter);</pre>
Sort Column(s)	Assigned Date, Customer ID, Display Name, Organizational Unit ID, Requisition ID, Requisition Entry ID, Service Item Classification ID, Service Item ID, Service Item Type ID, Service Item Type Name, Submitted Date
Response XML	<pre><AllServiceItems totalCount="x" recordSize="y" startRow="z"> <serviceitemsubscription displayName=" " id=" "> . . . <serviceItemTypeName> </serviceItemTypeName> <organizationalUnitID></organizationalUnitID> <assignedDate> </assignedDate> <requisitionID></requisitionID> <submittedDate> </submittedDate> <submittedDateRaw></submittedDateRaw> <assignedDateRaw></assignedDateRaw> <customerID></customerID> <requisitionEntryID></requisitionEntryID> <serviceItemID></serviceItemID> <serviceItemTypeID></serviceItemTypeID> <organizationalUnitName> </organizationalUnitName> <customerName> </customerName> <serviceitem id=" "> <logicName> </logicName> <name> </name> <serviceItemData rowId=" "> <serviceItemAttribute name=" "> </serviceItemAttribute> . . . </serviceitemsubscription> </AllServiceItems></pre>

Standards

Area	Examples
CoreAPI	<p>Get by Name</p> <p>By default returns the first 50 entries for the specified Standard.</p> <p>REST URL: <a href="http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>">http://<ServerURL>/RequestCenter/nsapi/standard/<standardName></p> <p>Java Example:</p> <pre>StandardDTO standards = NSApiClient.getStandard().getStandardData("<standardName>", null);</pre>
Filters	<p>Supports up to three filters.</p> <p>REST URL:</p> <p>Comparison Operators:</p> <ul style="list-style-type: none"> • Number/Date columns: =, >, <, >=, <= (case-sensitive) • String columns: = (like, contains and starts-with operators are explained below) <p>Relational Operators: AND, OR (case-insensitive, order precedence is not supported)</p> <p>Filter Separator: </p> <p>Supported Columns: All columns</p> <p><a href="http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName1><operator1><value1>[<AND OR> <columnName2><operator2><value2>][<AND OR> <columnName3><operator3><value3>]">http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName1><operator1><value1>[<AND OR> <columnName2><operator2><value2>][<AND OR> <columnName3><operator3><value3>]</p> <p>Java Example:</p> <pre>String filter: "<columnName1><operator1><value1> <and or> <columnName2><operator2><value2> <and or> <columnName3><operator3><value3>"; StandardDTO standards = NSApiClient.getStandard().getStandardDataWithFilters("<standardName>", filter);</pre>

Area	Examples
	<p>String column filters</p> <p>Support % (starts with) operator.</p> <p>Support * (Contains) operator.</p> <p>Does not support (ends with) operator.</p> <p>Examples:</p> <pre>Name=service* Name=*g* Name=*g -- not allowed</pre> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName>=<wildcardName></p> <p>Java Example:</p> <pre>StandardDTO standards = NSApiClient.getStandard().getStandardDataWithFilters("<standardName>", "<wildcardName>");</pre>
	<p>Date column filters</p> <p>Date field values should be in mm-dd-yyyy format.</p> <p>Comparison Operators: =, >, <, <=, >=</p> <p>Example:</p> <pre>SubmittedDate=12-10-2010</pre> <p>REST URL:</p> <p>http://<ServerURL>/RequestCenter/nsapi/standard/<standardName>/<columnName><operator><mm-dd-yyyy></p> <p>Java Example:</p> <pre>StandardDTO standards = NSApiClient.getStandard().getStandardDataWithFilters("<standardName>", "<columnName><operator><mm-dd-yyyy>");</pre>
Sort Column(s)	All table columns.
Response XML	<pre><standard totalCount="x" startRow="y" recordSize="z" id="a"> <loginName></loginName> <name></name> <standardData rowId=""> . . . <standardAttribute name="id" /> . . . <standardURL><a ></standardURL> <standardURLOnly> </standardURLOnly> . . . </standardData> </standard></pre>

Service Portal Data

Custom Content

Area	Examples
CoreAPI	<p>Get by Name</p> <p>By default returns the first 50 entries in the specified table.</p> <p>REST URL:</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName></code></p> <p>Java Example:</p> <pre>CustomContentDTO customs = NSApiClient.getCustomContent().getcontentData("<customContentName>", null);</pre>
Filters	<p>Support all custom content columns.</p> <p>Support % (starts with) operator.</p> <p>Do not support (ends with) operator.</p> <p>Examples:</p> <pre>Name=custom* Name=*g -- not allowed</pre> <p>REST URL:</p> <p>Comparison Operators: =, >, <, >=, <= (case-sensitive)</p> <p>Relational Operators: AND, OR (case-insensitive)</p> <p>Filter Separator: </p> <p>Supported Columns: All columns</p> <p><code>http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName>/<columnName1><operator1><value1>[<AND OR> <columnName2><operator2><value2>][<AND OR> <columnName3><operator3><value3>]</code></p> <p><code>http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName>/<columnName>=<wildcardName></code></p> <p><code>http://<ServerURL>/RequestCenter/nsapi/customcontent/<customContentName>/<columnName><operator><mm-dd-yyyy></code></p> <p>Java Example:</p> <pre>String filter = "<columnName><operator><value>"; CustomContentDTO customs = NSApiClient.getCustomContent().getcustomContentDataWithFilters("CoPortalContent", filter);</pre>

Area	Examples
Sort Column(s)	All table columns
Response XML	<pre><customContent totalCount="x" startRow="y" recordSize="z" id="a"> <logicName></logicName> <name></name> . . . <customContentData rowId=""> <customContentAttribute name=" "></customContentAttribute> . . . </customContentData> </customContent></pre>

The complete list of column names and descriptions for the entities can be found in the “Reference Data” section in the Portal Manager chapter of the *Cisco Service Portal Designer Guide*.

Error Messages

Different HTTP response codes are returned by nsAPI depending on the nature of the exceptions:

- HTTP Status code **401** (Unauthorized) and XML error response message “**User does not have proper authentication.**” – Invalid or no authentication parameters.
- HTTP Status code **404** (Not Found) and XML error response message “**Requested resource could not be found.**” – Data could not be fetched for the specified parameter/URL values.

EXAMPLES:

```
nsapi/directory/people/id/-1
nsapi/directory/people/id/foo
nsapi/directory/people/id/1000 (there is no person with id = 1000)
nsapi/directory/people/name/<non-existent person>
nsapi/directory/people/idxyz/1
```

- HTTP Status code **403** (Forbidden) and XML error response message “**The user does not have sufficient permissions to perform the operation this object.**” – Data could not be fetched because the user does not have sufficient permissions to perform the specified operation on the object.
- HTTP Status **500** (Internal Error) and XML error response message “**Internal Error: Invalid parameter values specified or unexpected error.**” – Incorrect parameters, any other exceptions that occur within nsAPI or any other general server error.

EXAMPLES:

```
nsapi/directory/people?startRow=5000 (non-existent 5000 row)
nsapi/directory/people?sortBy=wrongColumn&sortDir=Asc (unsupported column)
nsapi/directory/people?recordSize=-1 (negative or zero value for recordSize)
```

- Http Status **422** (Unprocessable Entity) and XML validation error response message for post/update operations – The request data does not have values for mandatory fields or contains invalid values for the fields.

nsAPI does not return any error if no result is found for the filters provided; for example,

```
nsapi/directory/people?name=<non-existent person>
```

nsAPI throws an NSAPIException from Java for all exceptions encountered when executing methods in nsAPI.

Quick Reference

The chart below provides a summary of operations supported for the different entity types.

Entity Name	Get by Id	Get by Name	Get all	Wildcard Name Search	Sorting	Paging	Convert REST Filters to Java Client methods	Update	Filters	Nested Entities
Authentication										
Authentication - Login/Logout					X		X			
Definitional Data										
Categories	X	X	X	X	X	X	X		Catalog Type	Subcategories, Included Services, Included Offerings
Services	X	X	X	X	X	X	X		Categoryid, Category Name, Keyword	Categories, Keywords, Included Services
Service Offering	X	X	X	X	X	X	X		Name	Categories, Keywords, Objectives, Cost Drivers, Component Services
Agents	X	X	X	X	X	X	X		Name	
Transactional Data										
Agreement	X	X	X	X	X	X	X		Name	
Requisition	X		X		X	X	X		View Name, Status	
Requisition Entry	X		X		X	X				
Authorization	X		X		X	X	X		View Name, Status	
Task	X		X		X	X	X		View Name	
Task – Tasks for a specific requisition entry			X		X	X	X		Task Type, Status (Skipped)	Included Service Tasks
Task – Milestones for a specific requisition			X							
Task – Actions								X		

Entity Name	Get by Id	Get by Name	Get all	Wildcard Name Search	Sorting	Paging	Convert REST Filters to Java Client methods	Update	Filters	Nested Entities
Directory Data										
Organizational Units	X	X	X	X	X	X	X		OU Type	
Person	X	X	X	X	X	X	X	X	OU Name, Group Name, Role Name	OU, Group, Roles, Address, Contact, Preferences, Delegates
Groups	X	X	X	X	X	X	X		Name	
Accounts	X	X	X	X	X	X	X		Name	
Lifecycle Center Data										
All Service Items			X		X	X			Any column	
Service Items			X		X	X	X		Any column	
Standards			X		X	X	X		Any column	
Service Portal Data										
Custom Content		X	X	X	X	X	X		Any Column	



CHAPTER 7

JSR Portlets

- [Overview, page 7-1](#)
- [Portlet Structure and Packaging, page 7-1](#)
- [Portlet Development, page 7-6](#)
- [Compiling JSR Portlet Controller, page 7-27](#)
- [Portlet Deployment, page 7-28](#)

Overview

The Portal Manager solution within Service Portal provides a rich platform for integrating with external applications through JSR Portlets. The portal front-end uses Apache Pluto 1.1 libraries as the framework. Portlets developed using APIs which meet the Java Portlet Specification (JSR168, JSR286) standards may be deployed along with Service Portal. Once deployed, these will appear in Portal Designer as “Third-Party Portlets” and can be added to portal pages. For more information on how to maintain JSR portlets and other content in the Portal Manager solution, see the *Cisco Service Portal Designer Guide*.

This chapter covers some guidelines on the development and deployment of JSR portlets for the Portal Manager solution. A sample portlet named “MyJSR” is used throughout the chapter as an illustration. The portlet is developed with Spring 3.0 Annotation-based Controller and Sencha’s Ext JS—the JavaScript framework for the portal front-end.

Portlet Structure and Packaging

The portlet files should be packaged according to the JSR 168 or 286 specifications, in the form of web application (war) files appropriate for the application server used. A typical portlet war file may include servlets, resource bundles, images, html, jsp, css files, and so on.

JBoss Application Server

Here is the anatomy of a simple portlet named “MyJSR.war”:

1. `css`
 MyJSR.css
2. `images`
 <Custom Images that the Portlet needs can be placed here>
3. `js`

```

MyJSRCreatePersonView.js
MyJSREdit.js
MyJSRHelp.js
MyJSRView.js
4. WEB-INF
  classes
    com
      myjsr
        MyJSRController.class
      config
        spring
          MyJSRApplicationContext.xml
        jsrportlet.properties
        log4j.properties
  jsp
    MyJSREdit.jsp
    MyJSRHelp.jsp
    MyJSRView_listperson.jsp
    MyJSRView_updateperson.jsp
  lib
    newscale_appclient.jar
    newscale_core.jar
    cxf-2.2.7.jar
    pluto-portal-driver-2.0.2.jar
    org.springframework.aop-3.1.0.RELEASE.jar
    org.springframework.asm-3.1.0.RELEASE.jar
    org.springframework.aspects-3.1.0.RELEASE.jar
    org.springframework.beans-3.1.0.RELEASE.jar
    org.springframework.context-3.1.0.RELEASE.jar
    org.springframework.context.support-3.1.0.RELEASE.jar
    org.springframework.core-3.1.0.RELEASE.jar
    org.springframework.expression-3.1.0.RELEASE.jar
    org.springframework.instrument-3.1.0.RELEASE.jar
    org.springframework.instrument.tomcat-3.1.0.RELEASE.jar
    org.springframework.jdbc-3.1.0.RELEASE.jar
    org.springframework.jms-3.1.0.RELEASE.jar
    org.springframework.orm-3.1.0.RELEASE.jar
    org.springframework.oxm-3.1.0.RELEASE.jar
    org.springframework.test-3.1.0.RELEASE.jar
    org.springframework.transaction-3.1.0.RELEASE.jar
    org.springframework.web-3.1.0.RELEASE.jar
    org.springframework.web.portlet-3.1.0.RELEASE.jar
    org.springframework.web.servlet-3.1.0.RELEASE.jar
    org.springframework.web.struts-3.1.0.RELEASE.jar
  tld
    c.tld
    pluto.tld
    portlet.tld
    portlet_2_0.tld
    portlet-el.tld
    portlet-el_2_0.tld
  portlet.xml
  web.xml
  jboss-deployment-structure.xml

```

In this sample portlet, the nsAPI java client—`newscale_appclient.jar`—is included in the `lib` folder as the portlet invokes the REST API to retrieve data from Service Portal. Pluto libraries and other libraries that the nsAPI java client depends on need to be included in the `lib` folder. In addition, the `jboss-deployment-structure.xml` is included to describe the dependencies on the JBoss modules.

An additional descriptor—`portlet.xml`—must be present to specify the portlet-related configurations.

Weblogic Application Server

On WebLogic, the JSR portlet war structure is similar except that it must contain all the libraries the portlet uses because it is deployed outside of the Service Portal application.

Some of the libraries included below are required for invoking the nsAPI java client and to read portlet common settings. Certain libraries are needed here because the portlet uses JSTL tags in JSPs and Spring portlets MVC.

1. css
 - MyJSR.css
2. images
 - <Custom Images that the Portlet needs can be placed here>
3. js
 - MyJSRCreatePersonView.js
 - MyJSREdit.js
 - MyJSRHelp.js
 - MyJSRView.js
4. WEB-INF
 - classes
 - com
 - myjsr
 - MyJSRController.class
 - config
 - spring
 - MyJSRApplicationContext.xml
 - jsrportlet.properties
 - log4j.properties
 - jsp
 - MyJSREdit.jsp
 - MyJSRHelp.jsp
 - MyJSRView_listperson.jsp
 - MyJSRView_updateperson.jsp
 - lib
 - newscale_appclient.jar
 - newscale_core.jar
 - newscale_udkernel.jar
 - newscale_compbeans.jar
 - newscale_conf.jar
 - castor-0.9.5.4.jar
 - commons-beanutils-1.8.3.jar
 - commons-httpclient-3.1.jar
 - commons-logging-1.0.4.jar
 - cxf-2.2.7.jar
 - ezmorph-1.0.4.jar
 - json-lib-2.2.2-jdk13.jar
 - jsr311-api-1.0.jar
 - neethi-2.0.4.jar
 - oscache-2.4.jar
 - standard.jar
 - wsdl4j-1.6.1.jar
 - XmlSchema-1.4.6.jar
 - commons-collections-3.2.1.jar
 - commons-lang-2.4.jar
 - neethi-2.0.4.jar
 - castor-0.9.5.4.jar
 - jstl.jar
 - org.springframework.aop-3.0.2.RELEASE.jar
 - org.springframework.asm-3.0.2.RELEASE.jar
 - org.springframework.aspects-3.0.2.RELEASE.jar
 - org.springframework.beans-3.0.2.RELEASE.jar
 - org.springframework.context-3.0.2.RELEASE.jar
 - org.springframework.context.support-3.0.2.RELEASE.jar

```

org.springframework.core-3.0.2.RELEASE.jar
org.springframework.expression-3.0.2.RELEASE.jar
org.springframework.instrument-3.0.2.RELEASE.jar
org.springframework.instrument.tomcat-3.0.2.RELEASE.jar
org.springframework.jdbc-3.0.2.RELEASE.jar
org.springframework.jms-3.0.2.RELEASE.jar
org.springframework.spring-library-3.0.2.RELEASE.libd
org.springframework.test-3.0.2.RELEASE.jar
org.springframework.transaction-3.0.2.RELEASE.jar
org.springframework.web-3.0.2.RELEASE.jar
org.springframework.web.portlet-3.0.2.RELEASE.jar
org.springframework.web.servlet-3.0.2.RELEASE.jar
org.springframework.web.struts-3.0.2.RELEASE.jar
tld
  c.tld
  pluto.tld
  portlet.tld
  portlet_2_0.tld
  portlet-el.tld
  portlet-el_2_0.tld
portlet.xml
web.xml

```

WebSphere Application Server

The portlet war for WebSphere is also deployed as a web application outside of the Service Portal application.

Here notice that some different library versions are used to work with WebSphere.

1. css
 - MyJSR.css
2. images
 - <Custom Images that the Portlet needs can be placed here>
3. js
 - MyJSRCreatePersonView.js
 - MyJSREdit.js
 - MyJSRHelp.js
 - MyJSRView.js
4. WEB-INF
 - classes
 - com
 - myjsr
 - MyJSRController.class
 - config
 - spring
 - MyJSRApplicationContext.xml
 - jsrportlet.properties
 - log4j.properties
 - jsp
 - MyJSREdit.jsp
 - MyJSRHelp.jsp
 - MyJSRView_listperson.jsp
 - MyJSRView_updateperson.jsp
 - lib
 - newscale_appclient.jar
 - castor-0.9.5.4.jar
 - commons-beanutils-1.8.3.jar
 - commons-collections-3.2.1.jar
 - commons-httpclient-3.1.jar
 - commons-lang-2.4.jar
 - commons-logging-1.0.4.jar

```

cxf-2.2.12.jar
ezmorph-1.0.4.jar
json-lib-2.2.2-jdk13.jar
jsr311-api-1.0.jar
neethi-2.0.4.jar
newscale_compbeans.jar
newscale_conf.jar
newscale_core.jar
newscale_udkernel.jar
oscache-2.4.jar
standard.jar
wsdl4j-1.6.1.jar
XmlSchema-1.4.6.jar
jstl.jar
org.springframework.aop-3.0.2.RELEASE.jar
org.springframework.asm-3.0.2.RELEASE.jar
org.springframework.aspects-3.0.2.RELEASE.jar
org.springframework.beans-3.0.2.RELEASE.jar
org.springframework.context-3.0.2.RELEASE.jar
org.springframework.context.support-3.0.2.RELEASE.jar
org.springframework.core-3.0.2.RELEASE.jar
org.springframework.expression-3.0.2.RELEASE.jar
org.springframework.instrument-3.0.2.RELEASE.jar
org.springframework.instrument.tomcat-3.0.2.RELEASE.jar
org.springframework.jdbc-3.0.2.RELEASE.jar
org.springframework.jms-3.0.2.RELEASE.jar
org.springframework.spring-library-3.0.2.RELEASE.libd
org.springframework.test-3.0.2.RELEASE.jar
org.springframework.transaction-3.0.2.RELEASE.jar
org.springframework.web-3.0.2.RELEASE.jar
org.springframework.web.portlet-3.0.2.RELEASE.jar
org.springframework.web.servlet-3.0.2.RELEASE.jar
org.springframework.web.struts-3.0.2.RELEASE.jar
tld
  c.tld
  pluto.tld
  portlet.tld
  portlet_2_0.tld
  portlet-el.tld
  portlet-el_2_0.tld
portlet.xml
web.xml

```

Dependent Libraries

The set of libraries required for inclusion in the JSR Portlet war file are available in either the deployed RequestCenter application on the application server or the Service Portal installer image, as described in the table below:

File Name	JBoss	WebLogic	WebSphere
newscale_applient.jar	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
newscale_compbeans.jar	—	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
newscale_conf.jar	—	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib

File Name	JBoss	WebLogic	WebSphere
newscale_core.jar	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
commons-beanutils-1.8.3.jar	—	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
commons-collections-3.2.1.jar	—	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
commons-lang-2.4.jar	—	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
cxf-2.2.7.jar	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib	—
cxf-2.2.12.jar	—	—	preinstall/websphere/jsrportlet (located in the product image)
json-lib-2.2.2-jdk13.jar	—	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
org.springframework*.jar	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
pluto-container-2.0.2.jar	—	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
pluto-portal-driver-2.0.2.jar	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
pluto-portal-driver-impl-2.0.2.jar	—	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib
pluto-taglib-2.0.2.jar	—	preinstall/weblogic/cisco_lib (located in the product image)	preinstall/websphere/lib_ext (located in the product image)
portlet-api_2.0_spec-1.0.jar	—	RequestCenter.war/WEB-INF/lib	RequestCenter.war/WEB-INF/lib

Portlet Development

A typical JSR portlet should cover the three rendering modes —View, Edit, and Help. In addition, the portlet would support different window states—Normal, Minimized, and Maximized.

The MyJSR portlet example shown below provides a user interface that supports two functionalities:

1. List Service Portal users in a grid.

MyJSRPortlet

Logged In User (admin) : admin , admin
HomeOUId : 1 , PersonId : 1

+ Create Person

First Name	Last Name	Email	Home OU	Login Name	Timezone	Language
admin	admin	requestcen...	Site Admini...	admin	America/Tj...	German
giri	giri	giri243@ya...	ou1	giri	America/Tj...	US English
hamilton	jordan	hamir@mer...	ou1	hamilt	America/Tj...	US English
jacob	martin	jacob@emc...	ou1	jacob	America/Tj...	US English
tom	hanks	tommy@hol...	ou1	tommy	America/Tj...	US English

Page 1 of 1 | Displaying 1 - 5 of 5

2. Allow user to be added/updated.

My JSR Portlet

Person Details - Add

First Name:

Last Name:

Login Name:

Email:

Home OU:

Language:

Timezone:

Address

Business:

Home:

To achieve the above requirements, the sample code that follows includes these high-level operations:

- Retrieval of Service Portal users using nsAPI java client
- Returning the user details in JSON format to the user interface
- Rendering the list of users in a Ext JS grid on the browser
- Display/entry of user details in a form designed using Ext JS
- Adding/updating user details in the Service Portal repository using nsAPI java client

MyJSR.css

#Code Custom Styles for the portlet can be designed here.

Now let us examine the content of each of the components within the MyJSR.war.

MyJSRCreatePersonView.js

Example code using Ext JS to display a form for creating a user.

```
/* Code custom JavaScript for the portlet here */

Ext.onReady(function() {
var tab2 = new Ext.FormPanel({
id : 'personEditForm',
labelAlign : 'top',
title : 'Person Details - Add',
bodyStyle : 'padding:5px',
width : 600,
renderTo : MyJSREditDiv,
items : [{
layout : 'column',
border : false,
items : [{
columnWidth : .5,
layout : 'form',
border : false,
items : [{
xtype : 'textfield',
fieldLabel : 'First Name',
value : personListObj.firstName,
name : 'firstName',
anchor : '95%'
}, {
xtype : 'textfield',
fieldLabel : 'Login Name',
value : personListObj.login,
name : 'login',
anchor : '95%'
}, {
xtype : 'textfield',
fieldLabel : 'Home OU',
name : 'homeOrganizationalUnitName',
value : personListObj.homeOrganizationalUnitName,
anchor : '95%'
}, {
xtype : 'textfield',
fieldLabel : 'Timezone',
name : 'timeZoneName',
value : personListObj.timeZoneName,
anchor : '95%'
}
]
}, {
columnWidth : .5,
layout : 'form',
border : false,
items : [{
xtype : 'textfield',
fieldLabel : 'Last Name',
name : 'lastName',
value : personListObj.lastName,
```



```

        anchor : '95%'
                }, {
        xtype : 'textfield',
        fieldLabel : 'Email',
        name : 'email',
                value : personListObj.email,
        vtype : 'email',
        anchor : '95%'
                }, {
        xtype : 'textfield',
        fieldLabel : 'Language',
        name : 'languageName',
                value : personListObj.languageName,
        anchor : '95%'
                }]
        ]}
    }, {
    xtype : 'tabpanel',
    plain : true,
    activeTab : 0,
    height : 235,
    defaults : {
        bodyStyle : 'padding:10px'
    },
    items : [{
        title : 'Address',
        layout : 'form',
        defaults : {
            width : 230
        },
        defaultType : 'textfield',
        items : [{
            fieldLabel : 'Business',
            name : 'businessAddress',
            disabled : true
        }, {
            fieldLabel : 'Home',
            name : 'homeAddress',
            disabled : true
        }
        ], {
        title : 'Contacts',
        layout : 'form',
        defaults : {
            width : 230
        },
        defaultType : 'textfield',
        items : [{
            fieldLabel : 'Business',
            name : 'businessPhone',
            disabled : true
        }, {
            fieldLabel : 'Home',
            name : 'homePhone',
            disabled : true
        }, {
            fieldLabel : 'Mobile',
            name : 'mobilePhone',
            disabled : true
        }, {
            fieldLabel : 'Fax',
            name : 'faxNumber',
            disabled : true
        }
        ]
    }
}

```

```

        }
    }
    }],
    buttons : [{
        text : 'Save',
        handler : function() {
            Ext.getCmp("personEditForm").getForm().submit({
                url : addPersonActionUrl,
                params : {},
                success : function(form, action) {
                    var responseObj = Ext.util.JSON.decode(action.response.responseText);
                    if(responseObj.success == "true")
                        Ext.Msg.alert('Success', responseObj.successMsg);
                    else
                        Ext.Msg.alert('Error', responseObj.errorMsg);
                }
            });
        }
    }, {
        text : 'Cancel',
        handler : function() {
            window.location=viewPersonUrl;
        }
    }
    ]
    });
});

```

MyJSREdit.js

This JavaScript can be used to add any custom code for portlet edit mode.

```

/* Code custom JavaScript for the Portlet here */

Ext.onReady(function() {

});

```

MyJSRHelp.js

This JavaScript can be used to add any custom code for portlet edit mode.

```

/* Code custom JavaScript for the Portlet here */

Ext.onReady(function() {

});

```

MyJSRView.js

Example JavaScript to display users in Ext JS grid.

```

/* Code custom JavaScript for the Portlet here */
Ext.onReady(function() {
    // Demonstrates how to getUser info from Java Script and set it to div
    varLogin=document.getElementById('MyJSRLoginNameDiv');

```

```

Login.innerHTML=nsAPP_CurrentUserLoginName;
varFirstName=document.getElementById('MyJSRFirstNameDiv');
    FirstName.innerHTML=nsAPP_CurrentUserFirstName;
varLastName=document.getElementById('MyJSRLastNameDiv');
    LastName.innerHTML=nsAPP_CurrentUserLastName;
varHomeOU=document.getElementById('MyJSRHomeOUDiv');
    HomeOU.innerHTML=nsAPP_CurrentUserHomeOuId;
var PersonID=document.getElementById('MyJSRPersonIDDiv');
    PersonID.innerHTML=nsAPP_CurrentUserId;

var pid = portletId.substr(pidPrefix.length);
if (Ext.getCmp(pid).height && Ext.getCmp(pid).height >= 29) {
    var gridHeight = Ext.getCmp(pid).height - 29;
    }

var gridStore = new Ext.data.JsonStore({
    proxy : new Ext.data.HttpProxy({
        url : pagingUrl,
        timeout : connectionTimeout
    }),
    autoLoad: {params:{start: 0, limit: defaultRecordSize}},
    root: 'rows',
    totalProperty: 'results',
    fields : [{
        name : 'firstName',
        type : 'string'
    }, {
        name : 'lastName',
        type : 'string'
    }, {
        name : 'email',
        type : 'string'
    }, {
        name : 'homeOrganizationalUnitName',
        type : 'string'
    }, {
        name : 'login',
        type : 'string'
    }, {
        name : 'timeZoneName',
        type : 'string'
    }, {
        name : 'languageName',
        type : 'string'
    }, {
        name : 'businessPhone',
        type : 'string'
    }, {
        name : 'homePhone',
        type : 'string'
    }, {
        name : 'mobilePhone',
        type : 'string'
    }, {
        name : 'faxNumber',
        type : 'string'
    }, {
        name : 'businessAddress',
        type : 'string'
    }, {
        name : 'homeAddress',
        type : 'string'
    }
    ]
});

```

```

gridStore.load();

var expander = new Ext.ux.grid.RowExpander({
    tpl : new Ext.Template(
        '<h2 class="title">Address</h2><table>',
        '<tr><td width=400><b>Business</b> {businessAddress}</td>',
        '<td width=400><b>Home</b> {homeAddress}</td></tr></table>',
        '<h2 class="title">Contact</h2><table>',
        '<tr><td width=400><b>Business</b> {businessPhone}</td>',
        '<td width=400><b>Home</b> {homePhone}</td></tr>',
        '<tr><td width=400><b>Mobile</b> {mobilePhone}</td>',
        '<td width=400><b>Fax</b> {faxNumber}</td></tr></table>'
    )
});

var gridColModel = new Ext.grid.ColumnModel({
    defaults : {
        sortable : true,
        autoWidth : true
    },
    columns : [{
        header : "First Name",
        dataIndex : 'firstName'
    }, {
        header : "Last Name",
        dataIndex : 'lastName'
    }, {
        header : "Email",
        dataIndex : 'email'
    }, {
        header : "Home OU",
        dataIndex : 'homeOrganizationalUnitName'
    }, {
        header : "Login Name",
        dataIndex : 'login'
    }, {
        header : "Timezone",
        dataIndex : 'timeZoneName'
    }, {
        header : "Language",
        dataIndex : 'languageName'
    }
    ]
});

var gridConfig = {
    renderTo : MyJSREditDiv,
    width : "100%",
    layout : 'fit',
    store : gridStore,
    cm : gridColModel,
    loadMask: true,
    autoWidth : true,
    plugins : expander,
    tbar : [{
        text : 'Create Person',
        iconCls : 'add',
        handler : function() {
            window.location=createNewPersonActionUrl;
        }
    }, '-'],
    bbar : new Ext.PagingToolbar({
        pageSize : defaultRecordSize,
        store : gridStore,
        displayInfo : true,
        params:{

```

```

start: 0,
limit: defaultRecordSize
    }
    })
};

if ('maximized' == portletWindowState) {
gridConfig.height = document.documentElement.clientHeight - 188;
} elseif ('normal' == portletWindowState) {
var viewConfig = {
forceFit : true
};
gridConfig.viewConfig = viewConfig;

if (gridHeight && gridHeight > -1) {
gridConfig.height = gridHeight;
} else {
gridConfig.autoHeight = true;
}
}

var grid = new Ext.grid.GridPanel(gridConfig);
});

```

portlet.xml

Example of the portlet specification. The portlet-class needs to be set along with a pair of init-params—contextConfigLocation and nsContentPortlet (nsContentPortlet is always set to “false”).

```

<?xmlversion="1.0"encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.

See the License for the specific language governing permissions and
limitations under the License.
-->
<portlet-app
xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd"
version="1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd
http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
<portlet>
<description>MyJSR Description</description>
<portlet-name>nsMyJSR</portlet-name>
<display-name>My JSR Portlet</display-name>
<portlet-class>org.springframework.web.portlet.DispatcherPortlet</portlet-class>
<init-param>
<name>contextConfigLocation</name>

```

```

<value>/WEB-INF/classes/config/spring/MyJSRApplicationContext.xml</value>
</init-param>
<init-param>
<name>nsContentPortlet</name>
<value>>false</value>
</init-param>
<expiration-cache>-1</expiration-cache>
<supports>
<mime-type>text/html</mime-type>
<portlet-mode>VIEW</portlet-mode>
<portlet-mode>EDIT</portlet-mode>
<portlet-mode>HELP</portlet-mode>
</supports>
<portlet-info>
<title>My JSR Portlet</title>
</portlet-info>
</portlet>
</portlet-app>

```

web.xml

Example deployment descriptor with the servlet and servlet mapping is required by the portal server; in this case, Apache Pluto.

```

<?xmlversion="1.0"encoding="UTF-8"?>
<!DOCTYPEweb-appPUBLIC"-Sun Microsystems, Inc. DTD Web Application
2.3EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
<display-name>My JSR Portlet Application</display-name>
<description>My JSR Portlet</description>

<!-- Resources bundle base class -->
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>
    /WEB-INF/classes/config/spring/MyJSRApplicationContext.xml
</param-value>
</context-param>

<context-param>
<param-name>parameter-name</param-name>
<param-value>parameter-value</param-value>
</context-param>

<servlet>
<servlet-name>ViewRendererServlet</servlet-name>
<servlet-class>
org.springframework.web.servlet.ViewRendererServlet
</servlet-class>
</servlet>

<servlet>
<servlet-name>MyJSR</servlet-name>
<servlet-class>org.apache.pluto.container.driver.PortletServlet</servlet-class>
<init-param>
<param-name>portlet-name</param-name>
<param-value>MyJSR</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>

```

```

<servlet-mapping>
<servlet-name>ViewRendererServlet</servlet-name>
<url-pattern>/WEB-INF/servlet/view</url-pattern>
</servlet-mapping>

<servlet-mapping>
<servlet-name>MyJSR</servlet-name>
<url-pattern>/PlutoInvoker/nsMyJSR</url-pattern>
</servlet-mapping>

<!-- Declare Tag libraries that are used in which are going to use in JSP pages-->
<taglib>
<taglib-uri>http://portals.apache.org/pluto</taglib-uri>
<taglib-location>/WEB-INF/tld/pluto.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://java.sun.com/portlet_2_0</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet_2_0.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>/WEB-INF/tld/c.tld</taglib-uri>
<taglib-location>/WEB-INF/tld/c.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://java.sun.com/portlet</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://portals.apache.org/pluto/portlet-el</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet-el.tld</taglib-location>
</taglib>

<taglib>
<taglib-uri>http://portals.apache.org/pluto/portlet-el_2_0</taglib-uri>
<taglib-location>/WEB-INF/tld/portlet-el_2_0.tld</taglib-location>
</taglib>

</web-app>

```

MyJSREdit.jsp

JSP for portlet edit mode.

```

<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http:java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="portlet2"uri="http:java.sun.com/portlet_2_0"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

<% String contextPath = request.getContextPath(); %>

<!-- This is for IE -->
<scripttype="text/javascript">

```

```

if(document.createStyleSheet) {
document.createStyleSheet('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');
}
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>');";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
}
</script>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSREdit.js") %>';
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet Edit here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
Portlet Mode = <c:outvalue='${portletMode}'/>
Portlet Window State = <c:outvalue='${portletWindowState}'/>
</c:if>

<divid="MyJSREditDiv-<portlet:namespace/>"class="x-grid-mso"></div>

<script>
var MyJSREditDiv = 'MyJSREditDiv-<portlet:namespace/>';
var addPersonActionUrl = '<portlet2:resourceURL id="addPersonData" escapeXml="false" />';
var personListObj = Ext.util.JSON.decode('<c:out value="${PersonData}"
escapeXml="false"/>');
</script>

```

MyJSRHelp.jsp

JSP for portlet help mode.

```

<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@tagliburi="http://java.sun.com/portlet"prefix="portlet"%>
<%@taglibprefix="c"uri="/WEB-INF/tld/c.tld"%>

```



```

<% String contextPath = request.getContextPath(); %>

<!-- This is for IE -->
<scripttype="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet("<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>");
}
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>')";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
}
</script>

<!-- This is foFirefox -->
<linkrel="stylesheet"type="text/css"href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSRHelp.js") %>';
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet Help here -->
<c:iftest="&${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
Portlet Mode = <c:outvalue='${portletMode}' />
Portlet Window State = <c:outvalue='${portletWindowState}' />
</c:if>

<c:iftest="&${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
Portlet Mode = <c:outvalue='${portletMode}' />
Portlet Window State = <c:outvalue='${portletWindowState}' />
</c:if>

<c:iftest="&${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
Portlet Mode = <c:outvalue='${portletMode}' />
Portlet Window State = <c:outvalue='${portletWindowState}' />
</c:if>

<divid="MyJSRHelpDiv-<portlet:namespace/>"class="x-grid-mso"></div>

<script>
var MyJSRHelpDiv = 'MyJSRHelpDiv-<portlet:namespace/>';
</script>

```

MyJSRView_listperson.jsp

JSP code for portlet view mode.

```

<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

```



```

<!--Write declare divs to display user info -->
<div>Logged In User (<div id="MyJSRLoginNameDiv"
style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span></div>)&nbsp;:&nbsp;&nbsp;<div id="MyJSRFirstNameDiv"
style="text-align:right;display:inline;width:100%;"><span></span></div>&nbsp;&nbsp;:&nbsp;&nbsp;<div id="MyJSRLastNameDiv" style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span></div>
</div>
<div>HomeOUIId&nbsp;&nbsp;:&nbsp;&nbsp;<div id="MyJSRHomeOUIDiv"
style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span></div>&nbsp;&nbsp;:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
PersonId&nbsp;&nbsp;:&nbsp;&nbsp;<div id="MyJSRPortletPersonIDDiv"
style="text-align:right;display:inline;width:100%;"><span
style="padding-left:20px;"></span>
</div>

<div id="MyJSREditDiv-<portlet:namespace/>" class="x-grid-mso"></div>

<script>
var MyJSREditDiv = 'MyJSREditDiv-<portlet:namespace/>';
var createNewPersonActionUrl = '<portlet:renderURL><portlet:param name="formAction"
value="createNewPerson" /></portlet:renderURL>';
var addPersonActionUrl = '<portlet2:resourceURL id="addPersonData" escapeXml="false" />';
var personListObj = Ext.util.JSON.decode('<c:out value="{PersonData}"
escapeXml="false"/>');
</script>

```

MyJSRView_updateperson.jsp

Example JSP code to demonstrate update user operation.

```

<%
/**
 * Copyright (c) 2012, Cisco Systems, Inc. All rights reserved.
 */
%>

<%@taglib uri="http://java.sun.com/portlet" prefix="portlet"%>
<%@taglib prefix="portlet2" uri="http://java.sun.com/portlet_2_0"%>
<%@taglib prefix="c" uri="/WEB-INF/tld/c.tld"%>

<%String contextPath = request.getContextPath(); %>

<!-- This is foFirefox -->
<link rel="stylesheet" type="text/css" href="<%= response.encodeURL(contextPath +
"/css/MyJSR.css") %>"></link>

<!-- This is for IE -->
<script type="text/javascript">
if(document.createStyleSheet) {
document.createStyleSheet("<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>");
}
else {
var styles = "@import url('<%= response.encodeURL(contextPath + "/css/MyJSR.css") %>')";
var newSS=document.createElement('link');
newSS.rel='stylesheet';
newSS.href='data:text/css,'+escape(styles);
document.getElementsByTagName("head")[0].appendChild(newSS);
}
</script>

```

```

<script>
var head = document.getElementsByTagName('head')[0];
var script = document.createElement('script');
script.type = 'text/javascript';
script.src = '<%= response.encodeURL(contextPath + "/js/MyJSRCreatePersonView.js") %>';
head.appendChild(script);
</script>

<!-- Write your JSP Code for Portlet View here -->
<c:iftest="${portletWindowState == 'NORMAL' or portletWindowState == 'normal'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MINIMIZED' or portletWindowState == 'minimized'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/> -->
</c:if>

<c:iftest="${portletWindowState == 'MAXIMIZED' or portletWindowState == 'maximized'}">
<!--Portlet Mode = <c:out value='${portletMode}'/>
Portlet Window State = <c:out value='${portletWindowState}'/>-->
</c:if>

<div id="MyJSREditDiv-<portlet:namespace/>" class="x-grid-mso"></div>
<script>
var MyJSREditDiv = 'MyJSREditDiv-<portlet:namespace/>';
var addPersonActionUrl = '<portlet2:resourceURL id="addPersonData" escapeXml="false" />';
var viewPersonUrl = '<portlet:renderURL></portlet:renderURL>';
var personListObj = Ext.util.JSON.decode('<c:out value="${PersonData}"
escapeXml="false"/>');
</script>

```

MyJSRController.java

The steps for developing java portlet controllers typically include:

-
- Step 1** Write handler code for the three portlet modes—View, Edit, and Help.
 - Step 2** Write handler code for the three portlet views—Normal, Minimized, and Maximized.
 - Step 3** For JSR portlets that process/display Service Portal entities, the nsAPI client can be used to invoke the related REST APIs in the portlet controller.
 - a. Get reference to nsAPI client API.
 - b. Call nsAPI client to get a list of the instances for the required Service Portal entity.
 - c. Optionally get the details for the currently logged-in user; for example, Person ID, First Name, Last Name.
 - Step 4** Render the instances in a grid or other format (this example also demonstrates how to do paging in nsAPI for a Ext JS Grid).
-

```

package com.myjsr;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

```

```

importjava.util.Iterator;
importjava.util.List;
import java.util.Map;
importjava.util.Properties;

importjavax.portlet.RenderRequest;
importjavax.portlet.RenderResponse;
importjavax.portlet.ResourceRequest;
importjavax.portlet.ResourceResponse;

importnet.sf.json.JSON;
importnet.sf.json.JSONSerializer;
importjavax.portlet.ActionRequest;
importjavax.servlet.http.HttpSession;
importjavax.portlet.PortletURL;
importorg.apache.commons.collections.map.MultiValueMap;
importorg.apache.commons.lang.StringEscapeUtils;
importorg.apache.commons.lang.StringUtils;
importorg.springframework.ui.Model;
importorg.springframework.web.portlet.ModelAndView;
importorg.springframework.web.bind.annotation.ModelAttribute;
importorg.springframework.web.bind.annotation.RequestMapping;
importorg.springframework.web.portlet.bind.annotation.ResourceMapping;
importorg.springframework.web.bind.annotation.RequestParam;

importcom.newscale.comps.conf.domain.AppParamUtil;
importcom.newscale.nsapi.directory.person.Person;
importcom.newscale.nsapi.directory.person.PersonList;
importcom.newscale.nsapiclient.NSApiClient;
importcom.newscale.nsapiclient.NSApiClientConstants;
importcom.newscale.nsapiclient.NSApiClientFactory;
importcom.newscale.portlets.GenericNewScaleSpringPortletBase;

/**
 *MyJSRController
 */
publicclass MyJSRController extends GenericNewScaleSpringPortletBase {
    privatestaticfinal String configPropsFile = "jsrportlet.properties";
    privatestaticfinal String viewPageList = "MyJSRView_listperson";
    privatestaticfinal String viewPageUpdate = "MyJSRView_updateperson";
    privatestaticfinal String editPage = "MyJSREdit";
    privatestaticfinal String helpPage = "MyJSRHelp";
    private NSApiClient nsApiClient = getNSApiClient();

    public NSApiClient getNsApiClient() {
        return nsApiClient;
    }

    public void setNsApiClient(NSApiClient nsApiClient) {
        this.nsApiClient = nsApiClient;
    }

    public String viewNormal(RenderRequest request, RenderResponse response, Model model) {
        try {
            super.viewNormal(request, response, model);
            getLoginUsername(request, model);
        } catch(Exception e){
            e.printStackTrace();
        }
    }

    return doView(request, response, model);
}

```

```

public String viewMinimized(RenderRequest request, RenderResponse response, Model model) {
    try {
        super.viewMinimized(request, response, model);
    } catch(Exception e){
        e.printStackTrace();
    }
    return viewPageList;
}

public String viewMaximized(RenderRequest request, RenderResponse response, Model model) {
    try {
        super.viewMaximized(request, response, model);
        getLoginUsername(request, model);
    } catch(Exception e){
        e.printStackTrace();
    }

    return doView(request, response, model);
}

private void getLoginUsername(RenderRequest request, Model model){
    Properties properties = getConfigProperties(configPropsFile);
    nsApiClient.login(properties.getProperty("BASE_URL"),
request.getPortletSession().getId());
    // Get Currently Logged-in user from nsAPI client
    Person persons = nsApiClient.getDirectory().getCurrentUser();

    // Set user info into model so that JSP can access it
    model.addAttribute("PersonID", persons.getPersonId());
    model.addAttribute("HomeOUIId", persons.getHomeOrganizationalUnitId());
    model.addAttribute("firstName", persons.getFirstName());
    model.addAttribute("lastName", persons.getLastName());
    model.addAttribute("userName", persons.getLogin());
}

private String doView(RenderRequest request, RenderResponse response, Model model) {
    try {
        Properties properties = getConfigProperties(configPropsFile);
        nsApiClient.login(properties.getProperty("BASE_URL"),
request.getPortletSession().getId());

        int defaultRecordSize = AppParamUtil.getInstance().getMaxMaxPagingSizeInNSAPI();
        model.addAttribute("defaultRecordSize", "" + defaultRecordSize);
        int connectionTimeOut = 0;
        if (AppParamUtil.getInstance().isParamExists((AppParamUtil.SESSION_TIMEOUT)) {
            connectionTimeOut =
AppParamUtil.getInstance().getIntegerParam(AppParamUtil.SESSION_TIMEOUT);
        }
        if (connectionTimeOut < 1) {
            connectionTimeOut = 20;
        }
        model.addAttribute("connectionTimeOut", "" + connectionTimeOut * 1000 * 60);

        String formAction = request.getParameter("formAction");
        String personIdStr = request.getParameter("personId");

        if (null != formAction && formAction.equals("createNewPerson") ) {
            showAddPersonPage(request, response, model);
            return viewPageUpdate;
        } else if (null != personIdStr){
            editPerson(request, response, model, new Integer(personIdStr).intValue());
            return viewPageUpdate;
        } else {
            return viewPageList;
        }
    }
}

```

```

    }
    } catch (Exception e){
        e.printStackTrace();
    }
    return null;
}

@RequestMapping("VIEW")
@ResourceMapping
public ModelAndView doPagingOrSavePerson(ResourceRequest request, ResourceResponse
response, Person person)
throws Exception {
    String instanceName= getInstanceName(request.getWindowID());
    Save Button is clicked while adding person
    if ("createNewPerson".equals(request.getParameter("formAction")) || null !=
request.getParameter("personId")) {
addPersonData(person ,request, response );
    } elseif (null != request.getParameter("start") &&null !=
request.getParameter("limit")) { Paging
int startInt = Integer.parseInt(request.getParameter("start")) + 1; extjs sends 1 less
than what nsAPI wants
int limit = Integer.parseInt(request.getParameter("limit")) + 1; extjs sends 1 less than
what nsAPI wants
try {
if (request.getWindowState().equals(request.getWindowState().NORMAL)) {
doPagingInternal(request, response, instanceName, 1, startInt, limit);
    }
if (request.getWindowState().equals(request.getWindowState().MAXIMIZED)) {
doPagingInternal(request, response, instanceName, 2, startInt, limit);
    }
    } catch (Exception e) {
e.printStackTrace();
    }
    }
return null;
}

private void doPagingInternal(ResourceRequest request, ResourceResponse response, String
portletInst,
int windowStateInt, int start, int limit) throws Exception {
    Map<String, Object> jsonMap = new HashMap<String, Object>();
    List recordList = new ArrayList();
int totalCount = 1;
    String editPersonUrl = request.getParameter("editPersonUrl");
    MultiValueMap paramsmap = new MultiValueMap();
    paramsmap.put(NSApiClientConstants.QUERYPARAM_START_ROW, "" + start);
    paramsmap.put(NSApiClientConstants.QUERYPARAM_RECORD_SIZE, "" + limit);
    PersonList personList = nsApiClient.getDirectory().getPeople(paramsmap);
if (personList.getPeople() != null) {
for (Iterator iterator = personList.getPeople().iterator(); iterator.hasNext();) {
    Person portalPerson = (Person) iterator.next();
portalPerson.setPersonURL(StringEscapeUtils.escapeXml(portalPerson.getPersonURL()));
    PortletURL editPersonURL = response.createRenderURL();
editPersonURL.setParameter("personId", "" + portalPerson.getPersonId());
    String firstNameUrl = "<a href='" + editPersonURL.toString() + "'" + ">" +
portalPerson.getFirstName() + "</a>";
    String lastNameUrl = "<a href='" + editPersonURL.toString() + "'" + ">" +
portalPerson.getLastName() + "</a>";
portalPerson.setFirstName(firstNameUrl);
portalPerson.setLastName(lastNameUrl);
recordList.add(portalPerson);
    }
}
jsonMap.put("success", "true");
jsonMap.put("results", personList.getTotalCount());

```

```

jsonMap.put("rows", recordList);
    JSON json = (JSON) JSONSerializer.toJSON(jsonMap);
    String jsonStr = json.toString();

response.setContentType("text/plain");
response.getPortletOutputStream().write(jsonStr.getBytes());
response.getPortletOutputStream().flush();
    }
}

private ModelAndView addPersonData(@ModelAttribute("personData") Person person,
    ResourceRequest request, ResourceResponse response) throws Exception {
    Add Person from Form Data in Request
    Map jsonMap = newHashMap();
    try {
    Person Updateperson = nsApiClient.getDirectory().updatePerson(person);
    jsonMap.put("success", "true");
    jsonMap.put("successMsg", "Person Added/Updated Successfully");
    jsonMap.put("rows", Updateperson);
        JSON json = (JSON) JSONSerializer.toJSON(jsonMap);
        String jsonStr = json.toString();

response.setContentType("text/plain");
response.getPortletOutputStream().write(jsonStr.getBytes());
response.getPortletOutputStream().flush();
    } catch (Exception e) {
e.printStackTrace();
    jsonMap.put("success", "false");
    jsonMap.put("errorMsg", "Person Add/Update Failed : " + e.getMessage());

        JSON json2 = (JSON) JSONSerializer.toJSON(jsonMap);
        String jsonStr2 = json2.toString();

response.setContentType("text/plain");
response.getPortletOutputStream().write(jsonStr2.getBytes());
response.getPortletOutputStream().flush();
    }

returnnull;
    }

private void editPerson(RenderRequest request, RenderResponse response, Model model, int
personId) {
int person = personId;
    Map<String, Object> jsonMap = new HashMap<String, Object>();
    try {
    Person persons = nsApiClient.getDirectory().getPersonById(person);
    persons.setPersonURL(StringEscapeUtils.escapeXml(persons.getPersonURL()));
        JSON json = (JSON) JSONSerializer.toJSON(persons);
        String jsonStr = json.toString();
    model.addAttribute("PersonData", jsonStr);
    } catch (Exception e) {
e.printStackTrace();
    }
}

private String showAddPersonPage(RenderRequest request, RenderResponse response, Model
model) {
    Person DummyPerson = newPerson();
    DummyPerson.setPersonURL(StringEscapeUtils.escapeXml(DummyPerson.getPersonURL()));
    JSON json1 = (JSON) JSONSerializer.toJSON(DummyPerson);
    String jsonStr1 = json1.toString();
    model.addAttribute("PersonData", jsonStr1);
}

```



```
return viewPageUpdate;
    }

    public String editNormal(RenderRequest request, RenderResponse response, Model model) {
        try {
            super.editNormal(request, response, model);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return editPage;
    }

    public String editMinimized(RenderRequest request, RenderResponse response, Model model) {
        try {
            super.editMinimized(request, response, model);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return editPage;
    }

    public String editMaximized(RenderRequest request, RenderResponse response, Model model) {
        try {
            super.editMaximized(request, response, model);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return editPage;
    }

    public String helpNormal(RenderRequest request, RenderResponse response, Model model) {
        try {
            super.helpNormal(request, response, model);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return helpPage;
    }

    public String helpMinimized(RenderRequest request, RenderResponse response, Model model) {
        try {
            super.helpMinimized(request, response, model);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return helpPage;
    }

    public String helpMaximized(RenderRequest request, RenderResponse response, Model model) {
        try {
            super.helpMaximized(request, response, model);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return helpPage;
    }

    private NSApiClient getNSApiClient() {
        return NSApiClientFactory.getInstance();
    }
}
```

MyJSRApplicationContext.xml

Spring application context XML for the portlet.

```
<?xmlversion="1.0"encoding="UTF-8"?>
<beansxmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"xmlns:p="http://www.springframework.org/s
chema/p"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-2.5.xsd">

<beanid="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<propertyname="cache"value="true"/>
<propertyname="viewClass"
value="org.springframework.web.servlet.view.JstlView"/>
<propertyname="prefix"value="/WEB-INF/jsp"/>
<propertyname="suffix"value=".jsp"/>
</bean>
<context:annotation-config/>
<bean
class="org.springframework.web.portlet.mvc.annotation.DefaultAnnotationHandlerMapping">
<propertyname="interceptors">
<bean
class="org.springframework.web.portlet.handler.ParameterMappingInterceptor"/>
</property>
</bean>
<beanid="MyJSRController"class="com.myjsr.MyJSRController">
</bean>
</beans>
```

jsrportlet.properties

URL of the MyJSRServer for the use by nsAPI.



Note

In a clustered environment, if the portlet references the Service Portal application URL, then specify the URL as “http://localhost:<port>/RequestCenter” where <port> is the port number used by each node 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 one of the hosts within the cluster.

```
 #(Port number and host has to changed as per the application server).
BASE_URL=http://localhost:8088/RequestCenter
```

Log4j.properties

```
log4j.rootCategory=INFO,CONSOLE

log4j.appender.CONSOLE=org.apache.log4j.ConsoleAppender
log4j.appender.CONSOLE.layout=org.apache.log4j.PatternLayout
log4j.appender.CONSOLE.layout.ConversionPattern=%d{ABSOLUTE}%-5p[%c{1}]:%L}%m%n
```

jboss-deployment-structure.xml

```
<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>
```

Compiling JSR Portlet Controller

Include the following libraries in the classpath when compiling the portlet controller:

1. newscale_compbeans.jar
2. newscale_conf.jar
3. newscale_core.jar
4. newscale_appclient.jar
5. pluto-container-2.0.2.jar
6. pluto-portal-driver-2.0.2.jar
7. pluto-portal-driver-impl-2.0.2.jar
8. pluto-container-api-2.0.2.jar
9. pluto-container-driver-api-2.0.2.jar
10. pluto-taglib-2.0.2.jar
11. org.springframework.aop-3.0.2.RELEASE.jar
12. org.springframework.asm-3.0.2.RELEASE.jar
13. org.springframework.aspects-3.0.2.RELEASE.jar
14. org.springframework.beans-3.0.2.RELEASE.jar
15. org.springframework.context-3.0.2.RELEASE.jar
16. org.springframework.context.support-3.0.2.RELEASE.jar
17. org.springframework.core-3.0.2.RELEASE.jar
18. org.springframework.expression-3.0.2.RELEASE.jar
19. org.springframework.instrument-3.0.2.RELEASE.jar
20. org.springframework.instrument.tomcat-3.0.2.RELEASE.jar
21. org.springframework.jdbc-3.0.2.RELEASE.jar
22. org.springframework.jms-3.0.2.RELEASE.jar
23. org.springframework.orm-3.0.2.RELEASE.jar
24. org.springframework.oxm-3.0.2.RELEASE.jar
25. org.springframework.spring-library-3.0.2.RELEASE.libd
26. org.springframework.test-3.0.2.RELEASE.jar
27. org.springframework.transaction-3.0.2.RELEASE.jar
28. org.springframework.web-3.0.2.RELEASE.jar
29. org.springframework.web.portlet-3.0.2.RELEASE.jar
30. org.springframework.web.servlet-3.0.2.RELEASE.jar
31. org.springframework.web.struts-3.0.2.RELEASE.jar
32. commons-collections-3.2.1.jar
33. commons-lang-2.4.jar
34. j2ee.jar
35. json-lib-2.2.2-jdk13.jar
36. portlet-api_2.0_spec-1.0.jar

Portlet Deployment

The deployment procedures vary with the application server used. As a general note, a JSR portlet can be deployed like any typical web application through the application server administrator console.

For detailed instructions on how to deploy the JSR portlet, and how to use the portlet on a portal page after deployment, see the *Cisco Service Portal Designer Guide*.



INDEX

A

Adapters [2-8, 2-42 to 2-59](#)

- Auto-Complete Adapter [2-43](#)
- Components [3-5](#)
- Database [2-43](#)
- Defined [2-2, 3-4](#)
- Dummy Adapter [2-43](#)
- File Adapter [2-48](#)
- HTTP/WS [2-49](#)
- JMS Adapter [2-54](#)
- MQ Adapter [2-55](#)
- Properties [3-6](#)
- Remedy [4-1](#)
- Service Item Listener Adapter [2-56](#)
- Types [3-5](#)
- VMware Adapter [2-57](#)
- Web Service Listener Adapter [2-58](#)

ADK. See Service Link Adapter Development Kit (ADK).

Agents [2-10](#)

- Adapter Selection [2-12](#)
- Creating [2-10 to 2-11](#)
- Defined [2-3](#)
- Dictionary Mappings [2-15](#)
- Failed Email Notification [2-11](#)
- Outbound Message Content [2-11](#)
- Parameters [2-13](#)
- Prebuilt Functions [2-17](#)
- Starting and Stopping Agents [2-66](#)

Apache CXF [5-4](#)

Apache Pluto [7-1](#)

Application Programming Interface (API) [1-2](#)

Authentication, Web Services [5-6 to 5-7](#)

Authentication Method [1-3](#)

Auto-Complete Adapter [2-43](#)

Axis 2 [5-4](#)

B

BindDN [1-3](#)

BMC Remedy Adapter. See Remedy Adapter.

Business Engine [2-3](#)

C

Capability, Manage Global Settings [1-2](#)

Certificates, Configuring [1-24](#)

Configuration Files, Application Server [2-68](#)

Connection Mechanism [1-3](#)

Custom Code [1-36 to 1-48](#)

- Coding Guidelines [1-49](#)
- Configuring in the Administration Module [1-49](#)
- Deploying [1-51](#)
- Operation Interfaces [1-38](#)
- Operations [1-19, 1-37](#)

Custom Java Class Mapping Interface [1-44](#)

Custom Mappings [1-8](#)

D

Database Adapter [2-43](#)

- Database Connection [2-43](#)

Datasources

- Adding or Editing [1-22](#)
- Configuring [1-22](#)
- Configuring Certificates [1-24](#)

Configuring Connection Information [1-23](#)
 Configuring Referral Datasources [1-25](#)
 Defining [1-3](#)
 SQL [1-51](#)
 Testing the Connection [1-25](#)
 Dictionaries, Defined [2-3](#)
 Directory Attribute [1-5](#)
 Directory Integration [1-1](#)

- Configuring [1-20, 1-22 to 1-34](#)
- Configuring Events [1-34](#)
- Enabling [1-20](#)
- Operations [1-36, 1-37](#)
- Using Custom Code [1-36 to 1-48](#)

 Directory Map Testing, Enabling [1-31](#)
 Directory Server API [1-45](#)
 Dummy Adapter [2-43](#)

E

Email, Failed Email Notification [2-11](#)
 Enable Web Services [5-2](#)
 Encryption, Web Services [5-7](#)
 Error Messages, Web Services [5-40 to 5-42](#)
 Events [1-9](#)

- Configuring [1-34](#)
- Configuring Events [1-35](#)
- Sample Event Configuration [1-54](#)

 Expression Mapping [1-28](#)
 External Authentication Operation [1-9](#)
 External Tasks

- Creating [2-23 to 2-24](#)
- Filter and Search [2-40](#)
- Viewing [2-39 to 2-41](#)

 External User Authentication (EUA) Operation [1-12](#)
 Ext JS [6-1](#)

F

Failed Messages, Resending [2-38](#)
 File Adapter [2-48](#)
 Filter and Search [2-37, 2-40](#)
 Filters, nsAPI [6-3 to 6-6](#)

H

HTTP/WS Adapter [2-49](#)

I

Import/Refresh Person API [1-47](#)
 Import Manager Operation [1-9, 1-16 to 1-19](#)
 Import Person Operation [1-9, 1-16](#)
 Inbound and Outbound Documents, Sample [3-24 to 3-31](#)
 Integration

- Components [2-65](#)
- Events [1-9](#)
- Operations [1-9](#)

 Integration Wizard [2-2, 2-3, 2-4, 2-59 to 2-65](#)

- Integration Summary [2-64](#)
- Outbound Request Parameter Mappings [2-63](#)
- Outbound Response Parameter Mappings [2-64](#)
- Using [2-60](#)

J

Java Development Kit (JDK) [3-2](#)
 Java Portlet Specification [7-1](#)
 JDK. See Java Development Kit (JDK).
 JMS Adapter [2-54](#)
 JSR Portlets [7-1](#)

- Deployment [7-28](#)
- Development [7-6](#)
- Structure and Packaging [7-1](#)

L

- LDAP [1-1, 1-3](#)
- Logging [2-66](#)
 - JBoss [2-66](#)
 - Online Error Log [2-68](#)
 - Remedy Adapter Messages [4-12](#)
 - WebLogic [2-67](#)
 - WebSphere [2-67](#)
- Login Event [1-9, 1-10](#)
 - Custom Code [1-39](#)

M

- Mappings
 - Configuring [1-26 to 1-31](#)
 - Custom [1-8](#)
 - Custom Java Class Mapping Interface [1-44](#)
 - Data Mapping Test Controls [1-33](#)
 - Defined [1-4](#)
 - Expression Mapping [1-28](#)
 - Java Class Mapping [1-31](#)
 - Mandatory [1-5](#)
 - Optional [1-6](#)
 - Sample Mapping Configuration [1-53](#)
 - Simple and Composite [1-28](#)
 - Testing [1-31 to 1-34](#)
 - Time Zones [1-6, 1-63](#)
 - Types [1-28](#)
- Message Details Popup [2-36](#)
- Messages
 - Filter and Search [2-37](#)
 - Purging [2-67](#)
 - Sending a Manual Message [2-40](#)
- MQ Adapter [2-55](#)

N

- Name Search [6-5](#)
- nsAPI [6-1](#)
 - Conventions and Syntax [6-2](#)
 - Filters [6-3 to 6-6](#)
 - Nested Entities [6-8 to 6-9](#)
 - Operations [6-2](#)
 - Paging [6-7 to 6-8](#)
 - Sorting [6-6 to 6-7](#)
 - Supported Entities [6-1](#)
- nsXML [2-3](#)
 - Messages [2-25](#)
 - Transformations [2-32](#)
- nsXML Format [3-13 to 3-23](#)
 - Agent Parameter [3-23](#)
 - Data Values [3-19](#)
 - Dictionary [3-21](#)
 - Form [3-22](#)
 - Message [3-13](#)
 - Requisition [3-17](#)
 - Requisition Entry [3-18](#)
 - Task [3-15](#)
 - Task Started or Task Cancelled [3-14](#)

O

- Online Error Log [2-68](#)
- OOB. See Order-On-Behalf (OOB).
- Operations [1-9](#)
 - Custom Code [1-37](#)
 - Directory Integration [1-36](#)
 - nsAPI [6-2](#)
- Order-On-Behalf (OOB) [1-1](#)

P

Person Lookup

- Custom Code [1-43](#)

- Event [1-9, 1-13](#)

Person Search Operation [1-9, 1-13 to 1-15](#)Portal Manager [7-1](#)Portfolio Management, Web Services [5-19 to 5-20](#)

Portlets

- JSR [7-1](#)

- Third-Party [7-1](#)

Prebuilt Functions [2-17, 2-69 to 2-71](#)Protocol [1-3](#)Purge, Messages [2-67](#)**R**

RAPI 2. See Requisition API.

RBAC. See Role-Based Access Control (RBAC).

Recent Failed Messages [2-6, 2-7, 2-34](#)Remedy Adapter [4-1](#)

- Configuration Steps [4-4](#)

- Configuring the Agent [4-5 to 4-7](#)

- Configuring the Transformation [4-7 to 4-11](#)

- Designing a Service [4-11](#)

- Installation [4-5](#)

- Integration Scenarios [4-2](#)

- Log Messages [4-12](#)

- Viewing [4-5](#)

- XSL Transformation (XSLT) [4-2](#)

Representational State Transfer. See REST API.

Requisition API [5-1](#)REST API [6-1](#)

- API Reference [6-16 to 6-48](#)

- Invoking [6-9 to 6-15](#)

- Quick Reference [6-49](#)

Role-Based Access Control (RBAC) [1-5, 6-1](#)**S**SASL (Simple Authentication and Security Layer) [1-3](#)Server Log File [2-66](#)Service Item Listener Adapter [2-56](#)

Service Link

- Accessing [2-5](#)

- Adapters [2-8, 2-42 to 2-59](#)

- Agents [2-10](#)

- Business Engine [2-3](#)

- Components [2-2](#)

- Configuring a Task to use a Service Link Agent [2-23 to 2-25](#)

- Creating and Deploying a Service Link Agent [2-33 to 2-34](#)

- Defined [2-1](#)

- Home Page [2-6](#)

- Integration Wizard [2-59 to 2-65](#)

- Logging [2-66](#)

- nsXML [2-3](#)

- Prebuilt Functions [2-69 to 2-71](#)

- Recent Failed Messages [2-7, 2-34](#)

- Resending Failed Messages [2-38](#)

- Screens [2-7](#)

- Status [2-6, 2-66](#)

- Viewing Messages [2-35](#)

Service Link Adapter Development Kit (ADK) [2-1, 3-1](#)

- Compiling Adapters [3-3](#)

- Creating Adapter Source Structures [3-3](#)

- Deploying Adapters [3-4](#)

- Example Adapter [3-6 to 3-13](#)

- Installing [3-2](#)

- Subdirectories [3-2](#)

Single Sign-On (SSO) [1-1, 1-2, 1-9, 1-10 to 1-12](#)

- Administrative Bypass [1-11](#)

- Customizing [1-39](#)

SQL-Based Person Lookup, Sample Code [1-55](#)SQL Datasource [1-51](#)

SSO. See Single Sign-On (SSO).

Status, Service Link [2-6](#), [2-66](#)

T

Third-Party Portlets [7-1](#)

Time Zones

 Mappings [1-6](#), [1-63](#)

 Supported [1-63](#)

Transformations [2-3](#)

 Creating [2-20 to 2-22](#)

V

VMware Adapter [2-57](#)

W

Web Service Listener Adapter [2-58](#)

Web Services [5-1](#)

 Adding Comments to a Requisition [5-14](#)

 Approving or Rejecting an Authorization [5-18](#)

 Authentication [5-6 to 5-7](#)

 Cancelling a Requisition [5-15](#)

 Enabling [5-2](#)

 Encryption [5-7](#)

 Error Messages [5-40 to 5-42](#)

 Generating Code [5-4](#)

 Getting a List of Authorizations [5-16](#)

 Getting a List of Requisitions [5-12](#)

 Getting the Requisition Status [5-13](#)

 Getting the Service Definition [5-8 to 5-10](#)

 Portfolio Management [5-19 to 5-20](#)

 Roles and Capabilities [5-3](#)

 Sample Requests and Responses [5-21 to 5-40](#)

 Sample Service Definition [5-5](#)

 Submitting a Requisition [5-10 to 5-12](#)

 Task Management [5-16 to 5-19](#)

 Testing [5-4](#)

 WSDL [5-2](#)

Wildcard Character [1-15](#), [6-5](#)

WSDL [5-2](#)

