

Getting Started with IBM WebSphere Cast Iron Cloud Integration

Learn how to integrate applications,
both on-premise and in the cloud

Learn about WebSphere Cast Iron
Integration Solution technology

Design, build, and manage
integration solutions



Lars Besselmann-Hamandouche
Nick Bletzer
Simon Dickerson
Leonardo Rodriguez Leon
Roberto Mascarenhas
Giuliano Diniz de Moraes
Rajath Ramesh
Carla Sadtler

Redbooks



International Technical Support Organization

Getting Started with IBM WebSphere Cast Iron Cloud Integration

January 2012

Note: Before using this information and the product it supports, read the information in “Notices” on page xi.

First Edition (January 2012)

This edition applies to IBM WebSphere Cast Iron Cloud Integration Version 6.1.

© Copyright International Business Machines Corporation 2012. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Trademarks	xii
Preface	xiii
The team who wrote this book	xiii
Now you can become a published author, too!	xv
Comments welcome	xvi
Stay connected to IBM Redbooks	xvi
Chapter 1. Introduction and technical overview.....	1
1.1 Integrating cloud applications	2
1.2 Example use cases	3
1.3 The WebSphere Cast Iron approach to integrating applications.....	4
1.4 Architecture overview	6
1.4.1 On-premise Integration Appliance	6
1.4.2 Cast Iron Live	6
1.5 Developing integrations.....	9
1.5.1 Connectors	10
1.5.2 Secure communications	11
1.5.3 Data variables	12
1.6 Activities and configurations	13
1.6.1 Transforming and mapping data	15
1.6.2 Defining a starter activity for an orchestration job	16
1.6.3 Error handling	17
1.6.4 Testing	17
1.6.5 Reuse: Template Integration Projects	19
1.6.6 Deploying projects	20
1.6.7 Updating projects	20
1.7 Management and monitoring	21
1.8 Availability and scaling	22
1.9 Upgrading to a new release	22
1.10 Cast Iron editions	22
1.11 Express edition	23
1.12 Terminology	24
Chapter 2. Installing and setting up WebSphere Cast Iron integration	27
2.1 Installing and configuring WebSphere Cast Iron Studio	28
2.1.1 Installing Studio	28
2.1.2 Changing the user interface language	34
2.1.3 Installing module providers	36
2.1.4 Installing plug-in connectors	37
2.2 Installing and configuring the physical appliance	38
2.2.1 Installing and starting the appliance	39
2.2.2 Gathering basic information about the appliance	40
2.2.3 Setting up the appliance	41
2.3 Configuring the physical appliance for high availability	44
2.3.1 Prerequisites to set up high availability	45
2.3.2 Configuring the appliances for HA	46
2.3.3 Managing physical appliances in an HA configuration	49

2.4	Installing and configuring the virtual appliance	49
2.4.1	Installing and starting WebSphere Cast Iron Hypervisor Edition	50
2.4.2	Gathering basic information about the appliance	51
2.4.3	Setting up the appliance using DHCP	52
2.4.4	Setting up the appliance using static network settings	53
2.5	Finalizing the installation using the WMC	55
2.5.1	Logging in to the WMC	55
2.5.2	Installing the required SAP module providers	59
2.6	Installing the WebSphere Cast Iron Live Secure Connector	60
2.6.1	Creating a Secure Connector in Cast Iron Live	62
2.6.2	Downloading the Secure Connector configuration and the installer files	63
2.6.3	Installing the Secure Connector on a Windows operating system	64
2.6.4	Starting the Secure Connector and verifying communication	72
2.6.5	Installing third-party libraries	73
2.6.6	Installing the Secure Connector on Linux	76
2.7	Upgrading from WebSphere Cast Iron Cloud Integration V6.0 to V6.1	76
2.7.1	Upgrading WebSphere Cast Iron Studio V6.0	77
2.7.2	Upgrading the physical appliance	77
2.7.3	Upgrading the virtual appliance	77
2.7.4	Upgrading the cloud	78
Chapter 3.	Developing and unit testing with WebSphere Cast Iron Studio	79
3.1	Overview of Cast Iron development	80
3.2	Overview of WebSphere Cast Iron Studio	81
3.3	Projects	84
3.3.1	Project toolbox tab	87
3.3.2	Configuration properties	88
3.3.3	Project directory structure	91
3.3.4	Versioning projects	92
3.3.5	Project permissions	94
3.3.6	Project documentation	94
3.4	Orchestrations	95
3.4.1	The orchestration editor	96
3.4.2	What makes a valid orchestration	98
3.4.3	Orchestration properties	99
3.4.4	Orchestration documentation	100
3.5	Connectors and endpoints	102
3.5.1	Connectors	102
3.5.2	Endpoints	103
3.5.3	Installing module providers	106
3.6	Activities	107
3.6.1	Configuring and using an activity	108
3.6.2	Starter activities	111
3.6.3	Outbound activities	111
3.6.4	Transform activities	112
3.6.5	Logic and utility activities	112
3.6.6	Local web services calls	113
3.6.7	Database activities, assets, and the staging database	114
3.6.8	Using HTTP and web service starter activities with Cast Iron Live	116
3.7	Variables	117
3.7.1	Creating a variable	119
3.7.2	Shared variables	121
3.7.3	Deleting a variable	122

3.7.4 The JobInfo variable and Job Keys.....	122
3.8 Functions and lookup tables	123
3.8.1 Custom functions	124
3.8.2 Lookup tables	126
3.9 Maps	128
3.9.1 The mapping editor.....	129
3.9.2 Using functions in maps	141
3.9.3 The Map Variables activity and standalone maps.....	146
3.9.4 XML stylesheets	149
3.10 Parsing data	149
3.10.1 Maps and functions.....	150
3.10.2 XML parsing	150
3.10.3 MIME parsing	151
3.10.4 Flat File Schema parsing	151
3.11 Exception handling	155
3.11.1 Try activity.....	156
3.11.2 Global exception handler	158
3.11.3 Common Error Handler.....	159
3.12 Unit testing	159
3.12.1 Endpoint Test Connection.....	159
3.12.2 Test Flat File Schema	160
3.12.3 Test maps	161
3.12.4 Test Custom XSLT	163
3.12.5 XPath Evaluator	163
3.12.6 Validating orchestrations.....	164
3.12.7 Verifying orchestrations	164
3.12.8 HTTP Post Utility tool	168
3.12.9 Invoke Service tool	169
3.13 Exporting and publishing.....	170
3.13.1 Exporting a project	170
3.13.2 Publishing a project directly from Studio.....	171
Chapter 4. Management and monitoring.....	173
4.1 Management	174
4.1.1 Command Line Interface.....	174
4.1.2 Web Management Console	175
4.1.3 Management API	179
4.2 Project configuration life cycle.....	180
4.2.1 Project configurations	180
4.2.2 Starting a configuration	181
4.2.3 Pausing a configuration	183
4.2.4 Stopping a configuration	183
4.2.5 Undeploying a configuration	184
4.2.6 Changing configuration properties	185
4.2.7 Disabling orchestrations	186
4.2.8 Cloning configurations	188
4.2.9 Deleting configurations	189
4.2.10 Versioning	189
4.2.11 Assets	190
4.2.12 Scheduling downtime	194
4.2.13 Uploading and downloading a project.....	197
4.2.14 Importing and exporting repository configuration	198
4.3 Monitoring and troubleshooting	198

4.3.1 Logging overview	199
4.3.2 The system log repository.....	200
4.3.3 Job log	206
4.3.4 Notifications	213
4.4 Network.....	216
4.5 The staging database.....	217
4.6 Resource utilization.....	221
4.7 Commands	225
4.8 Security.....	226
4.9 Appliance and connector upgrades.....	227
4.9.1 Verifying the current version	227
4.9.2 Upgrading the Integration Appliance.....	228
4.9.3 Update connector libraries	229
Chapter 5. Security	231
5.1 WebSphere Cast Iron users and groups.....	232
5.1.1 Viewing and updating your user profile.....	233
5.1.2 Creating a new user	233
5.1.3 Logging in with a new user ID.....	237
5.1.4 Changing your password	237
5.1.5 Creating a new group	240
5.1.6 Assigning users to groups.....	241
5.1.7 Deleting a user	242
5.1.8 Deleting a group	243
5.2 Granting project configuration permissions.....	243
5.3 Permissions to publish projects from Studio	244
5.4 Configuring WebSphere Cast Iron to use LDAP authentication	246
5.5 Working with certificates	250
5.5.1 Managing certificates	252
5.6 Securing communication with endpoints.....	262
5.6.1 The WebSphere Cast Iron Secure Connector	262
5.6.2 Authentication mechanisms	266
5.7 Monitoring security	275
5.8 Additional security considerations.....	276
Chapter 6. Availability and scalability planning.....	277
6.1 High availability techniques for Integration Appliances.....	278
6.1.1 Integration Appliance availability features.....	278
6.1.2 Integration Appliance high availability modes	279
6.1.3 Orchestration design for availability	281
6.2 Scalability techniques for integration appliances	282
6.3 Maintenance and updates.....	284
6.3.1 Rolling out updates	284
6.3.2 Scheduling down time.....	285
6.3.3 Minimizing downtime when updating an orchestration	285
6.4 Backup and restore	285
6.4.1 Using an image or a snapshot and restore for virtual appliances.....	285
6.4.2 Exporting or importing the Integration Appliance configuration.....	286
6.4.3 Recovering a physical or virtual appliance	289
6.4.4 Recovering projects from the WMC	290
Chapter 7. Reusability with Template Integration Projects.....	293
7.1 Overview of Template Integration Projects	294
7.2 Introduction to the TIP Configuration Editor	296

7.2.1 Steps panel	297
7.2.2 Workspace panel	297
7.3 Creating and modifying TIPs	298
7.4 Uploading TIPs	305
7.5 Searching and downloading TIPs	309
7.6 Verifying the TIP	312
7.7 Rating and reviewing TIPs	318
7.7.1 Rating and reviewing in the TIP configuration	318
7.7.2 Rating and reviewing using Publish Review	319
Chapter 8. Building custom plugin connectors	321
8.1 Introduction to the CDK	322
8.2 The CDK components	324
8.2.1 Schema files	325
8.2.2 XML file	325
8.2.3 WSDL file	327
8.2.4 The .par file	329
8.3 Developing a CDK Connector	329
8.3.1 Using the CDK Wizard to create a connector	329
8.3.2 Developing a Google Calendar connector using the CDK Wizard	333
8.4 Publishing the CDK Connector to a local repository	342
8.5 Testing and debugging the CDK Connector	342
8.6 Sharing the CDK Connector	344
8.6.1 Exporting a CDK Connector	345
8.6.2 Importing a CDK Connector	345
8.6.3 Downloading a CDK Connector	345
Chapter 9. Common error handlers	349
9.1 The principles of a common error handler	350
9.2 The common error handler Template Integration Project	350
9.3 Altering the common error handler TIP	352
9.4 Using the common error handler TIP	353
Chapter 10. Scenario: Bidirectional account synchronization	357
10.1 Cast Iron concepts demonstrated in this scenario	358
10.2 Scenario overview	358
10.3 Prerequisites	359
10.4 Overview of the use cases for this scenario	360
10.5 Use case 1: Synchronizing from SAP to Salesforce.com	360
10.5.1 Downloading the TIP	360
10.5.2 Using the TIP configuration wizard	361
10.5.3 Customizing the orchestration	376
10.5.4 Reducing the SAP result set	385
10.5.5 Naming conventions and other definitions	389
10.5.6 Testing end-to-end	392
10.5.7 Using the Common Error Handler	397
10.5.8 Synchronizing dedicated SAP customers to Salesforce.com	399
10.5.9 Deploying and testing the orchestration on the Integration Appliance	400
10.5.10 Initial load of thousands of records	405
10.6 Use Case 2: Synchronizing from Salesforce.com to SAP	407
10.6.1 Building the orchestration	407
10.6.2 Deploying and testing the orchestration on the Integration Appliance	429
10.6.3 Enhancing the orchestration to avoid feedback loops	429

Chapter 11. Scenario: CRM to cloud calendar services	435
11.1 WebSphere Cast Iron concepts demonstrated in this scenario	436
11.2 Scenario overview	436
11.2.1 Prerequisites	436
11.2.2 Scenario description	437
11.3 Signing in to Cast Iron Live	437
11.4 Creating a project	438
11.5 Modifying the orchestration	445
11.5.1 Add the HTTP Receive Request starter activity to the orchestration	445
11.5.2 Performing an initial test on the NetSuite activity	448
11.5.3 Adding a response to the HTTP Receive Request	451
11.5.4 Adding a response if no events are to be synchronized	454
11.5.5 Catching errors that occur while searching NetSuite	455
11.5.6 Error handling	460
11.6 Publishing and deploying the project to the development environment	460
11.6.1 Changing the job schedule time	460
11.6.2 Publishing the project	461
11.6.3 Deploying the project	461
11.6.4 Testing the project	462
11.6.5 Connecting to Google	463
11.6.6 Google Connector	463
Chapter 12. Scenario: Data enrichment and aggregation	465
12.1 Cast Iron concepts demonstrated in this scenario	466
12.2 Scenario overview	466
12.3 The order file structure	468
12.4 The item structure	469
12.5 The product database structure	470
12.6 The Domino document structure	470
12.7 The scenario orchestration	470
12.8 Preparing the scenario	473
12.8.1 Preparing the FTP server	474
12.8.2 Preparing the DB2 database	474
12.8.3 Preparing the Domino database	474
12.9 Creating entities for the scenario	475
12.9.1 Creating and testing an FTP endpoint	475
12.9.2 Creating and testing the DB2 endpoint	476
12.9.3 Creating and testing the Domino endpoint	477
12.9.4 Creating the Orders flat file schema	478
12.9.5 Creating the Items flat file Schema	482
12.9.6 Creating a custom function	485
12.9.7 Creating a lookup table	487
12.10 Building the orchestration	488
12.10.1 Creating job keys	489
12.10.2 Polling the FTP directory	490
12.10.3 Creating and renaming orchestration variables	491
12.10.4 Parsing the input data	493
12.10.5 Transforming the data with an XML stylesheet	495
12.10.6 Adding a Lookup activity	497
12.10.7 Mapping outputs for the Lookup activity	499
12.10.8 Adding a Merge activity	502
12.10.9 Adding a For Each activity	504
12.10.10 Adding the Create Documents activity	505

12.10.11 Logging the job keys using the Create Job Keys activity	511
12.10.12 Testing the orchestration	513
12.10.13 Adding error handling	513
Appendix A. Additional material	517
Locating the Web material	517
Using the Web material	517
Related publications	519
IBM Redbooks	519
Online resources	519
Help from IBM	520

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Cast Iron®
DataPower®
DB2®
developerWorks®
Domino®

IBM®
Lotus Notes®
Lotus®
Maximo®
Notes®

RDN®
Redbooks®
Redbooks (logo) ®
Tivoli®
WebSphere®

The following terms are trademarks of other companies:

Adobe, the Adobe logo, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

Cloud computing provides companies with many capabilities to meet their business needs but can also mean that a hybrid architecture is created that includes on-premise systems and the cloud. Integration is needed to bridge the gap between the on-premise existing systems and the new cloud applications, platform, and infrastructure.

IBM® WebSphere® Cast Iron® meets the challenge of integrating cloud applications with on-premise systems, cloud applications-to-cloud applications, and on-premise to on-premise applications. It contains a graphical development environment that provides built-in connectivity to many cloud and on-premise applications and reusable solution templates that can be downloaded from a solution repository. The integration solutions that are created can then run on either an on-premise integration appliance or the multi-tenant WebSphere Cast Iron Live cloud service.

This IBM Redbooks® publication is intended for application integrators, integration designers, and administrators evaluating or already using IBM WebSphere Cast Iron. Executives, leaders, and architects who are looking for a way to integrate cloud applications with their on-premise applications are also shown how WebSphere Cast Iron can help to resolve their integration challenges.

The book helps you gain an understanding of Cast Iron and explains how to integrate cloud and on-premise applications quickly and simply. It gives a detailed introduction to the development tool and the administration interfaces and how they are used. It also discusses security, high availability, and re-usability. The book also includes three detailed scenarios covering real-world implementations of a Cast Iron Integration Solution.

The team who wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Lars Besselmann-Hamandouche has been a member of the WebSphere Client Technical Professional Team in Duesseldorf, Germany for more than 10 years, working with clients from all industries. He joined IBM as an IBM Systems Engineer in 1994 and became a WebSphere Technical Professional in 2000. He is an IBM and The Open Group Master Certified IT Specialist, and he holds a degree in mathematics from the University of Dortmund. His current focus is on application infrastructure and application integration. For over a year, he has the role of a specialist for WebSphere Cast Iron and worked with several business applications, such as Salesforce.com, SAP, and Oracle CRM On Demand. He published an article on IBM developerWorks® about application integration using WebSphere Cast Iron. You can contact Lars at Lars.Besselmann@de.ibm.com.

Nick Bletzer is an IBM Certified WebSphere Trainer in the United Kingdom (UK). He joined IBM in 1992 and has worked on technologies as diverse as Rexx, AS400, and Java. He is currently the European lead instructor for IBM WebSphere DataPower SOA Appliances and IBM WebSphere Cast Iron Cloud Integration. He also teaches WebSphere Message Broker and WebSphere MQ.

Simon Dickerson is a technical pre-sales consultant working for IBM in the UK and Ireland, dedicated to Cast Iron. He worked in a variety of IT roles from training, support, consulting,

and focused technical sales for over 10 years. He worked across a number of application areas including content management and document security, mobile applications, and data quality on a variety of platforms and across many industries. Simon holds a degree from the University of Reading in Cybernetics and Control Engineering with Mathematics.

Leonardo Rodriguez Leon began at IBM Venezuela in 2002 working as an IT Specialist for Operating Systems, such as Linux and Windows on multi platforms. In 2004, he moved to the Software Group department working as a WebSphere IT Specialist for Venezuela. In 2006, he was in charge of IBM WebSphere DataPower SOA Appliances for Latin America as an IT Specialist. He is currently an Advisory Software Engineer at IBM Mexico for Software Solutions Lab in the Connectivity area covering the Latin America region.

Roberto Mascarenhas is a Senior IT Specialist at IBM Software Group in Rio de Janeiro, Brazil. He has 15 years of experience in the IT field and has worked with WebSphere Business Integration solutions involving mainly the Connectivity portfolio as WebSphere MQ, WebSphere Message Broker, WebSphere DataPower®, WebSphere MQFTE, WebSphere Adapters, and also business-to-business (B2B) solutions for the last ten years. He also works to accelerate and take advantage of new IBM solutions that arise from acquisitions. Before joining the WebSphere Pre-Sales team, he worked on several projects for the IBM Networking Hardware Division and IBM Storage Systems Group. He studied Electronic Engineering at Federal University of Rio de Janeiro (UFRJ).

Giuliano Diniz de Moraes is an Advisory Software Engineer at IBM Brazil Software Labs. He joined IBM in 2008 and started at IBM as a Developer for IBM Tivoli® Maximo®. He is currently part of the WorldWide Technical Professionals team for WebSphere working with Business Partner enablement across Latin America, focusing on WebSphere integration appliances. He is also part of the developerWorks Brazil technical board. Giuliano holds certifications in Java, WebSphere Application Server, IBM WebSphere DataPower SOA Appliances, and IBM WebSphere Cast Iron Cloud Integration. Giuliano also teaches programming at Faculdade de Tecnologia de São Paulo (Fatec) and is obtaining a Masters Degree in Computer Graphics at Universidade Federal de Uberlandia.

Rajath Ramesh is a Software Engineer at IBM India Software labs. He started his career with IBM in 2010 and started as Developer in WebSphere Adapters and incubation projects. He is currently part of the WebSphere Cast Iron development team, specializing in Connectors, Studio, and CDK. Rajath is a recipient of the prestigious Rookie Award at IBM. He holds a Bachelor degree of information Science at VTU, India.

Carla Sadtler is the leader of this project. She is a Consulting IT Specialist at the ITSO, Raleigh Center. She writes extensively about WebSphere products and solutions. Before joining the ITSO in 1985, Carla worked in the Raleigh branch office as a Program Support Representative, supporting IBM customers. She has a degree in Mathematics from the University of North Carolina at Greensboro.

Thanks to the following people for their contributions to this project:

Patrik Hysky
Dave Bennin
Shari Deanna
Debbie Willmschen
Margaret Ticknor
International Technical Support Organization, Raleigh and Poughkeepsie Centers

Reinhard Hohberger
IBM Germany

Katherine Sanders
Ken Bullough
IBM UK



Figure 1 From left to right: Nick Bletzer, Leonardo Rodriguez Leon, Lars Besselmann, Simon Dickerson, Roberto Mascarenhas, Giuliano Diniz de Moraes

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:
ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an email to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



Introduction and technical overview

IBM WebSphere Cast Iron Cloud Integration enables companies to integrate applications, regardless of whether the applications are located on-premise or in public or private clouds.

This chapter provides an overview of the core components and use of WebSphere Cast Iron Standard and Enterprise editions and includes a brief introduction to the Express edition. It covers the use of the following form factors:

- ▶ The virtual appliance, WebSphere Cast Iron Hypervisor Edition
- ▶ The physical appliance, WebSphere DataPower Cast Iron Appliance XH40
- ▶ The multi-tenant cloud service, WebSphere Cast Iron Live

1.1 Integrating cloud applications

Cloud computing has become a business evolution that is impacting all facets of business today, including sales, marketing, human capital management, supply chain, finance, manufacturing, quoting, ordering, service, support, email, and so on. As each of these functions move to the cloud they require migration and integration. Additionally, each vendor providing a cloud solution creates their own interface to the application. This fact creates a challenge in IT organizations in companies of all sizes and all locations globally as they attempt to understand and then manage these unique application interfaces and integrate applications from cloud to cloud and cloud to on-premise. It is this for these reasons that Cast Iron® is an invaluable solution to a modern IT infrastructure.

Cloud computing can provide companies with the following capabilities to meet business needs:

- ▶ Flexibility in working practices, such as working from home, the office, or with mobile computing, which creates the need to access data and files from anywhere at any time
- ▶ Easy implementation of applications that are offered as a software as a service (SaaS) and that can be configured and running in a matter of days
- ▶ Low cost solutions that can be paid for incrementally
- ▶ A reduction in the IT resource that is required for the maintenance of servers and infrastructure
- ▶ A reduction of impact from software upgrades and maintenance

However, the adoption of SaaS applications can mean that additional silos of information are created, which can lead to further disparate sources of information. These disparate sources create a challenge in that there is a hybrid architecture that includes on-premise solutions and the new SaaS applications. Integrating these applications and data can be key to the efficiency of the business. Integration can ensure that business users have access to the information that they require with the least possible friction, thus maximizing productivity and improving customer interaction.

Integration can be beneficial in the following situations:

- ▶ Migrating data to new SaaS applications
- ▶ Providing a single view of customer information in cloud and on-premise applications, for example a cloud-based CRM system and an on-premise Enterprise Resource Planning (ERP) system
- ▶ Using information from cloud marketing automation in e-commerce solutions
- ▶ Providing connectivity between private cloud systems and business partner solutions

An integration solution must bridge the gap between the on-premise existing systems and new cloud applications, platform, and infrastructure, providing a rapid and easy-to-use method of setting up the integrations.

WebSphere Cast Iron provides a solution that meets the challenge of integrating cloud applications with on-premise systems, cloud applications-to-cloud applications, and on-premise to on-premise applications, as illustrated in Figure 1-1 on page 3. The WebSphere Cast Iron environment focuses on the business requirements, the applications, and the business user requirements. It removes the daunting complexity of integration.

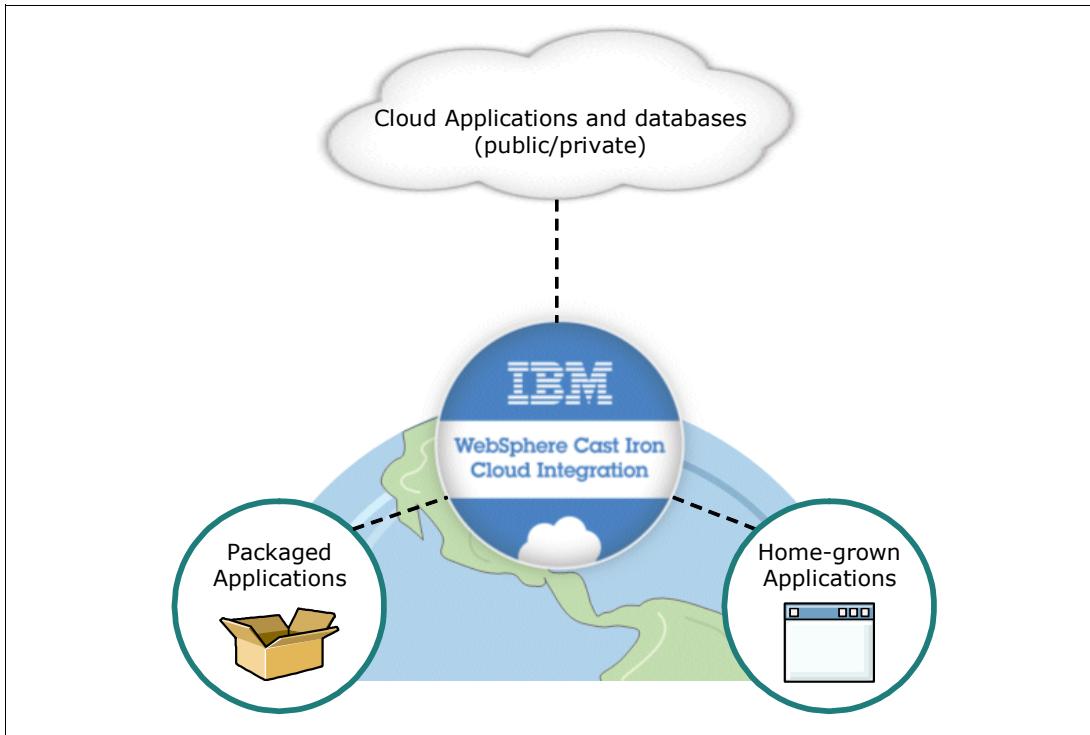


Figure 1-1 Cast Iron integration

1.2 Example use cases

WebSphere Cast Iron was beneficial in the following typical use cases:

- ▶ Exposing ERP information to sales agents

A company needed to make sales order information that was stored in multiple ERP systems available to sales agents who used a SaaS-based CRM system. Integrating the ERP information with the CRM system provides the agents visibility to the necessary information and avoids the agents having to sign on to multiple systems. The company had only a small IT department, with no substantial development capability, so they need a tool that business analysts can use.

The initial project involved using WebSphere Cast Iron to create three orchestrations that took less than two weeks to develop. The orchestrations provided real-time integration of the ERP information into the CRM system. Over time, the integration capabilities were expanded. There are now over 150 orchestrations connecting a multitude of systems that are running both in batch and on request.

- ▶ Integrating separate insurance policy systems

An insurance company has multiple systems that are used to manage new business. Having one system that provides quotes for policies and another system that is used to administer policies occasionally resulted in duplicate data entry and as a result, generated error conditions. In addition, many of the workflows involved in bringing in new business are manual, and vendors must provide documents in multiple formats.

The company used WebSphere Cast Iron to provide bidirectional integration between the two systems. Policy quote information is integrated into the administrative system and vice versa, making it necessary to enter data only one time and ensuring the integrity of the data in both systems. Integrating data through key business systems ensured that files

were routed correctly and that workflow was not interrupted if a user is unavailable. Vendors must provide data only in one format with the integration solution making the data available to the systems where it is needed in the proper format. Finally, the integration solution eliminates errors that were introduced because of duplicate data entry.

- ▶ Synchronizing e-commerce systems and linking to the SaaS CRM application to provide a single view of customers

A commercial enterprise has two e-commerce systems that are populated with customer and order data that is currently not synchronized. The enterprise also has a CRM application where customer data is kept and a fulfilment system with inventory and product data. The enterprise used WebSphere Cast Iron to integrate the data in these systems to provide a more cohesive view of customer data.

The initial implementation included the following orchestrations that provided the initial population of data throughout the systems and the synchronization of information between systems:

- Data providing a single view of the customer (SVC) was populated into the SaaS CRM system from the enterprise's existing systems using a web services integration.
- An orchestration with a short polling interval pushes account, order, inventory, and product information to the SVC from the e-commerce and fulfilment systems, ensuring that the customer service staff has the most current information.
- A nightly refresh of all stock keeping units (SKUs) in the e-commerce system occurs based on information from the fulfillment system. The orchestration uses secure file transfer to perform the refresh.
- Synchronization of customer records between the two e-commerce systems occurs on a regular basis, ensuring that when customers sign in to a system their data is available to them.

1.3 The WebSphere Cast Iron approach to integrating applications

WebSphere Cast Iron provides an approach to integrating applications that does not require any programming knowledge. You can build integration flows in WebSphere Cast Iron Studio (referred to in this book as *Studio*), which is a graphical development environment that is installed to a personal computer (PC). With Studio, you create an integration *project* that contains one or more *orchestrations*. Each orchestration is built with a number of *activities* that define the flow of data. You can define the details of an activity from the configuration panes within Studio.

A project contains all of the assets that are required for the orchestrations to run, including any file schemas, WSDL files, and functions. The project also defines connectivity to the sources of data, the *endpoints*. Cast Iron contains many built-in connectors to applications (for example SAP), databases, and web services that make connecting to these endpoints straightforward.

Figure 1-2 outlines the Cast Iron approach. Within Studio, there is a simulated runtime environment to enable unit testing of the orchestrations with visibility of all data at all points. After you test the project, you can publish the project to the run time, where you can then manage and monitor the project. A browser-based interface, the *Web Management Console (WMC)*, provides the capability to manage all functions on the run time.

Cast Iron Live: Cast Iron Live provides the same Studio development environment and runtime functionality as a physical or virtual Integration Appliance but through a cloud-based service. In Cast Iron Live, you publish a project to an environment, for example development, test, and production. The environment contains the functions of an on-premise Integration Appliance.

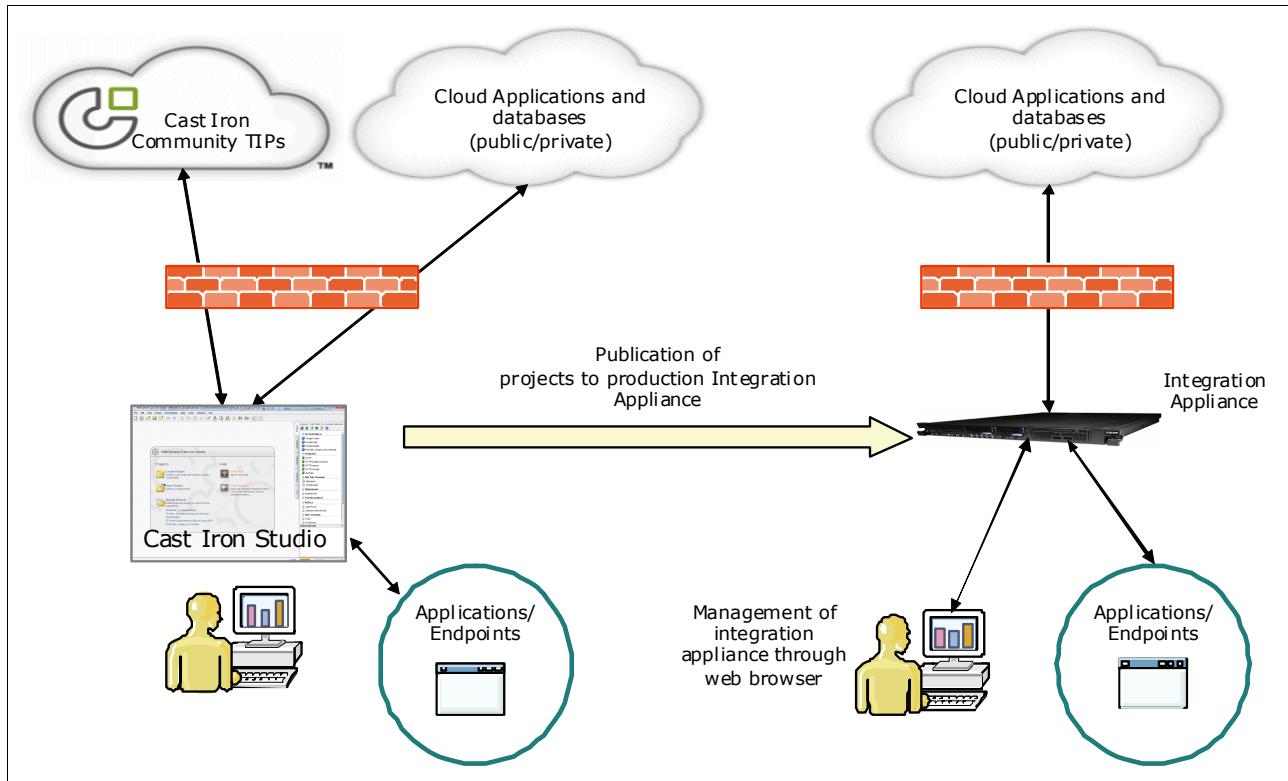


Figure 1-2 Cast Iron approach to developing integrations

As with any tool, there are preferred practices to implement designs. When processing large amounts of data, a poor design can mean that the integration takes an extraordinary amount of time or that it simply does not work. Each endpoint is different in its requirements, and any integration flow must use the characteristics of the endpoint to its best advantage, for example processing batches of data to optimize the speed of transactions.

Cast Iron also provides *Template Integration Projects (TIPs)* that encapsulate a specific integration use case between specific endpoints and that include preferred practices. You can download these TIPs from the Cast Iron community and modify to fit your precise needs.

Cast Iron Express uses a unique approach to the design and build of the integration workflows and is introduced in 1.11, “Express edition” on page 23.

1.4 Architecture overview

Cast Iron provides the following models of implementation:

- ▶ An appliance model where the appliance can be either physical hardware or a virtual machine and where the Integration Appliance is installed on-premise, behind the firewall
- ▶ A SaaS model (Cast Iron Live)

Each of the three form factors (physical, virtual, or Cast Iron Live) allows for cloud-to-cloud, cloud to on premise and on premise to on premise integration. Each form factor allows for real time, near real time and batch integrations.

This section introduces the components of a Cast Iron implementation, differentiating between the on-premise Integration Appliance and the cloud model (Cast Iron Live).

1.4.1 On-premise Integration Appliance

An on-premise appliance normally resides behind the firewall and not within a DMZ. For the development, test, and production life cycles, the runtime environments are typically separated and each has its own Integration Appliance that accesses the endpoints that are necessary for that environment. Figure 1-2 on page 5 outlines the approach for using an on-premise Integration Appliance.

You have full control of projects and their orchestrations on the appliance through the WMC. The orchestrations are started through a mechanism that is defined by an activity in the orchestration. These activities include scheduling, polling, or waiting for an incoming request, such as an HTTP Receive Request. Each orchestration must include at least one of these *starter activities*.

The Integration Appliance requires connectivity to the endpoints that are required by the orchestration. Data flows through the Integration Appliance and is stored internally as XML variables. The Integration Appliance allows you to control the logging for each orchestration.

1.4.2 Cast Iron Live

The multi-tenant Cast Iron Live service includes the following key components that allow you to design, run, and manage integrations, all in the cloud:

- ▶ A clustered runtime engine that runs the integrations and that has built-in fault-tolerance and recovery mechanisms
- ▶ A multi-tenant highly-available system to store the designed integrations
- ▶ A load-balancer to intelligently manage the loads throughout the various runtime engines
- ▶ Highly-available file-systems to store and manage logs that are related to the integrations

The design environment, referred to in Cast Iron Live as *Designer*, provides the same design capabilities and user interface as Studio does for the on-premise Integration Appliance.

You can access the Cast Iron Live interface from a web browser, as shown in Figure 1-3.



Figure 1-3 Cast Iron Live tabs for life cycle environments

You access the runtime environments through the Development, Production, and Test tabs. You can create projects in the Create tab and later change projects using the Modify tab.

The Modify tab, Figure 1-4, shows all projects that you are allowed to view.

The screenshot shows the 'WebSphere Cast Iron Live' interface. The top navigation bar includes links for Home, Setup, Create, Modify (which is highlighted in blue), Development, Production, and Test. On the right side of the header, there are links for user1@CILive.User.ibm, Settings, Help, and Log Out. Below the header is a table titled 'Projects'. The table has columns for Project, Created, Created By, Modified, Modified By, and Actions. Two projects are listed: 'SugarCRM_QueryTest' and 'SugarCRM_QueryTest (1.1)'. Each project row contains four action icons in the 'Actions' column: a pencil icon for Edit, a green checkmark icon for Publish, a red X icon for Delete, and a refresh/circular arrow icon for Create New Version.

Project	Created	Created By	Modified	Modified By	Actions
SugarCRM_QueryTest	09/30/2011 02:56:34 PM	user1@CILive.User.ibm	09/30/2011 03:46:09 PM	user1@CILive.User.ibm	
SugarCRM_QueryTest (1.1)	09/30/2011 03:49:32 PM	user1@CILive.User.ibm	09/30/2011 03:49:32 PM	user1@CILive.User.ibm	

Figure 1-4 Cast Iron Live Modify tab showing a list of projects

Each project has the following actions, as shown in Figure 1-5, which are initiated by clicking an icon:

- ▶ Edit Project ()
- ▶ Publish ()
- ▶ Delete ()
- ▶ Create New Version()

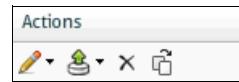


Figure 1-5 Cast Iron Live project Actions icons

To edit a project, select **Edit Project**, and then select **Edit Project in Designer**. The Studio development environment opens with the same functionality that the on-premise installation has. You can then publish the project to one of the runtime environments by selecting **Publish**.

When using Studio in Cast Iron Live, development is the same as for an installation on the local PC. Studio in Cast Iron Live has access to the same endpoints that the PC has access to. When publishing to a runtime environment, the orchestration is run in the Cast Iron Live cloud.

Figure 1-6 on page 8 illustrates the Cast Iron Live process. A developer accesses Cast Iron Live through a web browser, creates projects, and develops these projects with Studio (which runs in a Java runtime environment on the PC). When complete, the developer publishes the project to a test environment. After testing, the projects are published to a production

environment. You can also manage the environment, the projects, logs, and other components through a web browser. If you connect to applications or endpoints behind a firewall, the Secure Connector is required, as described in the following section.

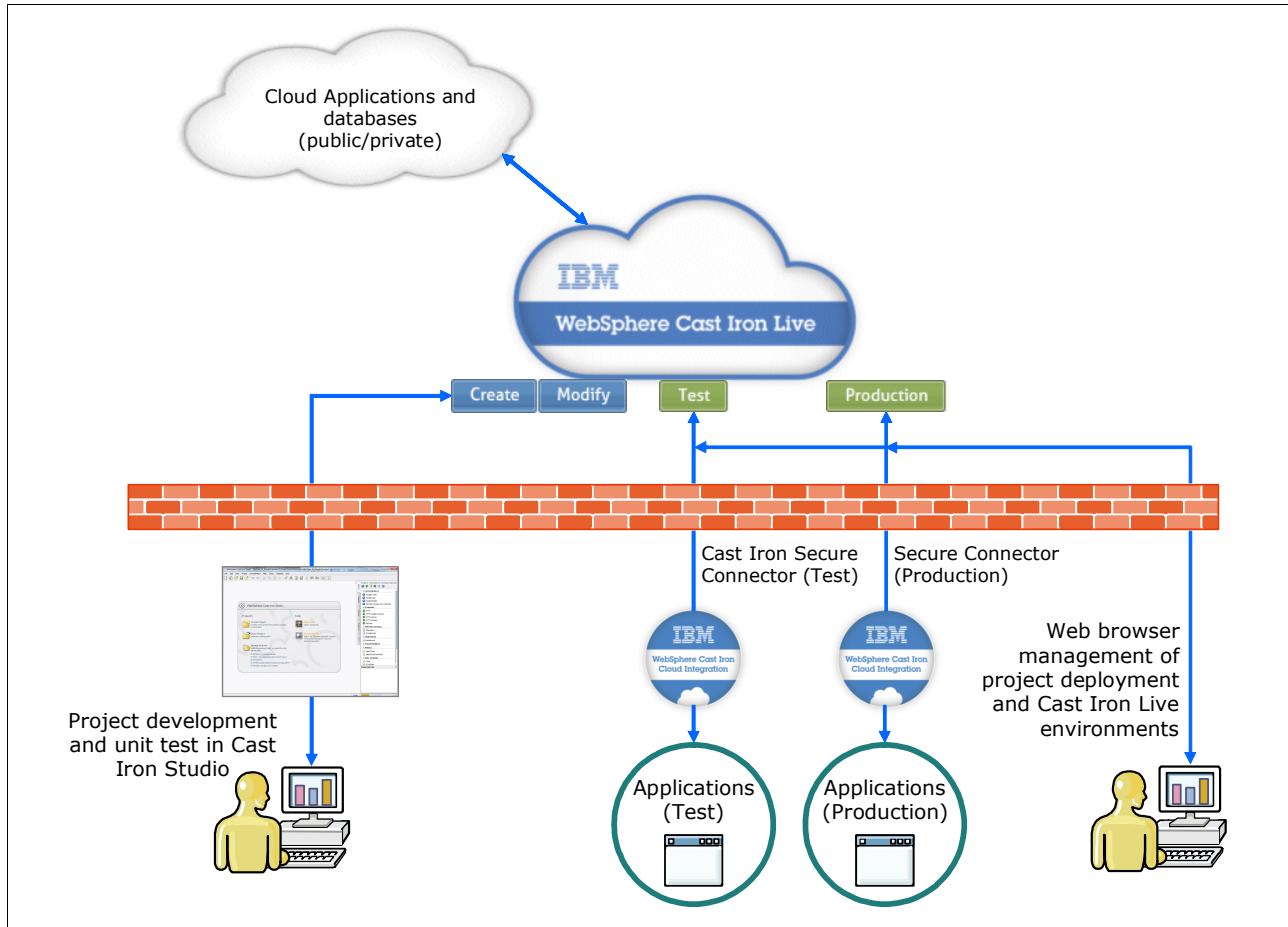


Figure 1-6 Cast Iron Live architecture

Secure Connector

The Cast Iron Live Secure Connector allows orchestrations in Cast Iron Live to connect to endpoints behind a firewall. A separate Secure Connector is configured for each environment and is installed to a computer that is running the Windows or Linux operating systems behind the firewall.

The Secure Connector includes the following security features:

- ▶ The Secure Connector always initiates communication with Cast Iron Live and that communication is validated before the Secure Connector attempts further processing.
- ▶ Communication between the Secure Connector and Cast Iron Live is based on the standard SSL 128-bit encryption over HTTPS through port 443. When the Secure Connector starts, it undergoes the SSL/TLS handshake, authenticates through standard X.509 certificates, and establishes a TLS-encrypted tunnel if all connections are successful.
- ▶ When the TLS connection is established successfully, the Secure Connector sends a request to Cast Iron Live for authentication. Based on information that is provided by the Secure Connector, including a private key, Cast Iron Live ensures that only the correct Secure Connector is granted access to a particular environment of a tenant.

- ▶ Data transmission and requests that are inbound to the Secure Connector from Cast Iron Live are limited to the available set of endpoint connectors that are provided by Cast Iron. Users must explicitly specify address and authentication information from each endpoint to which they are connecting.
- ▶ Local scripts or executables cannot be run directly through the Secure Connector.
- ▶ The Cast Iron Secure Connector can have the network or IPs, that it can access internally, restricted.

Further security information is available in the IBM white paper, *WebSphere Cast Iron Live Security*, which you can download from:

<http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=WH&htmlfid=WSW14139USEN>

1.5 Developing integrations

Integration workflows, or orchestrations, are contained within a project in Studio. When the project is complete, you publish it to the runtime environment.

Studio includes the following capabilities:

- ▶ The ability to design integrations from scratch using a drag-and-drop design palette
- ▶ A Template Integration Projects repository that allows you to reuse existing orchestrations
- ▶ Built-in connectivity to many cloud and on-premise applications
- ▶ Built-in transformation capabilities to improve data quality before the data reaches the target system
- ▶ Built-in functions to transform data from source format to a the format that is required by the target and to build relevant mapping rules
- ▶ The ability to build sophisticated workflows that include error-handling routines and re-try logic
- ▶ The ability to process messages in batch or real time
- ▶ Tools to unit test and verify whole or partial integration workflows before deploying them to the Integration Appliance
- ▶ One-click publishing of the completed integration workflows to the Integration Appliance
- ▶ A Connector Development Kit (CDK) to build connectors to custom applications that expose standards-based interfaces

The following Studio main components, illustrated in Figure 1-7 on page 10, enable the rapid development of integration projects:

- ▶ The main workspace where the activities are placed to build the orchestration (the top left pane in the figure)
- ▶ The configuration pane that contains the checklist and items that are configured for each activity (the bottom left pane in the figure)
- ▶ A tabulated section for assets of the project (the column to the right of the figure)

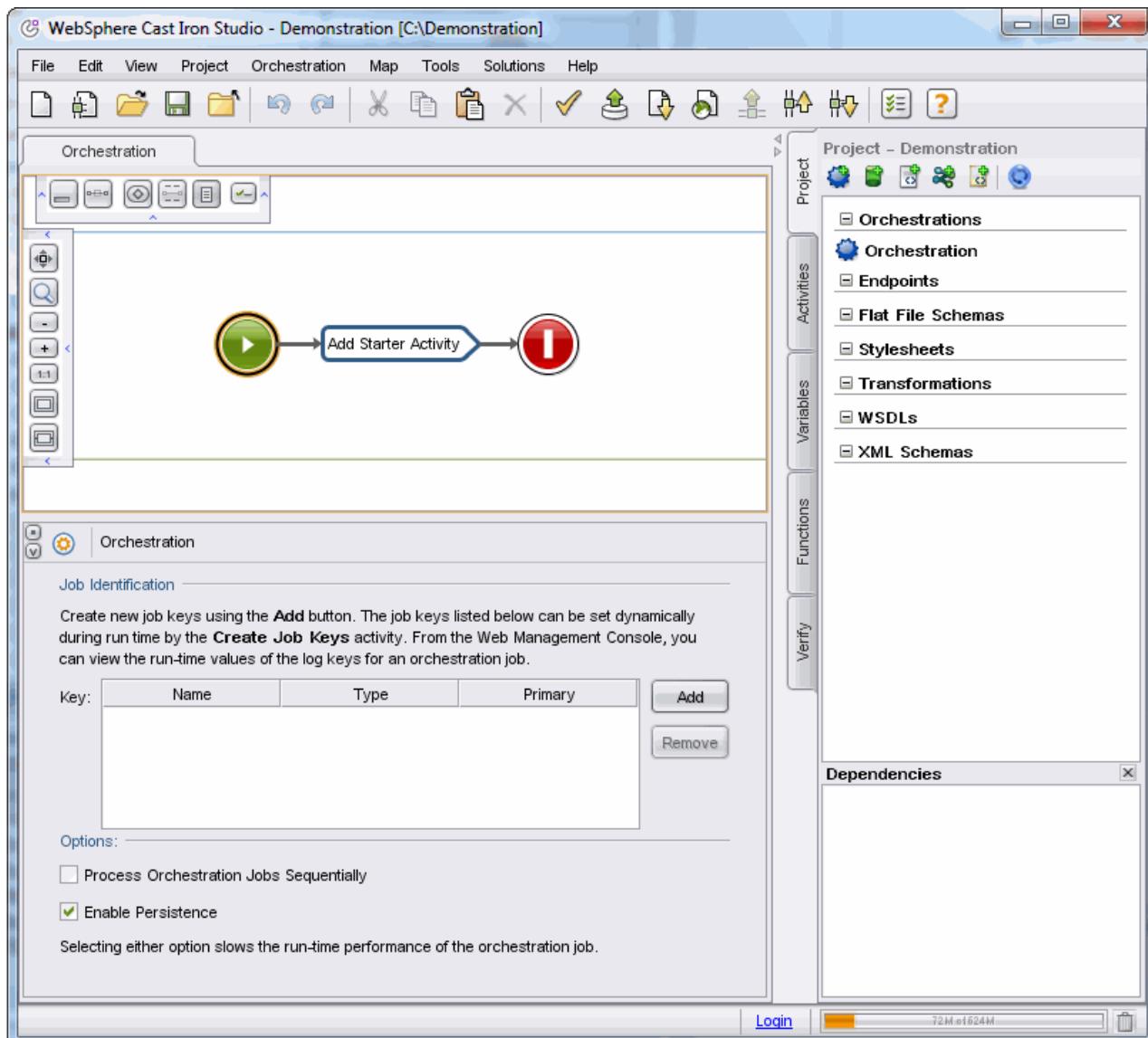


Figure 1-7 Cast Iron Studio showing a blank canvas

An orchestration is built by dragging activities onto the main workspace. Selecting an activity on the workspace opens the configuration pane for that activity. The configuration consists of a checklist of items that must be configured.

Chapter 3, “Developing and unit testing with WebSphere Cast Iron Studio” on page 79 provides more information about using Studio.

1.5.1 Connectors

Cast Iron contains built-in connectors to many applications. However, providing a connector for every application in the market is not practical. Companies also often build their own internal applications to which they want to connect. Although you can rely on generic connectivity through web services or can access data directly from the database to connect to applications, you can also use the Connector Development Kit (CDK) to develop a specific connector to an application.

The CDK is a wizard-driven tool that enables the development of a connector to abstract, specific functionality of the application, for example inserting new records or updating existing records. The CDK provides all the necessary tools to build the connector and also to perform unit testing, integration testing, and packaging and deploying the connector.

When new connectors are released by IBM Cast Iron, you can download them from Studio.

Built-in connectors: Studio built-in connectors have a discovery mechanism that lists objects, data fields, and parameters for the endpoint activity. For example, with the SAP Invoke RFC activity, you can browse the list of RFCs. When you select an RFC, the Cast Iron connector automatically retrieves the list of input parameters and the data fields that are returned.

1.5.2 Secure communications

When communicating with endpoints, such as applications, databases, and flat-files, Cast Iron can communicate using a variety of secure communication protocols:

- ▶ **HTTPS (HTTP over SSL):** Supports bilateral authentication, privacy, and integrity
- ▶ **Secure web services using SOAP/HTTP over SSL:** Supports bilateral authentication, privacy, and integrity
- ▶ **Secure FTP (FTP over SSH) and FTPS (FTP over SSL or Implicit FTPS):** Supports secure mechanisms for FTP server authentication, privacy, and integrity
- ▶ **Secure Databases (SSL):** Supports secure mechanism for database access

1.5.3 Data variables

Variables contain the data within the orchestration and are defined by a schema or data type. Mapping data for an activity uses variables, and new variables can be created as required. The Studio Variables tab, Figure 1-8 on page 12, shows all of the variables, their properties within the project, and the schema and activities where the variables are used. From this tab, you can jump to the activity where the variable is used.



Figure 1-8 Studio assets section showing Variables tab

1.6 Activities and configurations

You build a Cast Iron orchestration from a number of activities, which are listed by category in the Studio Activities tab, shown in Figure 1-9. Activities exist for all connectors, and there are activities for handling workflow logic, for example error handling with Try/Catch blocks and transformations. Some activities are defined as starter activities, as described in 1.6.2, “Defining a starter activity for an orchestration job” on page 16.

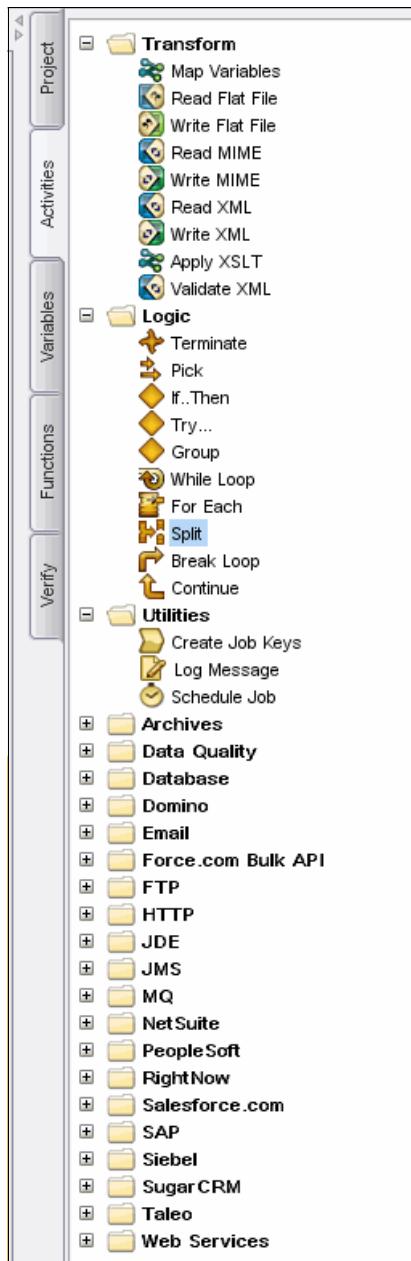


Figure 1-9 Studio Activities tab

You use an activity by dragging it to the workspace. After you place an activity on the workspace, you can move, delete, or reconfigure it at any time.

Figure 1-10 shows an orchestration in the workspace with the following activities:

- ▶ A Receive Request, which is the starter activity that receives a request from an HTTP endpoint
- ▶ A SAP activity, the Invoke RFC, which is the orchestration that the chosen RFC uses to retrieve data from SAP
- ▶ A Salesforce.com connector Upsert Objects, which performs the Salesforce.com function to update or insert a record to the selected Salesforce.com object
- ▶ A Send Response, which sends a response back to the HTTP requestor

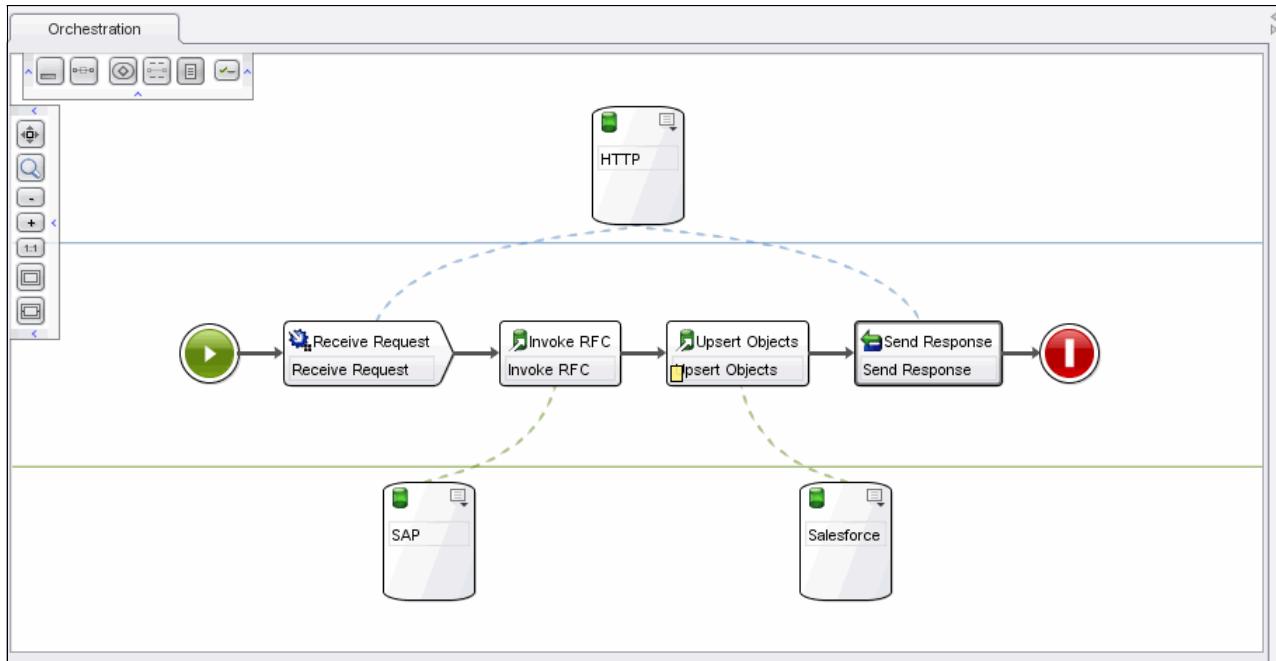


Figure 1-10 Studio canvas section showing initial development of an orchestration

When you select an activity in the workspace, Figure 1-11, the configuration pane is presented in the lower section of Studio. Activities for endpoints include a checklist of items and the input and output mapping requirements. Logic and transformation activities list items that are required for the configuration of that activity.

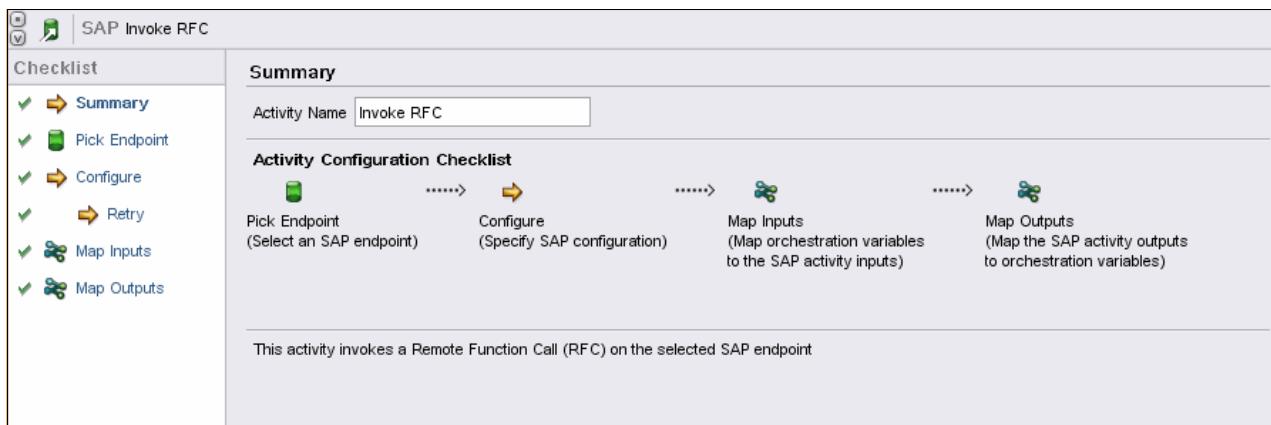


Figure 1-11 Studio configuration pane for an activity

You cannot verify or publish a project if there are incomplete configurations, so Studio provides the ability to validate the orchestrations before publishing them.

1.6.1 Transforming and mapping data

The activities for an endpoint include a checklist of items for mapping input and output values, except for those activities that only have either inputs or outputs, for example HTTP Receive Request.

In a mapping, the *map inputs* define data that is sent to the activity. The *map outputs* determine the data that can be used by other activities in the orchestration. Mapping is easy to configure using the configuration panes.

Defining map inputs

Use the Map Inputs option in the checklist to specify data that is sent to the activity:

1. Click **Select Inputs** in the From Orchestration section to select the variables that are to be used. You can select more than one input variable.
2. Map the data fields to the activity input in the To Activity section by dragging the activity.
3. A function is applied to a data field by dropping the function to the center section and mapping the input data to the function and the function to the activity.

Figure 1-12 shows an example of mapped inputs.

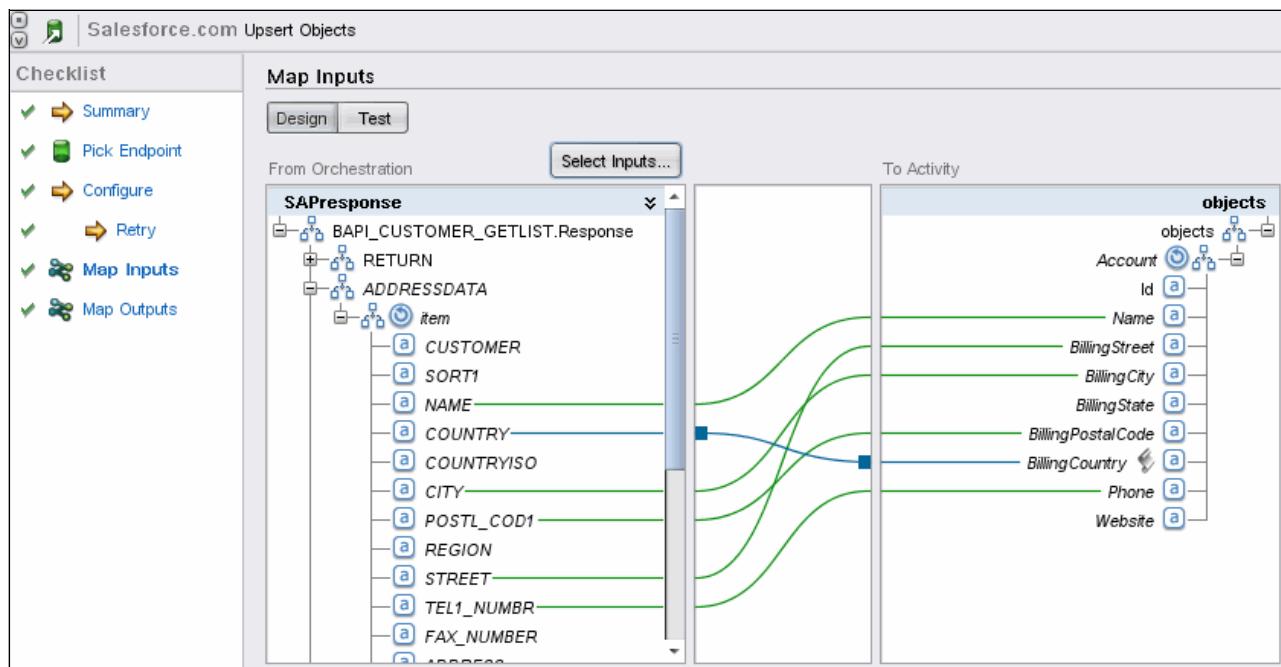


Figure 1-12 Studio example of mapped inputs

Map outputs

Use the Map Outputs option in the checklist to select the data that is required by other activities, for example, data that is required to be passed to an endpoint or return codes from an activity that need to be queried. One or more new variables are created for the data. Figure 1-13 shows an example of mapped outputs.

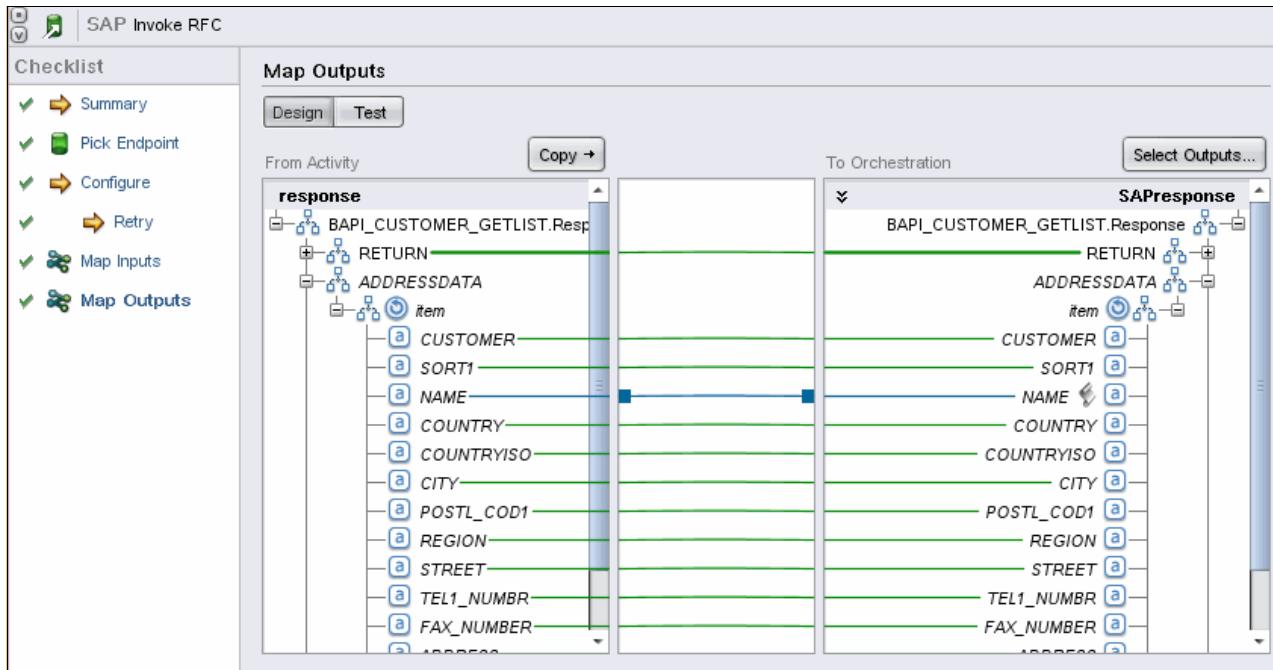


Figure 1-13 Studio example of mapped outputs

1.6.2 Defining a starter activity for an orchestration job

Orchestrations start according to the *starter activity* that is used. An orchestration must have at least one starter activity and can have more than one starter activity when a Pick activity is used. Starter activities are denoted in the Activities tab by a specific icon ().

When an orchestration starts, a new job is created on the appliance with a unique identifier. The WMC gives you full control over how an orchestration job is run and whether jobs can run in parallel.

Orchestrations can include the following starter activities:

- ▶ Schedule Job

A new orchestration job is created at each scheduled time and the date-time value is available in Map Outputs.

- ▶ HTML Receive Request

An HTTP request can be sent to the appliance to start the orchestration. A unique URL must be specified in the configuration for the activity, and the HTTP headers and body are available in the Map Outputs section.

- ▶ Provide Service

A web service request can be made to the appliance to start an orchestration job.

- ▶ Endpoint specific activities

These activities include receiving an IDOC from an SAP system and polling activities, such as for getting updated database records or updated records from Salesforce.com.

1.6.3 Error handling

Studio provides the following activities for handling errors:

- ▶ Catch blocks for the entire orchestration
- ▶ Try...Catch blocks containing other activities where multiple Catch blocks can be added
- ▶ If...Then logic for checking results from activities
- ▶ Retry connections for endpoints
- ▶ XML validation

In addition, you can set up Simple Network Management Protocol (SNMP) notifications on the Integration Appliance using the WMC.

1.6.4 Testing

Studio provides the Verify tool to unit test and verify that orchestrations are functioning as expected. An orchestration job can be started, and the orchestration then waits for the starter activity to initiate. The HTTP Post utility provides an easy way to test and verify the orchestration. Detailed logged information about how the data flows through the various steps in the orchestration is provided to help you easily identify problem areas.

Figure 1-14 shows a test of an orchestration in the Verify tab. Each activity is shown in the top panel, and you can view the data at any point for any activity. The data for the variable SAP Response is shown in the lower panel.

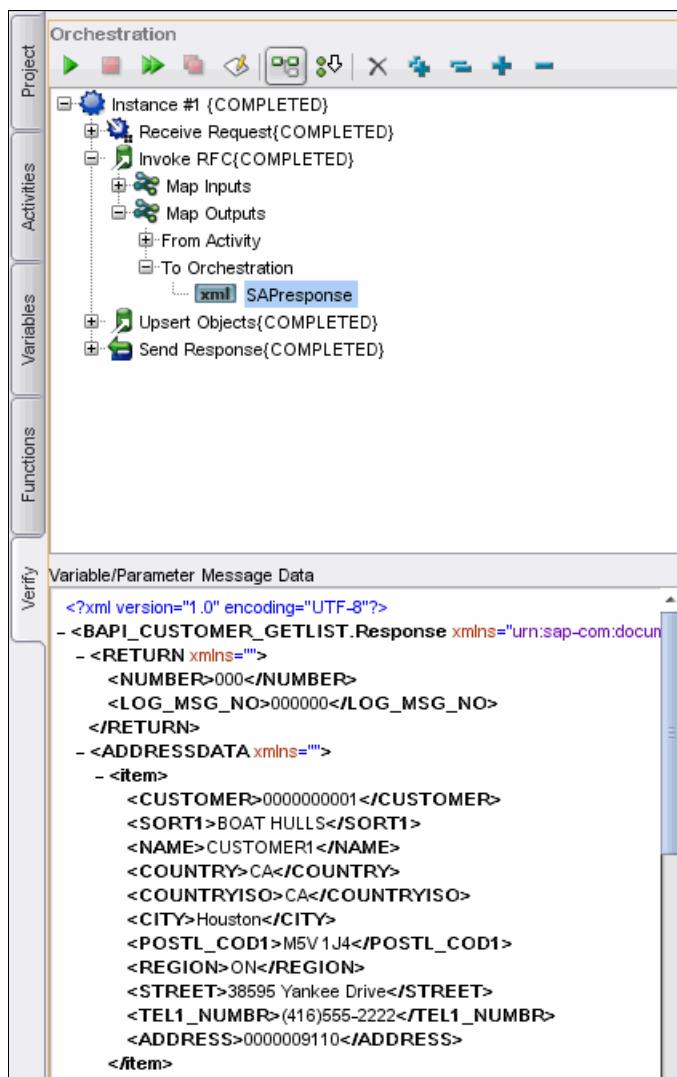


Figure 1-14 Example of Verify tab used for testing the orchestration

Individual activities can also be tested with the verify tool. Rather than running the entire orchestration, you can use a Verify Activity menu item.

You can test the mapping for both Map Inputs and Map Outputs in the configuration pane. This testing is useful for confirming any transformations that might occur in the map. Figure 1-15 shows a test of the map for the Salesforce.com Upsert activity. The ability to generate test data is provided.

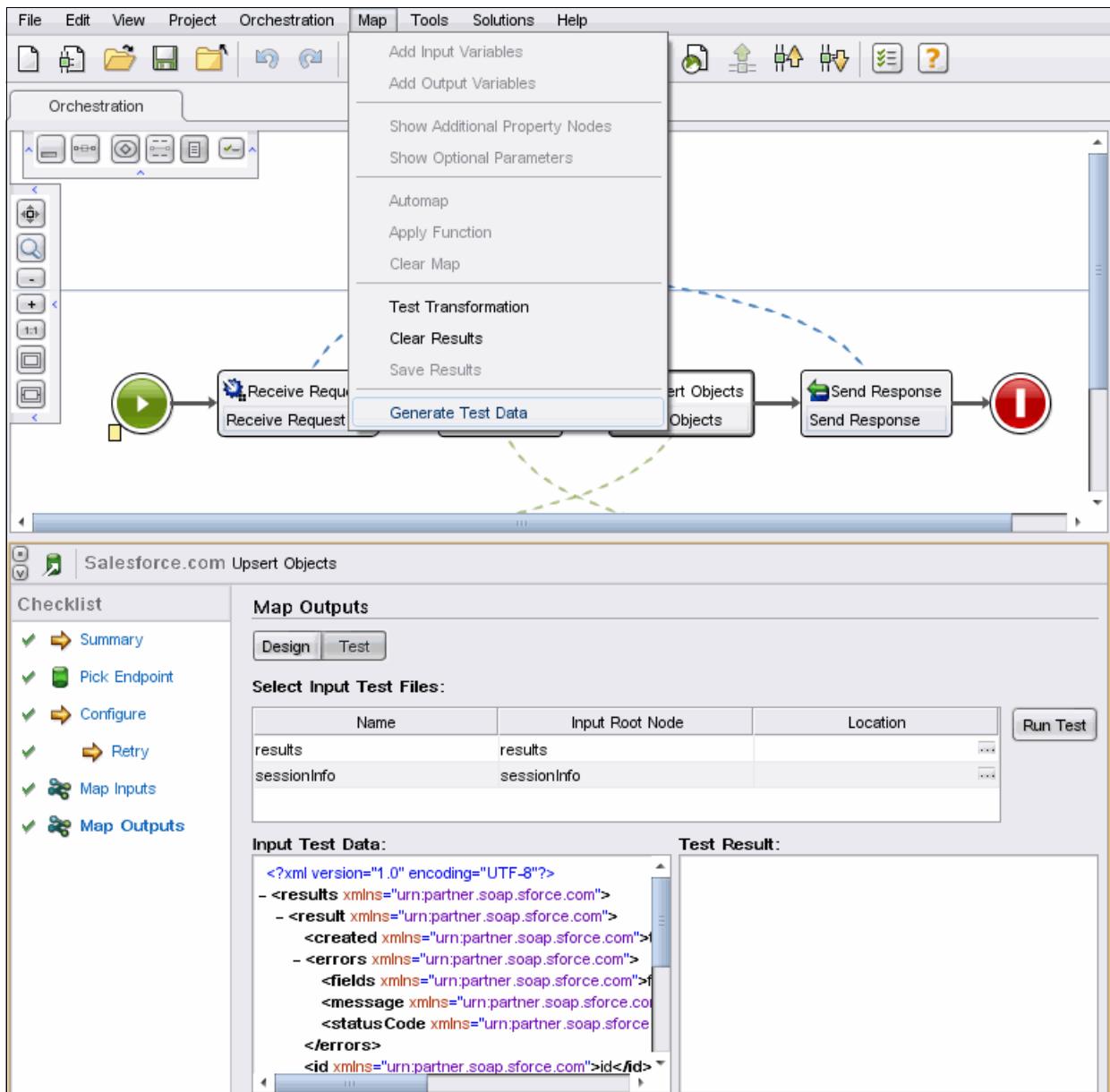


Figure 1-15 Testing the mapping for Map Output for an activity

1.6.5 Reuse: Template Integration Projects

Although Studio provides the capability to build sophisticated integrations from the beginning, Cast Iron also provides Template Integration Projects that encapsulate a specific integration use case between specific endpoints and include preferred practices. A common repository that includes all of the available TIPs that are published is available for download by any customer or IBM Business Partner. These TIPs are authored by internal or external integration experts who have expertise with the endpoints for which they authored the integrations.

Using the TIPs provides the following key benefits:

- ▶ Save costs because 80% of the developer effort to build an integration is already codified in the template. The vast majority of the remaining effort involves using a wizard to configure the endpoint and apply any special customizations, such as custom fields or objects mapping and special business logic.
- ▶ Readily referenceable preferred practices because each certified TIP is carefully authored and provides a method to reference and learn from other users and Cast Iron experts.
- ▶ Flexible to adapt to changes because the TIPs and TIP configuration wizard allow a user to modify an integration when business needs change and additional fields or new business rules must be accommodated. The wizard automatically provides options for a user to select custom fields, change maps, or change business rules.

1.6.6 Deploying projects

After testing the project, publish it to the runtime environment with a single Studio menu command (**File → Publish**). After publishing the project, click **Run Configuration** in the WMC to start it. The enabled orchestrations in the project run perpetually with the orchestration jobs created, according to the starter activities used.

Cast Iron also allows you to export projects from Studio. The exported project can then be imported into the run time (Cast Iron Live or an Integration Appliance) using the WMC. This technique is useful in situations where Studio and the Integration Appliance are on separate networks or when developers do not have the authority to publish directly to the runtime, but instead, export the project to a source control system where the administrator picks up the project for import into the runtime.

1.6.7 Updating projects

Integration rules and workflows are subject to changes as business evolves. Several capabilities are provided that allow administrators to manage and deploy changes to the Cast Iron environments. When you make changes to the orchestrations in Studio, you can easily save the orchestrations to a new version. The Cast Iron Live service also includes a one-click approach to creating a new version on the saved projects. In addition, you can clone published project configurations as a new version to keep a history of changes.

1.7 Management and monitoring

You can use the WMC dashboard to obtain an overview of the status of the Integration Appliance, as shown in Figure 1-16. Each project configuration is shown with a list of jobs, and an instant view of resource utilization is provided. You can drill down into each project configuration to view detailed logs, which are available for each job and for the system.

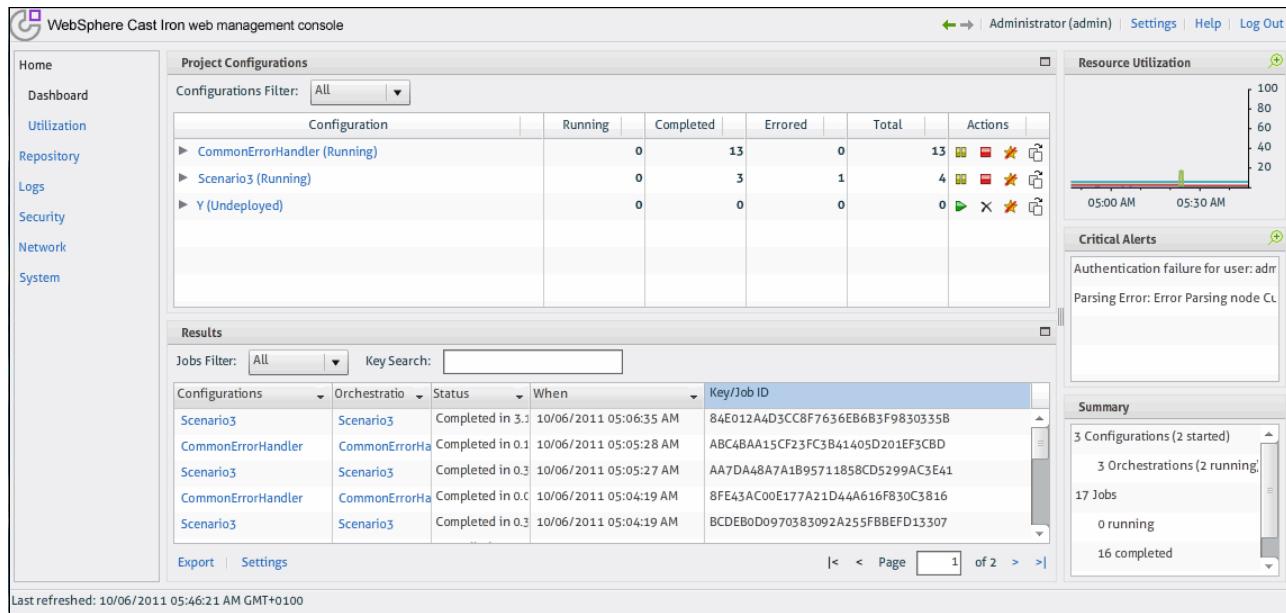


Figure 1-16 Web Management Console dashboard

Logging level settings, shown in Figure 1-17, allow for separate levels of error logging from none through to logging all details of each activity.

The screenshot shows the 'Edit Orchestration Settings' dialog. It contains two rows of settings. The first row applies to 'Name' and has columns for 'Enabled' (checkbox), 'Logging Level' (dropdown set to 'None'), 'Log Synchronously' (checkbox), and 'Max Simultaneous Jobs' (checkbox checked, value 'Unlimited'). The second row applies to 'Orchestration' and has columns for 'Enabled' (checkbox checked), 'Logging Level' (dropdown set to 'Error Values'), 'Log Synchronously' (checkbox), and 'Max Simultaneous Jobs' (checkbox checked, value '10').

Figure 1-17 Web Management Console orchestration settings

The dashboard also shows the resource utilization of the Integration Appliance, and you can drill down to more detail.

You can configure notifications. A management API is available that enables you to write scripts for all aspects of the Integration Appliance, including monitoring orchestrations, logging, and security.

Chapter 4, “Management and monitoring” on page 173 provides details about management and monitoring capabilities.

1.8 Availability and scaling

The Cast Iron architecture and design includes the following features for ensuring reliable and fast service and for providing for maximum availability:

- ▶ A high availability (HA) topology
- ▶ Parallel appliance capability
- ▶ Orchestration design to optimize efficiency
- ▶ Very low downtime for deployment of updates to projects
- ▶ Very low maintenance downtime for upgrades to the appliance

Chapter 6, “Availability and scalability planning” on page 277 provides further details about planning for availability and providing a reliable service.

1.9 Upgrading to a new release

New releases of Studio and the Integration Appliance are easily installed. A new version of Studio is installed to the PC and does not overwrite the currently installed version. Thus, you can open existing projects and save them with the new version.

For the Integration Appliance, an update file is released (.vcrypt2 for Cast Iron Hypervisor,.scrypt2 for the physical appliance). You upload this file to the Integration Appliance through the WMC.

IBM provides advance notice for any scheduled downtime during upgrades to Cast Iron Live. Not all upgrades require downtime.

1.10 Cast Iron editions

WebSphere Cast Iron is available in the following editions:

- ▶ WebSphere Cast Iron Standard Edition

This edition includes the base functionality for the development and deployment of orchestrations and connectivity to the SaaS and mid-market endpoints. The full mapping and function capabilities and the Connector Developer Kit (CDK) are included. The Standard Edition provides access to a repository of reusable templates to help you develop your integrations.

- ▶ WebSphere Cast Iron Enterprise Edition

This edition includes all the capability of the Standard Edition and adds data quality, enterprise connectivity (for example SAP), high availability (requires two appliances), and management APIs to the Integration Appliance. The Enterprise Edition provides access to the repository of reusable templates and gives you the ability to create new reusable templates.

- ▶ WebSphere Cast Iron Express Edition

This edition is a cloud-based offering that provides for integration between Salesforce.com and other data sources. Integrations can be created in minutes through a web browser. Full monitoring and administration is provided.

Standard and Enterprise editions are available on the following form factors:

- ▶ An appliance running within a virtual machine (WebSphere Cast Iron Hypervisor edition)

- ▶ A physical appliance (WebSphere DataPower Cast Iron Appliance XH40)
- ▶ The multi-tenant cloud offering Cast Iron Live

The functionality and orchestration development across these form factors are identical.

1.11 Express edition

Cast Iron Express Edition is a cloud solution that allows Salesforce.com administrators to configure, run, and manage integration projects between Salesforce.com and other data sources. The focus on simplicity allows you to configure connections and build maps between application endpoints. You can access, monitor, and administer all projects from a centralized management console.

You use Cast Iron Express Edition through a web browser. Figure 1-18 shows the Welcome window that opens when you first sign in to Cast Iron Express Edition. Full help with video tutorials is provided from this welcome window.

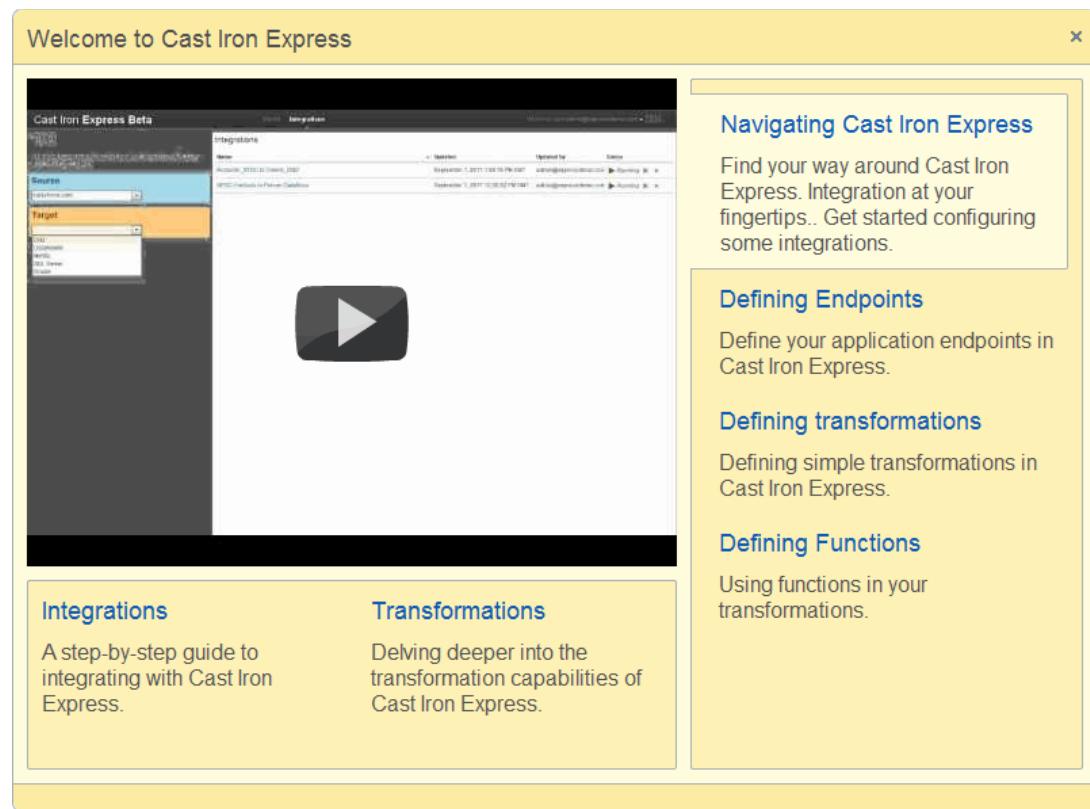


Figure 1-18 Express Edition welcome window

The user interface for Cast Iron Express Edition is different from the Cast Iron Live user interface. Figure 1-19 shows an example of the creation of the field mapping between endpoints.

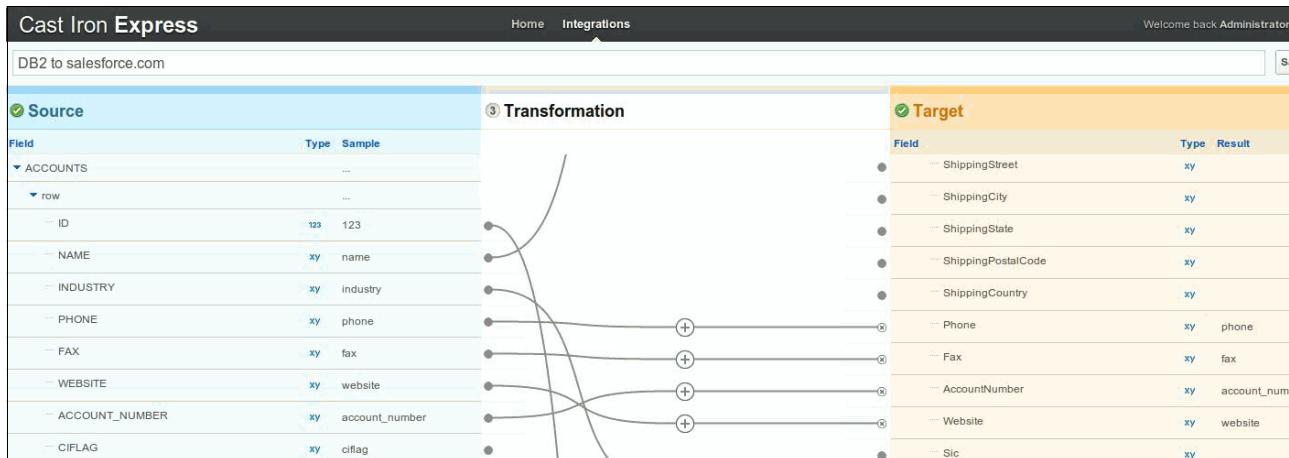


Figure 1-19 Example of Express Edition mapping capability

You can sign up for a trial of Cast Iron Express Edition at:

<http://express.castiron.com/express/>

This book does not provide further details about the Cast Iron Express Edition.

1.12 Terminology

When learning about Cast Iron, it might be helpful to have an understanding of the following terminology:

Activity	An activity performs a specific operation. Activities are organized in categories and are available on the Studio Activities tab.
Appliance	See <i>Integration Appliance</i> .
Checklist	A list of tasks that must be configured for the activity. The orchestration cannot be verified or published if one or more checklist tasks are incomplete.
Connector	The connector is the Cast Iron module that provides the integration to an endpoint. Use the activities for the endpoint found on the Studio Activities tab to use a connector.
Connector Developer Kit (CDK)	The CDK allows you to use Studio to develop and deploy new connectors.
Designer	The Designer provides the development environment for creating orchestrations. The term <i>Designer</i> is used in the Cast Iron Live menu and is also known as <i>Studio</i> .
Endpoint	The endpoint is an external system, for example an application, such as SAP or an FTP server. In Studio, an endpoint represents the properties of an external system.

Integration Appliance

The Integration Appliance is the Cast Iron device that provides the run time for integration projects that are developed with Studio.

Job	A job is an instance of an orchestration.
Orchestration	An orchestration is the specific sequence of processing activities (such as data mapping, transformation, and control logic activities), endpoints, and data types that are defined and configured using Studio and are deployed to a Cast Iron Runtime.
Project	A project is a Cast Iron container for managing orchestrations and the assets that the orchestrations require. You develop projects using Studio.
Run time	The run time is the component that runs the project configurations. In Cast Iron Live, run times are environments.
Secure Connector	The Cast Iron Live Secure Connector facilitates the secure transfer of data between Cast Iron cloud (Cast Iron Live and Express Edition) and an endpoint that is located behind a firewall.
Starter activity	The starter activity is an activity that can initiate an orchestration. An orchestration must have at least one starter activity. To have more than one starter activity, use the Pick activity.
Studio	Studio is the development environment for creating projects and orchestrations. In Cast Iron Live, Studio is also referred to as <i>Designer</i> .

Web Management Console (WMC)

The WMC is the browser interface to the Cast Iron Runtime and provides full functionality for you to manage and control all aspects of the runtime environment.



Installing and setting up WebSphere Cast Iron integration

WebSphere Cast Iron includes the following components:

- ▶ WebSphere Cast Iron Studio, a modelling tool
- ▶ The WebSphere Cast Iron runtime, which can be a physical appliance, a virtual appliance, or in the cloud (WebSphere Cast Iron Live)

With WebSphere Cast Iron Live, WebSphere Cast Iron Studio and the WebSphere Cast Iron Runtime are combined in the cloud and are managed through a web browser. Physical appliances are appliances that are based on DataPower hardware and that are installed in a rack. Virtual appliances are hypervisor editions that run on a hypervisor and that are packaged with a guest operating system and optional application software. The term *Integration Appliance* refers to both the virtual appliance and the physical appliance run times.

This chapter describes the following procedures to set up and configure the components of WebSphere Cast Iron Cloud Integration:

- ▶ Installing and configuring WebSphere Cast Iron Studio
- ▶ Installing and configuring the physical appliance
- ▶ Configuring the physical appliance for high availability
- ▶ Installing and configuring the virtual appliance
- ▶ Finalizing the installation using the WMC
- ▶ Installing the WebSphere Cast Iron Live Secure Connector

Version note: The examples included in this chapter use WebSphere Cast Iron Cloud Integration Version 6.1.

This chapter also explains how to upgrade from WebSphere Cast Iron Cloud Integration V6.0 to V6.1.

2.1 Installing and configuring WebSphere Cast Iron Studio

WebSphere Cast Iron Studio (referred to as *Studio*) is shipped with the Integration Appliance. Before you begin the installation, be sure that your system meets the prerequisites described at:

<http://www-01.ibm.com/software/integration/cast-iron-cloud-integration/reqs/>

Click the appropriate link to determine the prerequisites for your Integration Appliance, Studio, and for your web browser.

Updating Studio: Currently, the application of a fix pack for Studio is a full installation. Therefore, check the support page for the latest version before starting the installation:

<http://www-947.ibm.com/support/entry/portal/Overview>

Your version of Studio must match the version of your appliance. If you plan to upgrade Studio, plan to upgrade the appliance to match.

2.1.1 Installing Studio

To install Studio on a Windows operating system:

1. Run the 6.1.0.1-WS-WCI-20110916-1648_H2.exe Studio installation file.
2. At the Welcome panel, shown in Figure 2-1, click **Next**.



Figure 2-1 Studio installation Welcome panel

3. Read and accept the license agreement, shown in Figure 2-2, and then click **Next**.

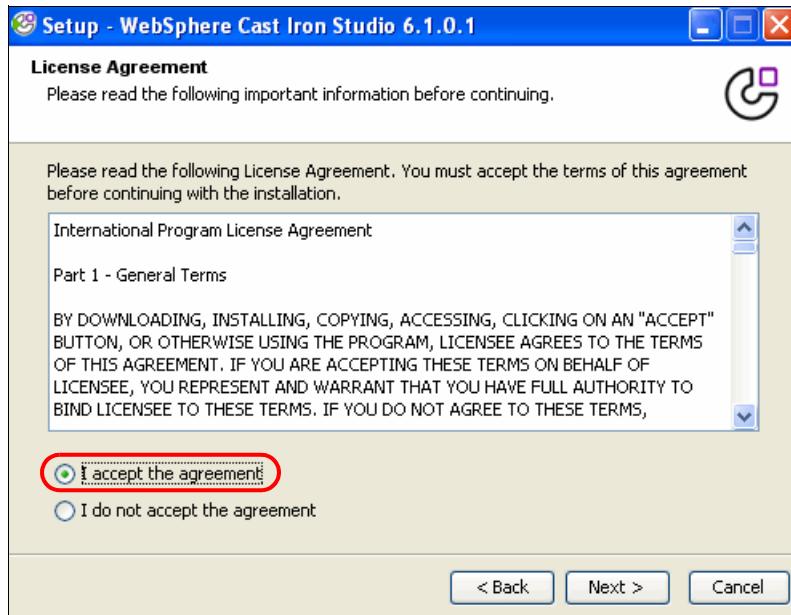


Figure 2-2 Studio license agreement

4. At the Destination Directory panel, specify the installation path. The default path is the C:\Program Files\IBM\WebSphere Cast Iron Studio 6.1.0.1 directory. This example uses the reduced path of the C:\IBM\WebSphereCastIronStudio6.1.0.1 directory, as shown in Figure 2-3. Click **Next** to continue.

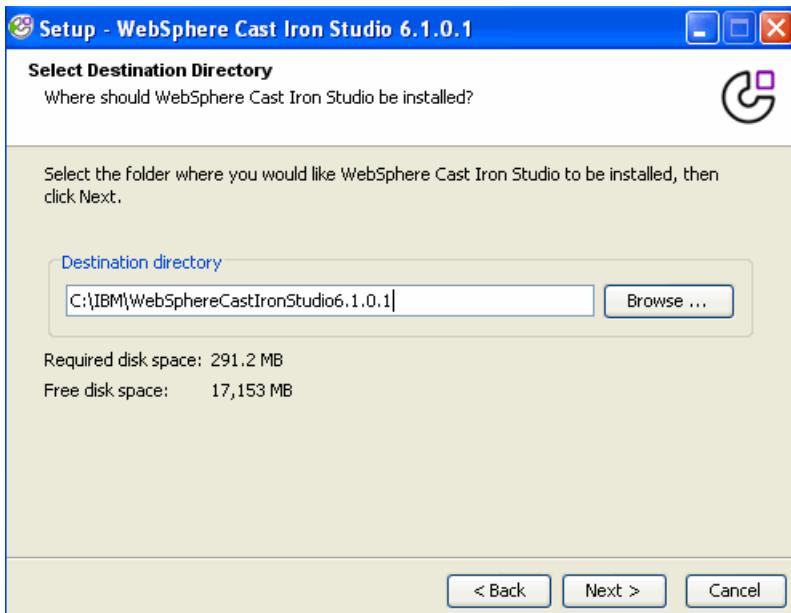


Figure 2-3 Studio installation path

5. Accept the default options for the Start Menu Folder, as shown in Figure 2-4, and then click **Next**.

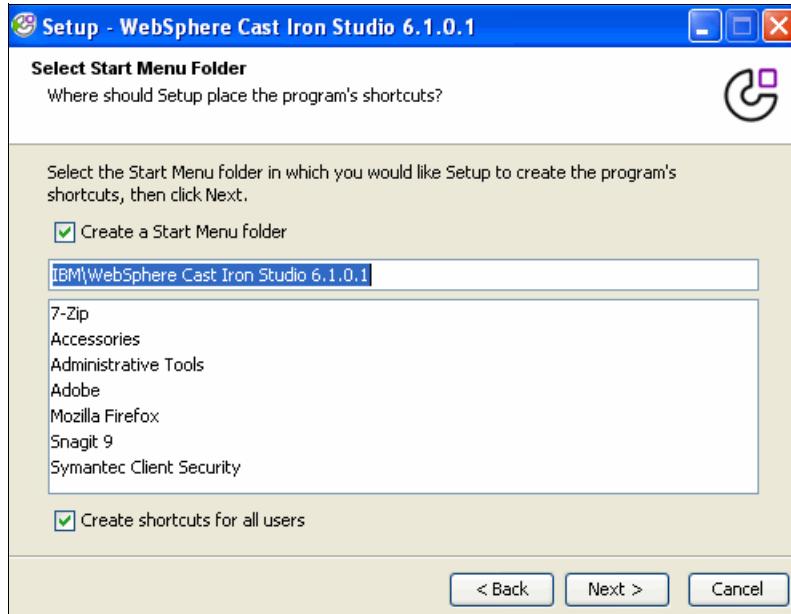


Figure 2-4 Studio installation Start menu

6. Accept the default options for File Associations, Figure 2-5, and then click **Next**.

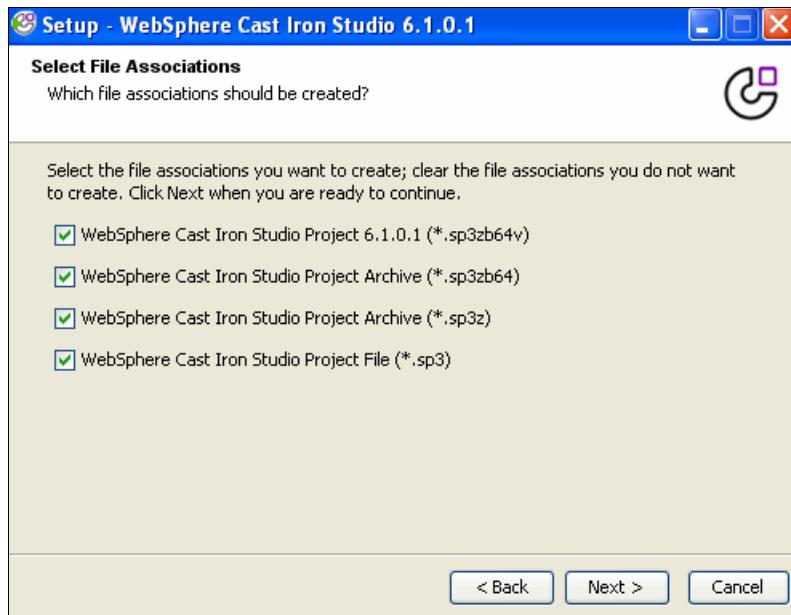


Figure 2-5 Studio installation File Associations

7. Accept the default options for Additional Tasks, as shown Figure 2-6. Click **Next**.

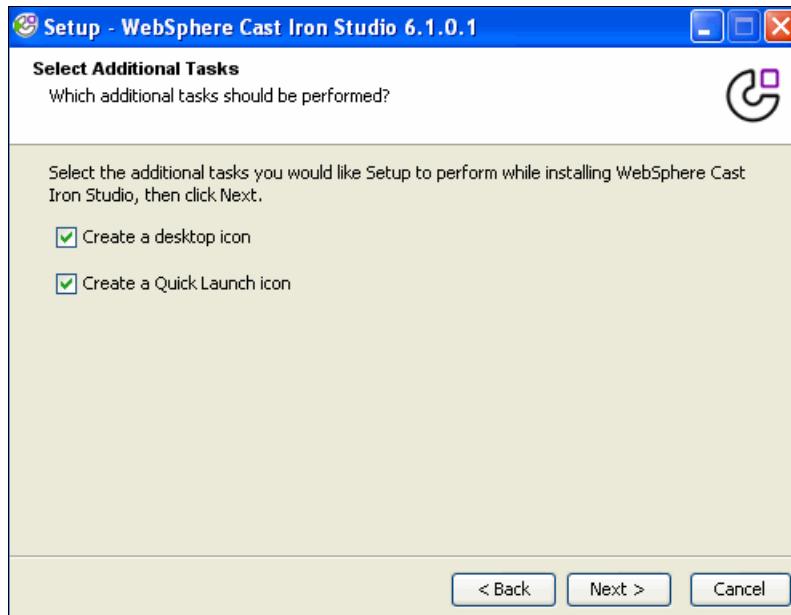


Figure 2-6 Studio installation Additional Tasks

A progress bar shows the status of the installation, as shown in Figure 2-7. The installation takes only a few minutes.

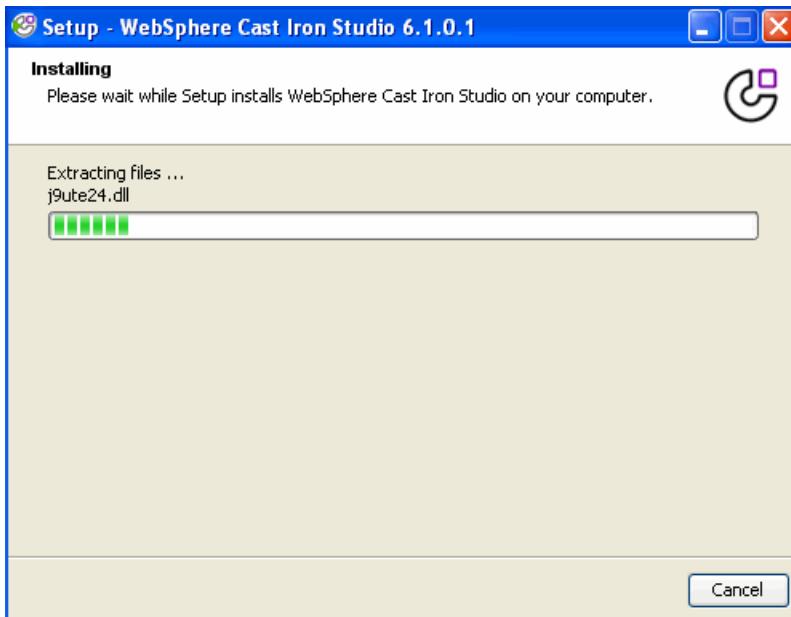


Figure 2-7 Studio installation progress bar

- When finished, the final panel opens. Leave the Run WebSphere Cast Iron Studio option enabled, and click **Finish**, as shown in Figure 2-8.



Figure 2-8 Studio installation final window

- Depending on your operating system and the security settings, a Windows Security Alert message might display, as shown in Figure 2-9. Click **Unblock** to allow Studio to start.



Figure 2-9 Windows Security Alert

10. The entry window of Studio displays, as shown in Figure 2-10. To set the language for the user interface, Studio determines the regional setting of the operating system. If, for example, the regional setting of the operating system is set to U.S., the Studio menus display in English.

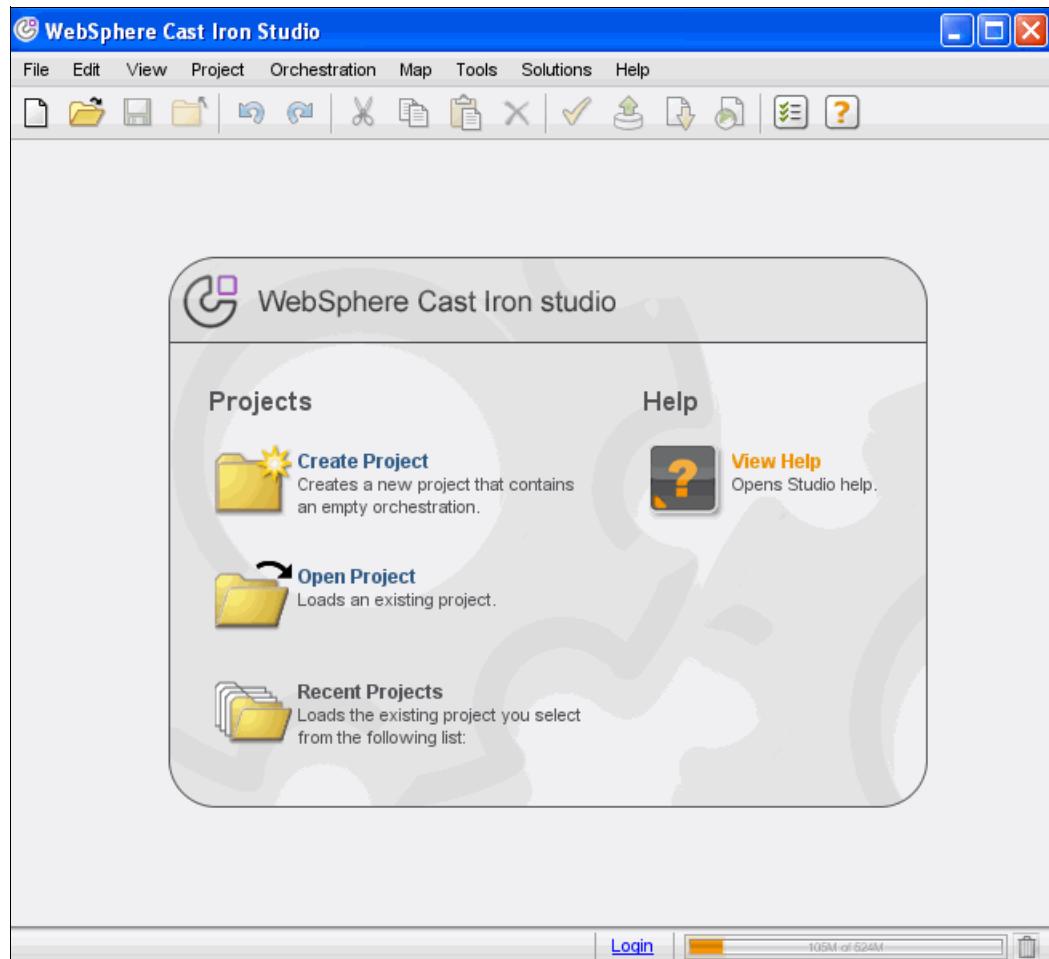


Figure 2-10 Studio panel

11. Click **Help** → **About** to determine the version of Studio.

Installation directories: Studio creates the following directories during installation and startup:

- ▶ The C:\IBM\WebSphereCastIronStudio6.1.0.1 installation directory contains the binaries and the output.log and error.log log files.
- ▶ The C:\Documents and Settings\username\.castiron configuration directory, where *username* is the administrator or the name of the user installing Studio, contains configuration properties, installed connectors libraries, and other configuration properties.

On Windows 7, the directory is C:\Users\username\.castiron

2.1.2 Changing the user interface language

Studio supports languages other than English. You can change the language settings by changing the regional settings. However, a more convenient and appropriate way to specify the language of choice is to add one of the following parameters to the startup command:

- ▶ **-J-Duser.language** *language_code*
- ▶ **-J-Duser.country** *country_code*

The language code conforms to the ISO-639 standard and the country codes conform to the ISO 3166 standard. The language code is usually entered first because the country code is used to specify a particular variant of a language. You can specify the language without the country code.

Add the parameters to the Studio start command, for example to use Brazilian Portuguese, enter the following command:

```
C:\IBM\WebSphereCastIronStudio6.1.0.1\CastIronStudio.exe  
-J-Duser.language=pt -J-Duser.country=BR
```

You can add these parameters to the shortcut that was created during the Studio installation, as shown in Figure 2-11.



Figure 2-11 Shortcut with language settings

Studio will now start with a German interface, as shown in Figure 2-12.

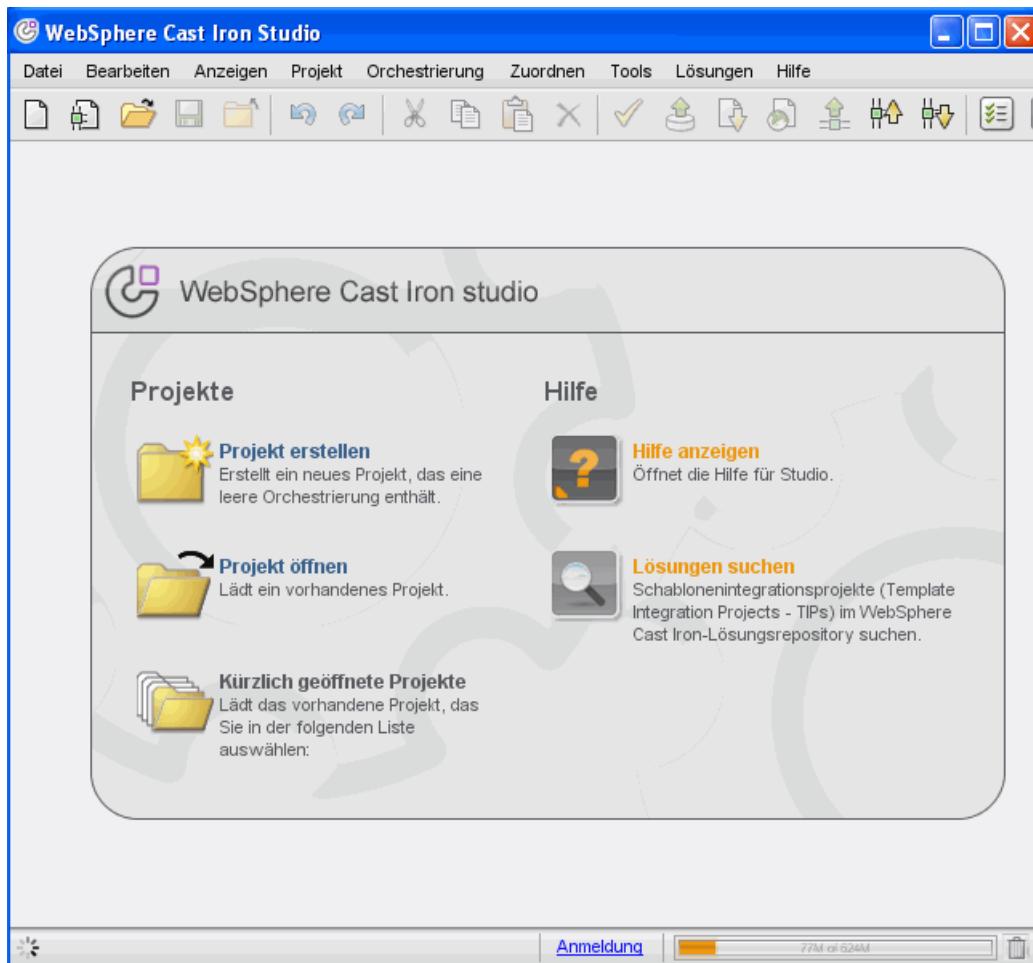


Figure 2-12 Studio with German

Table 2-1 provides a list of some of the supported languages.

Table 2-1 Language codes

Language	Language code	country code
English	EN	en
French	FR	fr
German	DE	de
Italian	IT	it
Japanese	JP	ja
Spanish	ES	es

The examples in this book use the English setting.

2.1.3 Installing module providers

Studio ships with several built-in connectors. For some of these connectors, Studio requires library files (.jar and .dll) to be imported. Database Connector, JDE, JMS, MQ, PeopleSoft, and SAP are some of the connectors that might require additional libraries. These files must be present in Studio before a connector can be used with the specific endpoint. The libraries can be retrieved from the supplier of the application.

The scenario that is described in Chapter 10, “Scenario: Bidirectional account synchronization” on page 357 connects to an SAP system. This connectivity requires that the SAP libraries are installed.

To install the SAP libraries:

1. In Studio, click **Tools** → **Install Module Providers**.
2. Click the plus sign (+) under SAP Connector to add the SAP JCo V3 Java libraries, sapjco3.jar and sapidoc3.jar, and the native SAP library, sapjco3.dll, as highlighted in Figure 2-13. (Studio is Windows-based, so the Windows version of the native libraries are needed.)

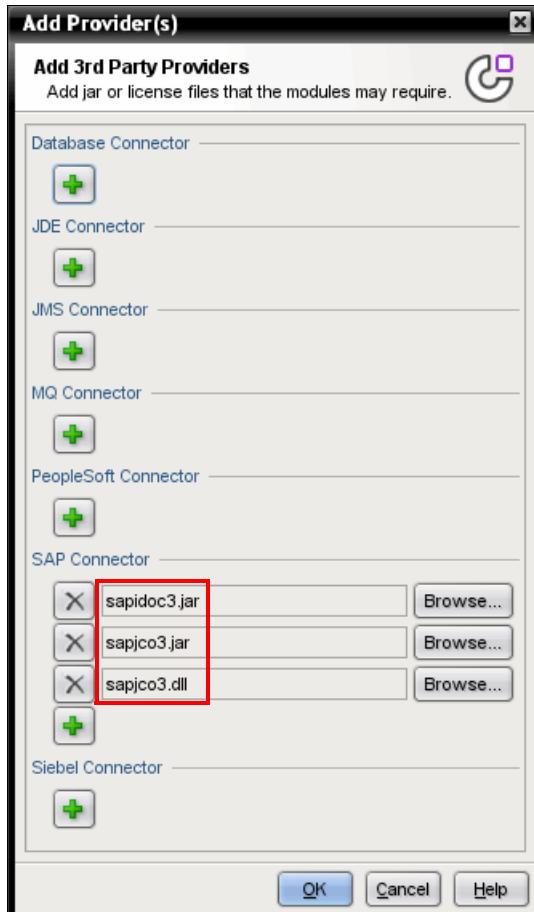


Figure 2-13 Studio: Add module providers

3. Click **OK** to upload the libraries. Studio is stopped after the upload, and you must start Studio again manually.

2.1.4 Installing plug-in connectors

Plug-in connectors are additional connectors that you can download into Studio and access in the same manner as a built-in connector. You must be a registered user to access the WebSphere Cast Iron plug-in repository. If you are not a registered user, you can submit a request at:

<https://community.castiron.com/user/register>

To install the currently available plug-in connectors:

1. In Studio, click **Solutions** → **Plugin Connectors**.
2. Log in to access the plug-in repository. A list of available Plugin Connectors displays, as shown in Figure 2-14.

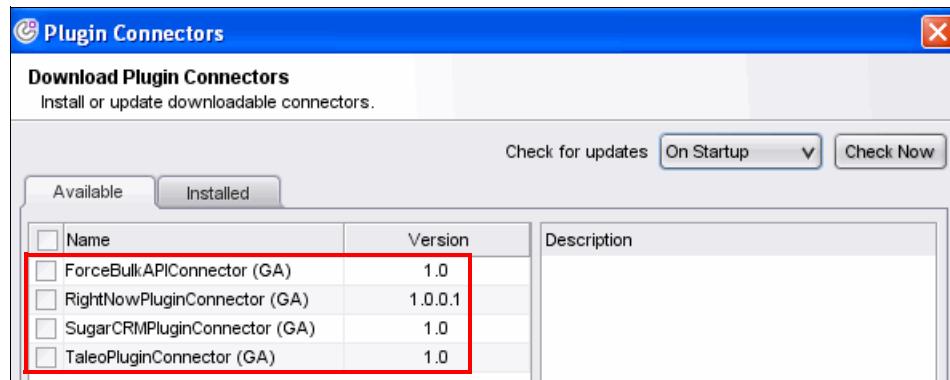


Figure 2-14 Studio: Download Plugin Connectors

3. Select the plug-in connectors that you want to download, and click **Install**. Click **Close** after the connectors are installed.

After you download a connector, the connector displays in the Activities tab with the built-in connectors, as shown in Figure 2-15.

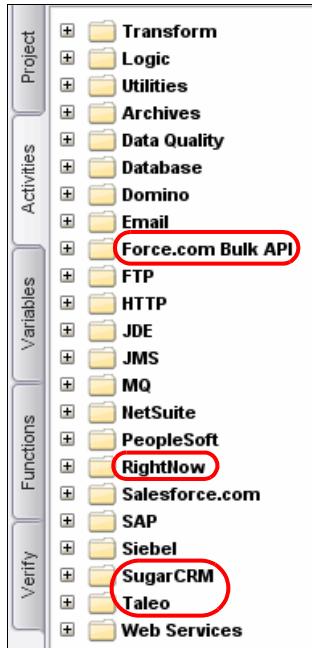


Figure 2-15 Studio: Enhanced activity list

You can now drag the plug-in connector activities into the Studio workspace to create an orchestration.

2.2 Installing and configuring the physical appliance

The WebSphere DataPower Cast Iron Appliance XH40 is a self-contained physical appliance that includes the WebSphere Cast Iron Runtime V6.1. Figure 2-16 shows the appliance.



Figure 2-16 Appliance XH40 front view

The physical appliance has six network interfaces, which are located to the right on the front of the appliance. From these six network interfaces, WebSphere Cast Iron uses the four that are indicated in Figure 2-17.

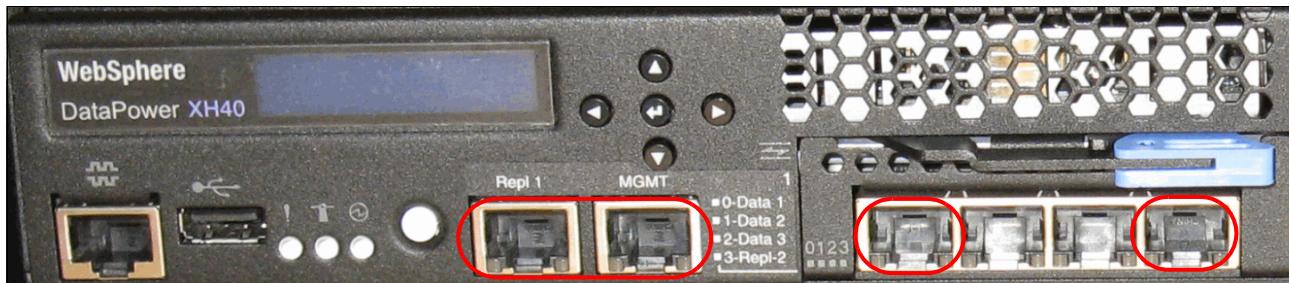


Figure 2-17 Appliance XH40 front ports

The network interfaces from left to right are:

- ▶ Repl 1: The first replication interface that is used for high availability scenarios
- ▶ MGMT: The interface for management traffic
- ▶ Data 1: The interface for data integration traffic
- ▶ Data 2: Currently not used in WebSphere Cast Iron
- ▶ Data 3: Currently not used in WebSphere Cast Iron
- ▶ Repl 2: The second replication interface that is used for high availability scenarios

The RJ45 serial port: The interface at the far left of the appliance's front looks similar to a network interface, but it is the serial port (RJ45). This port connects a terminal that is used primarily for initial configuration.

2.2.1 Installing and starting the appliance

This section explains how to install the physical environment into an existing network. It does not provide details about how to physically install the appliance into the data center.

To install the physical appliance in a stand-alone mode, you must provide the following components:

- ▶ An Ethernet cable for the data interface that is used for integration data traffic
- ▶ An Ethernet cable for the management interface that is used for access for management tools, such as the Web Management Console (WMC) and the Command Line Interface (CLI)

The management network can be on the same network as the data network or on a separate network.

- ▶ A serial console cable to connect the RJ45 serial port of the appliance with the PC

The appliance ships with a RJ45-to-DB9F serial cable. This cable is sufficient if your PC provides a DB9M serial port. The scenarios in this book use a DB9M-to-USB adapter.

- ▶ A PC with an appropriate terminal program

To configure the appliance, a terminal with a keyboard and display is required. The scenarios in this book use a PC with terminal emulation program, referred to as the *management PC*. For Windows-based management PCs, the Windows HyperTerminal program can be a good choice of a terminal program. For more information, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.cli.doc%2FCLI_log_hyperterminal.html

The scenarios in this book use a management PC that is based on Linux and use Putty as the terminal program.

To connect to the physical appliance:

1. Plug the DB9F plug of the RJ45-to-DB9F serial adapter cable into the DB9M plug of the DB9M-to-USB adapter.
2. Plug the RJ45 plug of the serial adapter cable into the RJ45 port of the appliance.
3. Plug the USB plug of the serial adapter cable into one of the USB ports of the management PC.
4. Start the physical appliance.
5. Start the terminal program.
6. Configure the terminal program to connect to the serial cable.

The settings that you use depend on the port that is used in the management PC, on the serial or USB port, and on the terminal program. The scenarios in this book, with the USB port and Putty on Linux, use the /dev/ttyUSB0 serial line and a 9600 speed.

7. A login panel displays, as shown in Example 2-1.

Example 2-1 Physical appliance login panel

```
IBM WebSphere DataPower Cast Iron Appliance (6.1.0-0000)
Copyright IBM Corp. 2003, 2010. All Rights Reserved.
none login:
```

2.2.2 Gathering basic information about the appliance

To determine basic information about the appliance:

1. Log in at the serial terminal as the administrator to access the CLI. (The default user is admin, and the default password is !n0r1t50C).

Tip: As long as you connect to the CLI using the terminal program, the keyboard is either set to the settings of the operating system or to the default settings that are defined in the terminal program. You might have to adjust the terminal keyboard settings to another emulation, or you might have to use Shift+2 instead of the at symbol (@) in the password. If you still run into problems, use the **resetpass** command to change the password.

The prompt varies, depending on the host name setting and the high availability configuration. If nothing is assigned, the prompt displays after you log in as one of the following prompts:

- A none prompt indicates that a host name is not assigned.
- A Standalone prompt indicates that high availability is not configured.

2. Verify the version using the **system show version** command. The output shown in Example 2-2 on page 41 indicates that the appliance is at the version level 6.1.0.0.

Important: The project that produced this book was run using an early version of WebSphere Cast Iron. Be aware that version 6.1.0.0 requires the mandatory fixpack 6.1.0.1: <http://www-304.ibm.com/support/docview.wss?uid=swg21516074>. If your version is 6.1.0.0, you must install the fixpack.

Example 2-2 Physical appliance system show version

```
none/Standalone> system show version
Model: Cast Iron XH40, Revision 2
Version: Cast Iron Operating System 6.1.0.0000 (Mon Aug 22 11:38:16 UTC 2011)
Serial Number: 6800193
```

3. Gather additional details about the physical appliance using the **system show platform** command. Example 2-3 shows the output of this command.

Example 2-3 Physical appliance system show platform

```
none/Standalone> system show platform
ROM Version: I009
Platform: 71988FX
Serial Number: 6800193
UUID: 03000200-0400-0500-0006-000700080009
Board Product: Greencyte
Board Serial Number: serial_number
MAC0: 00:0b:ab:51:11:b0
MAC1: 00:0b:ab:51:11:b1
Access Key: DKEY-3L9G-HLB4-4NGO
```

2.2.3 Setting up the appliance

This section explains how to configure the physical appliance. By default, the appliance is configured to use DHCP. At startup, it connects to the DHCP server and retrieves the required network configuration details, such as the host name and IP address. So, by default, you do not have to configure the appliance for it to run.

You do need to verify the status and determine the IP addresses that are assigned. However, physical appliances are typically configured to use static addresses. So, this section concentrates on how to set up the appliance with static network settings. For a setup with DHCP, refer to 2.4.3, “Setting up the appliance using DHCP” on page 52, because the process is similar.

Table 2-2 lists the values you must capture before setting up the appliance.

Table 2-2 Physical appliance: Required network parameters

Network	Parameter used in command	Explanation	Value
Management network (emgmt) Network used for administrative traffic	<i>appliance hostname</i>	Host name of the appliance	
	<i>domain-name</i>	Domain name of the appliance	
	<i>mgmt ip address</i>	IP address for management network	
	<i>mgmt netmask</i>	Subnet Mask for management network	
	<i>mgmt broadcast</i>	Broadcast for management network	
	<i>dns-ipaddress</i>	Primary DNS address	
	<i>gateway ip-address</i>	Gateway for management network	
	<i>ntp-ipaddress</i>	Network Timeserver address (can be none)	
Data network (edata) Network used for integration data traffic	<i>data ip address</i>	Data IP address	
	<i>data netmask</i>	Data Subnet Mask	
	<i>data broadcast</i>	Broadcast	

To improve startup time, configure the parameters to either static values or none. For additional details about the network parameters, refer to the Cast Iron CLI reference, which is available at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.cli.doc%2FCLI_network_netset.html

To set up the appliance network:

1. Complete the information listed in Table 2-2.

To configure the appliance based on the values in Table 2-2, use the following commands (the scenarios in this chapter do not use an NTP server or search the domain list):

- **net set hostname static <appliance hostname>**
- **net set domain static <domain-name>**
- **net set nameserver static <dns-ipaddress>**
- **net set gateway static <gateway ip-address> emgmt**
- **net set search none**
- **net set ntp none (net set ntp static <ntp-ipaddress>)**
- **net set interface emgmt address <mgmt ip address> mask <mgmt netmask> bcast <mgmt broadcast>**
- **net set interface edata address <data ip address> mask <data netmask> bcast <data broadcast>**

When you issue these commands, the settings are not activated automatically but are kept in memory to become active the next time the network is started.

2. Before activating these settings, make sure that there is no configuration error. Verify the configuration using the **net show memory** command. Example 2-4 shows the output for this command in the test environment for this book.

Example 2-4 Physical appliance: Net show memory

```
none/Standalone> net show memory
Hostname: castiron1
Domain: localdomain
Domain Search: [None]
Nameservers: 192.168.100.1
Gateway: 192.168.100.254 on emgmt
NTP: [None]

Carrier timeout: 30 seconds

Emgmt(00:0b:ab:51:11:b1): Static
    Address: 192.168.100.44, Netmask: 255.255.255.0, Broadcast: 192.168.100.0
    Emgmt Link: autonegotiate
Edata(00:0b:ab:50:bc:c4): Static
    Address: 192.168.100.55, Netmask: 255.255.255.0, Broadcast: 192.168.100.0
    Edata Link: autonegotiate

Routes: [None]
```

3. After validating the settings, restart the network using the **net restart** command. Verify the new settings using the **net show active** command. If the network is active and a host name is assigned, the command prompt switches from <none> to <hostname>.

Link not ready message: If the network is not set up correctly, a message similar to the following message might display. Review your installation process to ensure the network is set up correctly.

```
ADDRCONF(NETDEV_UP): eth1: link is not ready
e1000e: eth1 NIC Link is Up 100 Mbps Full Duplex, Flow Control: RX/TX
0000:0f:00.0: eth1: 10/100 speed: disabling TSO
ADDRCONF(NETDEV_CHANGE): eth1: link becomes ready
```

4. Determine the system status using the **system show status** command.

Example 2-5 shows a sample output from this command. The status for the network and run time might also be *Starting* or *Down*.

Example 2-5 Output from the system show status command

```
Appliance Status
-----
System: Up
Network: Up
Runtime: Up
Command complete
```

If the status of the network is Down or Starting, wait a couple of minutes, and then issue the **system show status** command again.

Tip: If the `net show active` command displays the older values and the `net show memory` command displays the newer values, reboot the appliance using the `system reboot` command.

5. Use ssh to connect to the management IP address in a more comfortable way.
6. Determine and validate the additional network settings, such as the MAC Address and Link Speed, using the following command:

```
netspect ifconfig interface <emgmt | edata>
```

Additional resources: For additional details about network settings and troubleshooting, refer to the Cast Iron CLI reference, which is available at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.cli.doc%2FCLI_about_CLI.html

To finalize the initial installation, log in to the WMC. For more information, see 2.5, “Finalizing the installation using the WMC” on page 55.

2.3 Configuring the physical appliance for high availability

High availability (HA) pairs consist of two physical appliances:

- ▶ An active appliance
- ▶ A standby appliance

This HA pair can then process orchestrations as a single entity.

The active appliance actively processes orchestrations. The standby appliance synchronizes data automatically with the active appliance in case the standby appliance needs to perform any failover operations. The active and standby appliances can switch roles, which is useful for maintenance or hardware tests, for example.

The WebSphere DataPower Cast Iron Appliance XH40 supports HA with an active/standby configuration. To provide this HA, you must install two physical appliances, as shown in Figure 2-18.



Figure 2-18 Front of the physical appliances in HA

The two appliances act as one appliance. To do so, they must stay synchronized and must know the status of each other.

The HA installation and setup consists of the following steps:

1. Install both appliances as separate boxes.

2. Connect the two appliances.
3. Activate high availability.
4. Test HA takeover.

When HA is active, both appliances share the host name and IP addresses and stay synchronized.

2.3.1 Prerequisites to set up high availability

For an HA configuration, you must provide two identical appliances with the same version of WebSphere Cast Iron. In addition, you need the following components:

- ▶ Two Ethernet cables for the two data interfaces

The data interface is used for integration data traffic. For HA, make sure that one appliance can connect to the data network if the other appliance is not working.
- ▶ Two Ethernet cables for the two management interfaces

The management interface manages traffic. For HA, make sure that one appliance can connect to the management network if the other is not working.
- ▶ A serial console cable to connect the RJ45 serial port of the appliance with the PC

You connect to the appliance using a serial cable. Although there are two appliances, you need only one appliance connected using a serial cable.
- ▶ A PC with an appropriate terminal program

You need a management PC with a terminal program. The scenarios in this book use a management PC that is based on Linux and uses Putty as the terminal program.
- ▶ Two Ethernet cables for HA related traffic

For HA, the two appliances must be synchronized and must be able to reach each other. For the HA-related data synchronization and for the heartbeat, the appliances use a direct link between the boxes. For redundancy, there are two direct links between the appliances.

In summary, you need a total of six Ethernet cables (but only four IP addresses), a serial cable, and a terminal.

For more details about the terminal program and the serial cable, refer to 2.2.1, “Installing and starting the appliance” on page 39.

To set up an HA configuration, you must provide more information than for a stand-alone configuration. Table 2-4 on page 54 lists the values that you must determine before setting up the appliances. For ease of documentation, we refer to *appliance1* and *appliance2*.

Table 2-3 Physical appliances in an HA configuration: Required network parameters

Network	Parameter in command	Explanation	Value
Management network (emgmt) Network used for administrative traffic	<i>appliance1 hostname</i>	Host name of the appliance1	
	<i>appliance2 hostname</i>	Host name of the appliance2, only used during setup.	
	<i>domain-name</i>	Domain name of the appliances	
	<i>mgmt ip address for appliance1</i>	IP address for management network for appliance1	
	<i>mgmt ip address for appliance2</i>	IP address for management network for appliance1, only used during setup.	
	<i>mgmt netmask</i>	Subnet Mask for management network	
	<i>mgmt broadcast</i>	Broadcast for management network	
	<i>dns-ipaddress</i>	Primary DNS address	
	<i>gateway ip-address</i>	Gateway for management network	
	<i>ntp-ipaddress</i>	Network Timeserver address (can be none)	
Data network (edata) Network used for integration data traffic	<i>data ip address for appliance1</i>	Data IP address for appliance1	
	<i>data ip address for appliance2</i>	Data IP address for appliance1, only used during setup.	
	<i>data netmask</i>	Data Subnet Mask	
	<i>data broadcast</i>	Broadcast	

2.3.2 Configuring the appliances for HA

The sections that follow describe how to configure the appliances for HA.

Installing both appliances as stand-alone appliances

First, install both appliances as stand-alone appliances:

1. Install each appliance as a stand-alone instance. Refer to 2.2, “Installing and configuring the physical appliance” on page 38.

Use static network settings, and make sure that the two appliances have different IP addresses but are within the same network (the same management network and the same data network).

- Verify that the two appliances have the same version of WebSphere Cast Iron installed using the **system show version** command. Upgrade to the latest available version if necessary.

Make sure at the end of the installation process that both appliances are up and running in parallel and that each appliance has the run time status Up.

Connect the cables for replication

As shown in Figure 2-19, there are two replication ports for each appliance. Connect the following cables:

- Connect the replication port Repl1 of appliance1 with the replication port Repl1 of appliance2.
- Connect the replication port Repl2 of appliance1 with the replication port Repl2 of appliance2.

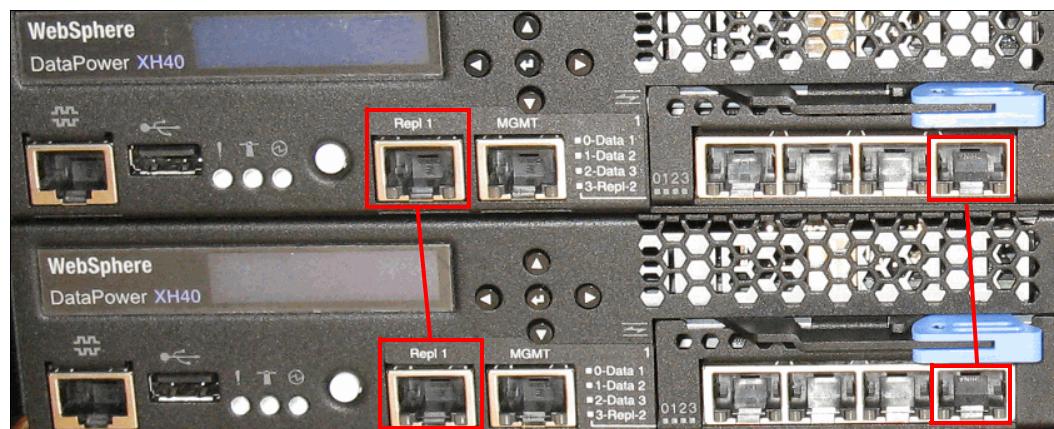


Figure 2-19 Ports of the physical appliances in HA

Activating HA

To activate HA:

- Use the serial cable to connect to the serial port of the primary appliance.
- The WebSphere Cast Iron Login screen displays in the CLI of the primary appliance. Log in as the administrator.
- Issue the **system haconfig enable active** command.

The appliance reboots and restarts as the active appliance.

- When the WebSphere Cast Iron Login panel of the CLI displays, log in again as the administrator.

The appliance name/Active> command prompt indicates that the appliance is the active appliance.

- Disconnect the serial console from the active appliance and connect it to the second appliance.
- The WebSphere Cast Iron Login screen displays in the CLI of the second appliance. Log in as the administrator.
- Issue the **system haconfig enable standby** command to configure the second appliance as the HA standby appliance.

The appliance reboots and restarts as the standby appliance.

- When the WebSphere Cast Iron Login panel of the CLI displays, log in again as the administrator.

The none/Standby> command prompt indicates that the appliance is the standby appliance.

- Switch the serial console back to the active appliance, and check the HA using the **ha show pairstatus** command.

If synchronization has not completed, the status is as shown in Example 2-6.

Example 2-6 Physical appliances: HA Pair Status not complete

Pair Status

```
-----  
NOT Highly Available  
Local State: ACTIVE_2  
Peer State: STANDBY  
Sync Status: 20%
```

Appliance Services

```
-----  
Network: Up  
Runtime: Starting
```

Command complete

The output indicates that the appliances are not highly available because the synchronization has not yet finished.

- The synchronization can take up to 15 minutes. So, wait a few minutes, and then issue the **ha show pairstatus** command again.

After synchronization finishes, the pair status looks as shown in Example 2-7.

Example 2-7 Physical appliances: HA Pair Status complete

Pair Status

```
-----  
Highly Available  
Local State: ACTIVE_2  
Peer State: STANDBY  
Sync Status: Synced
```

Appliance Services

```
-----  
Network: Up  
Runtime: Up
```

Command complete

Testing HA takeover

Log in to the active appliance, and issue the **ha switch** command to switch the roles of the HA pair. The standby peer takes over and becomes active. It takes over the host name and the IP addresses. The previously active appliance restarts.

Issue the **ha switch** command again to switch back to the original roles.

During takeover, by default, no synchronization is required.

For more details, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.HAOverview.doc%2FHAsupupair.html

2.3.3 Managing physical appliances in an HA configuration

This section lists typical questions when using an HA configuration.

How do I shut down a pair of HA appliances?

Log in to the active appliance, and issue the `ha poweroff standby` command. This action shuts down the standby appliance. The peer status switches to unknown in the `ha show pairstatus` command output.

To shut down the active appliance, issue the `system poweroff` command.

How do I start a pair of HA appliances?

Start the primary appliance with a power on, and wait until the appliance is up and running. On the CLI, issue the `ha poweron standby` command to start the standby appliance.

How do I change network settings in a pair of HA appliances?

Issue the commands on the active appliance (for example `net set gateway static`). Then, issue the `net restart` command to make the changes become active.

The changed network settings are replicated automatically to the standby appliance.

How do I apply maintenance on a pair of HA appliances?

You can apply a fix pack to the active appliance using either the WMC or the CLI. If HA is configured and both appliances are up and running, the fix pack is synchronized automatically and installed on the standby appliance without additional effort.

2.4 Installing and configuring the virtual appliance

You can install the WebSphere Cast Iron Hypervisor Edition virtual appliance on your existing hardware using virtualization technology. It is delivered in the following formats:

- ▶ As an open virtualization archive (OVA) image for the hypervisor based on VMware
- ▶ As a .tar file image for the hypervisor based on Xen

The examples in this book use the OVA image; however, the configuration is the same for both platforms.

A virtual appliance has the following network interfaces:

- ▶ A management interface that is used for access to management tools, such as the WMC or the CLI
- ▶ A data interface that is used for integration data traffic.

The two interfaces can be connected to the same or different networks but cannot share the same IP address.

2.4.1 Installing and starting WebSphere Cast Iron Hypervisor Edition

Before you configure the hypervisor image, you must deploy the image. With the license of WebSphere Cast Iron Hypervisor Edition, you have access to both the VMware and Xen images. You need to deploy one of these images:

- ▶ **VMware**

The VMware image uses the OVA format. For V6.1.0.1, the image is named 6.1.0.1-WS-WCI-20110915-1504_H2.ova.

Check the prerequisites before deploying the image:

<http://www-01.ibm.com/software/integration/cast-iron-cloud-integration/reqs>

Deploy the OVF image to the VMware ESX server as described at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.VAuserguide.doc%2FVA_deployingOVFtemplate.html

- ▶ **Xen**

The Xen image is packaged as a WebSphere Cast Iron Xen Hypervisor Edition .tar file. For V6.1.0.1, the image is named 6.1.0.1-WS-WCI-20110915-1504_H2.tar.

Xen is a virtualization technology that is available for the Linux kernel. WebSphere Cast Iron V6.1 Hypervisor Edition can run in a Xen Hypervisor 4.0 environment. Check the prerequisites before trying to deploy the image. You can find them at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.VAuserguide.doc%2FVA_Xeninstall.html

Extract and install the WebSphere Cast Iron Hypervisor Edition Xen image, as described at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.VAuserguide.doc%2FVA_Xeninstall.html

After you deploy the virtual appliance to the hypervisor, configure it as follows:

1. Start the WebSphere Cast Iron Hypervisor Edition instance.
2. Connect to the hypervisor console to have a display for the appliance (for example using VMware vSphere Client). The hypervisor console is needed only for initial setup or if the network is down. After the network configuration is set up correctly, you can use ssh to connect to the virtual appliance.

During startup, a progress panel displays, as shown in Figure 2-20.

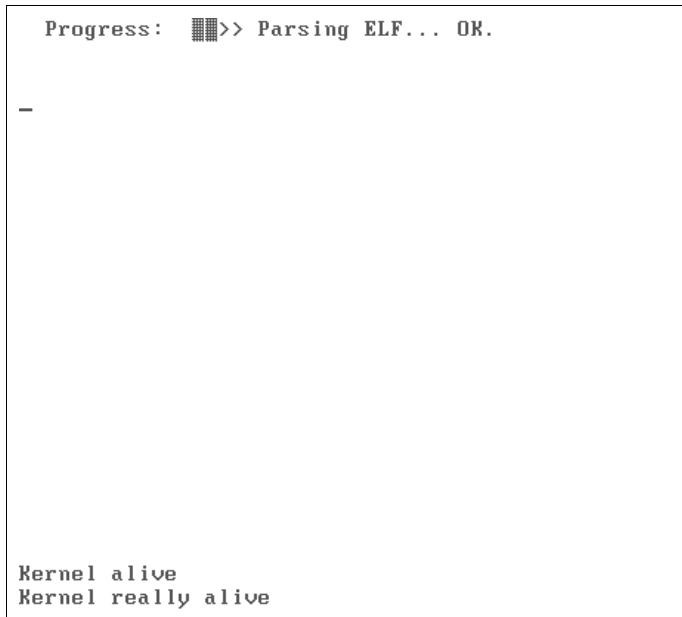


Figure 2-20 Startup panel for the virtual appliance

After some time, depending on the hardware, the login panel displays, as shown in Figure 2-21.

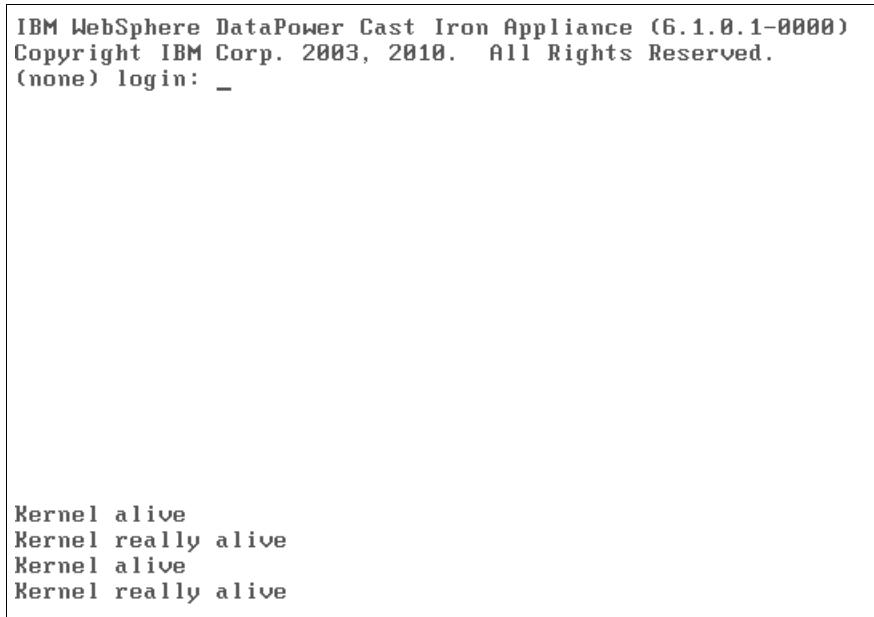


Figure 2-21 Login panel for the virtual appliance

2.4.2 Gathering basic information about the appliance

To determine basic information about the appliance:

1. Log in as the administrator to access the CLI (the default user is `admin`, and the default password is `!n0r1t50C`).

Password tip: The keyboard layout of the appliance is set to U.S. by default. If you are connecting using a console program, such as vSphere, you might need to use Shift+2 instead of the at symbol (@) in the password. If that does not help, use the **resetpass** command to change the password.

2. Verify the version using the **system show version** command, as shown in Example 2-8.

Example 2-8 The show version command for the virtual appliance

```
none/Standalone> system show version
Model: Cast Iron vA3000, Revision A
Version: Cast Iron Operating System 6.1.0.1.0000 (Thu Sep 15 14:16:52 UTC 2011)
Serial Number: VMW5NKY7RKX54NMA
```

Notice that the prompt varies, depending on the host name. If the host name was not assigned, the prompt appears as none/Standalone; otherwise, the prompt appears as *hostname*/Standalone.

3. Gather additional details about the virtual appliance using the **system show platform** command. The output of this command provides additional information, as shown in Example 2-9.

Example 2-9 The system show platform command for the virtual appliance

```
none/Standalone> system show platform
ROM Version: 6.00
Platform: VMware Virtual Platform
Serial Number: VMW5NKY7RKX54NMA
UUID: 423A3158-B97E-9B70-F2E2-D76EF8018893
Board Product: 440BX Desktop Reference Platform
Board Serial Number: None
MAC0: 00:50:56:ba:00:e8
MAC1: 00:50:56:ba:00:e9
Access Key: 36G3-Z077-RWYG-7E36
```

2.4.3 Setting up the appliance using DHCP

This section explains how to configure the virtual appliance when using DHCP. By default, the appliance is configured to use DHCP. At startup, it connects to the DHCP server and retrieves the required network configuration details, such as the host name and IP address. So, by default, no configuration is required.

You must verify the status and determine the IP addresses that are assigned as follows:

1. Determine the system status using the **system show status** command. Example 2-10 shows a sample output from this command. The status for the network and run time might also be Starting or Down.

Example 2-10 The system show status command output for the virtual appliance

```
Appliance Status
-----
System: Up
Network: Up
Runtime: Up
```

Command complete

If the status of the network is Down or Starting, wait a few minutes, and then issue the **system show status** command again.

If the network is still down, there is a problem getting the network settings using DHCP.

2. Issue the **net show active** command.

If the DHCP address cannot be retrieved, the network shows as inactive.

If the network is up, the two network addresses that are assigned for management and data display, as shown in Example 2-11.

Example 2-11 The net show active command output for the virtual appliance

```
ci-192-168-238-145/Standalone> net show active
Hostname (DHCP from emgmt): ci-192-168-238-145
Domain (DHCP from emgmt): localdomain
Domain Search (DHCP from emgmt): localdomain
Nameservers (DHCP from emgmt): 192.168.238.2
Gateway (DHCP from emgmt): 192.168.238.2
NTP (DHCP from emgmt): [None]

Carrier timeout: 30 seconds

Emgmt(00:0c:29:22:d1:ea): DHCP
  Address: 192.168.238.145, Netmask: 255.255.255.0, Broadcast: 192.168.238.255
  Link Status is not available
Edata(00:0c:29:22:d1:f4): DHCP
  Address: 192.168.238.146, Netmask: 255.255.255.0, Broadcast: 192.168.238.255
  Link Status is not available
  Routes: [None]

ci-192-168-238-145/Standalone>
```

3. Use ssh to connect to the management IP address, which makes it easier to copy and paste the command and results into the CLI or vice versa.

4. Determine and validate the additional network settings, such as the MAC Address and Link Speed, using the following command:

```
netspect ifconfig interface <emgmt | edata>
```

Additional resource: For additional details about network settings and troubleshooting, refer to the WebSphere Cast Iron CLI reference, which is available at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.cli.doc%2FCLI_about_CLI.html

2.4.4 Setting up the appliance using static network settings

You might want to use static network settings in the following circumstances:

- ▶ When the dynamic lookup of the network settings does not work
- ▶ When there are company requirements to set a static network configuration

Table 2-4 on page 54 lists the values that you must determine before setting up the appliance using a static network.

Table 2-4 Required network parameters for the virtual appliance

Network	Parameter in command	Explanation	Value
Management network (emgmt) Network used for administrative traffic	<i>appliance hostname</i>	Host name of the appliance	
	<i>domain-name</i>	Domain name of the appliance	
	<i>mgmt ip address</i>	IP address for management network	
	<i>mgmt netmask</i>	Subnet Mask for management network	
	<i>mgmt broadcast</i>	Broadcast for management network	
	<i>dns-ipaddress</i>	Primary DNS address	
	<i>gateway ip-address</i>	Gateway for management network	
	<i>ntp-ipaddress</i>	Network Timeserver address (can be none)	
Data network (edata) Network used for integration data traffic	<i>data ip address</i>	Data IP address	
	<i>data netmask</i>	Data Subnet Mask	
	<i>data broadcast</i>	Broadcast	

To improve startup time, configure all settings to either static values or none. For additional details about the network parameters, refer to the WebSphere Cast Iron CLI reference, which is available at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.cli.doc%2FCLI_network_netset.html

To set up the appliance network:

1. Complete the information listed in Table 2-4.
2. To configure the appliance based on the values in Table 2-4, use the following commands (the scenarios in this book do not use an NTP server or search the domain list):

- **net set hostname static <appliance hostname>**
- **net set domain static <domain-name>**
- **net set nameserver static <dns-ipaddress>**
- **net set gateway static <gateway ip-address> emgmt**
- **net set search none**
- **net set ntp none (net set ntp static <ntp-ipaddress>)**
- **net set interface emgmt address <mgmt ip address> mask <mgmt netmask> bcast <mgmt broadcast>**
- **net set interface edata address <data ip address> mask <data netmask> bcast <data broadcast>**

When you issue these commands, the settings are not activated automatically but are kept in memory to become active the next time the network is started.

3. Before activating these settings, make sure that there is no configuration error. Verify the configuration using the **net show memory** command.
4. After validating the settings, restart the network using the **net restart** command. Verify the new settings using the **net show active** command. If the network is active and a host name is assigned, the command prompt switches from <none> to <hostname>.
5. Determine the system status using the **system show status** command.

Example 2-12 shows a sample output from this command. The status for the network and run time might also be Starting or Down.

Example 2-12 The system show status command output for the virtual appliance

Appliance Status

```
-----
System: Up
Network: Up
Runtime: Up
Command complete
```

If the status of the network is Down or Starting, wait a few minutes, and then issue the **system show status** command again.

Tip: If the **net show active** command displays the older values and the **net show memory** command displays the newer values, reboot the appliance using the **system reboot** command.

6. Use ssh to connect to the management IP to make it easier to copy and paste results into the CLI or vice versa.
7. Determine and validate the additional network settings, such as the MAC Address and Link Speed, using the following command:

```
netspect ifconfig interface <emgmt | edata>
```

Tip: For additional details about network settings and troubleshooting, refer to the Cast Iron command line interface reference at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.cli.doc%2FCLI_about_CLI.html

2.5 Finalizing the installation using the WMC

To this point, you configured the Integration Appliance using the CLI. Now that the Integration Appliance is up and running, meaning that the management and data network and the run time are up, you can use the WMC to perform additional configuration tasks, such as creating users, installing libraries, and other tasks.

2.5.1 Logging in to the WMC

Follow these steps for the first login to the WMC:

1. Open a browser that is supported for use with the WMC.

You can find details about the supported browsers at:

<http://www-01.ibm.com/software/integration/cast-iron-cloud-integration/reqs/>

2. Insert the following URL:

`http://<management IP>`

Replace `<management IP>` with the management IP address that you retrieved using DHCP or with the IP address that was defined as static IP.

3. A Security Alert window displays explaining that the certificate cannot be trusted and does not match the requested site. Click **View Certificate** to get details about the certificate, as shown in Figure 2-22.



Figure 2-22 Security alert during first login to WMC

4. As indicated in Figure 2-23 on page 57, the Integration Appliance uses a self-signed certificate that was issued by Cast Iron Appliance to *Cast Iron Appliance*. As indicated by the "Valid from" date, the certificate is valid for two years from day of installation.

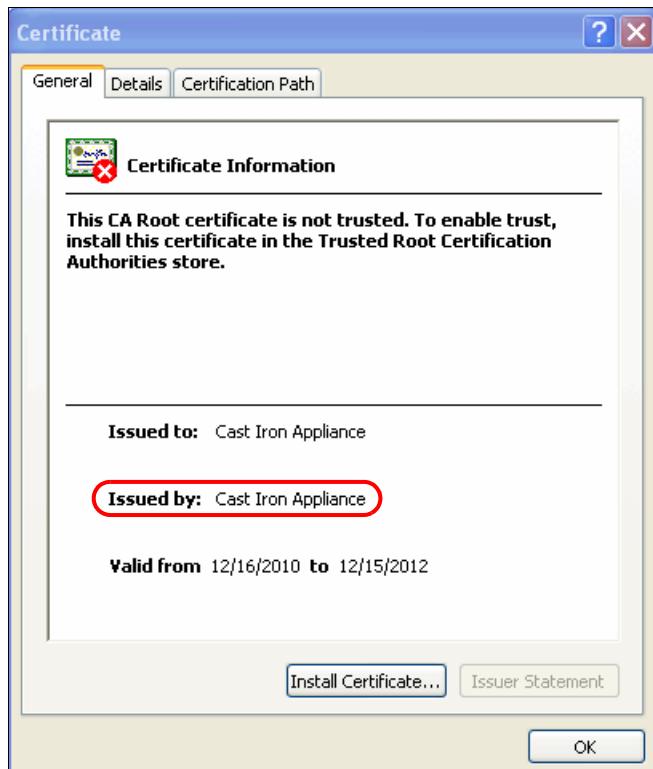


Figure 2-23 Certificate of the Integration Appliance

Install the certificate into your browser if you do not want to see the warning again. However, a better approach is to replace the certificate shortly after installation and to keep the certificate uninstalled as a reminder.

Click **OK** to close the Certificate window.

The WMC login panel displays. Insert the credentials for the administrative user, as shown in Figure 2-24 on page 58. (The default user name is admin, and the default password is !n0r1t5@C).

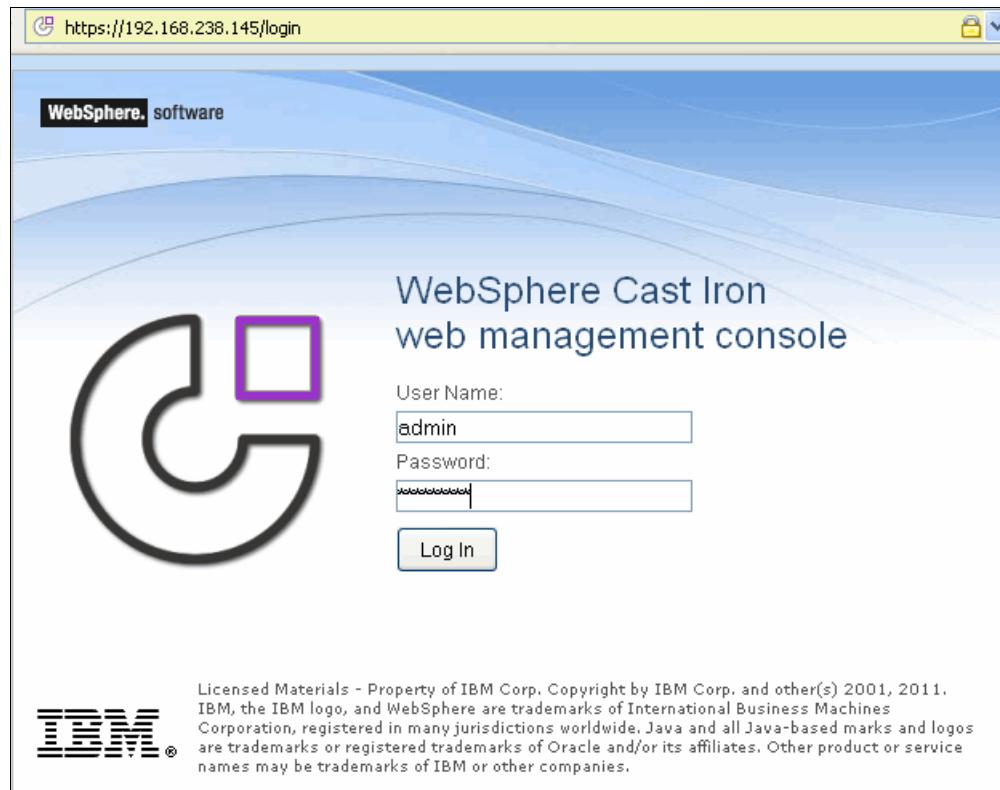


Figure 2-24 WMC login panel

- The WMC uses Adobe Flash Player to display the content. If the player is not installed, you are prompted to install it. If necessary, install the plug-in. If you need to restart the browser after the installation, log in to the WMC again.

You can now use the WMC. The console looks similar to that shown in Figure 2-25.

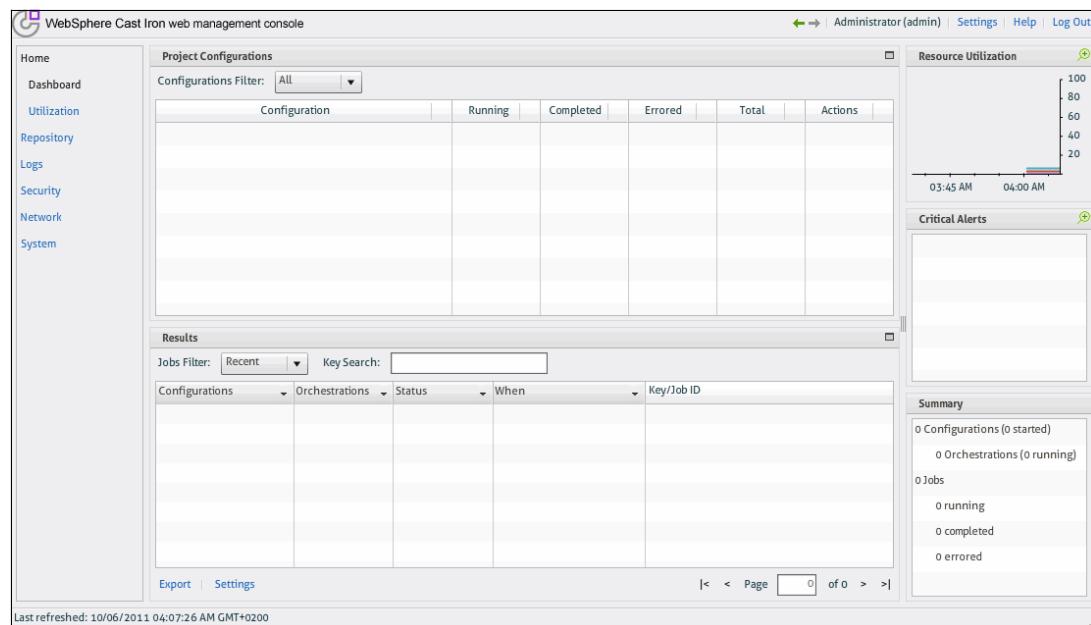


Figure 2-25 WMC overview

Change the admin password: For security purposes, change the admin default password. For more information, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.appliance.doc%2FPermissions%2FchangingPasswordOthers.htm

2.5.2 Installing the required SAP module providers

During the installation of WebSphere Cast Iron Studio, you installed the SAP module provider into Studio (as described in 2.1.3, “Installing module providers” on page 36). Because the module provider is not bundled and deployed with projects, you must also deploy it to the run time to support connectivity between the Integration Appliance and SAP.

Tip: Some browsers are restrictive in supporting self-signed certificates and might block browser functions, such as file upload, which is required for the installation of module providers (and for uploading projects). The WMC might display an error message, as shown in Figure 2-26.

If you have this problem, either install a trusted certificate before starting the upload or use a different browser.

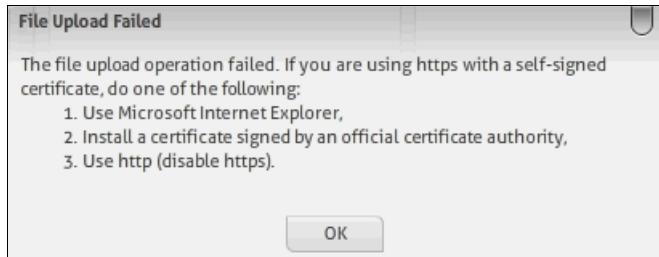


Figure 2-26 WMC message that the upload failed

To install the SAP libraries:

1. In the WMC, go to **System → Upgrade**.

The current version of the appliance displays.

2. Click **Update Connector Libraries**, as indicated in Figure 2-27.

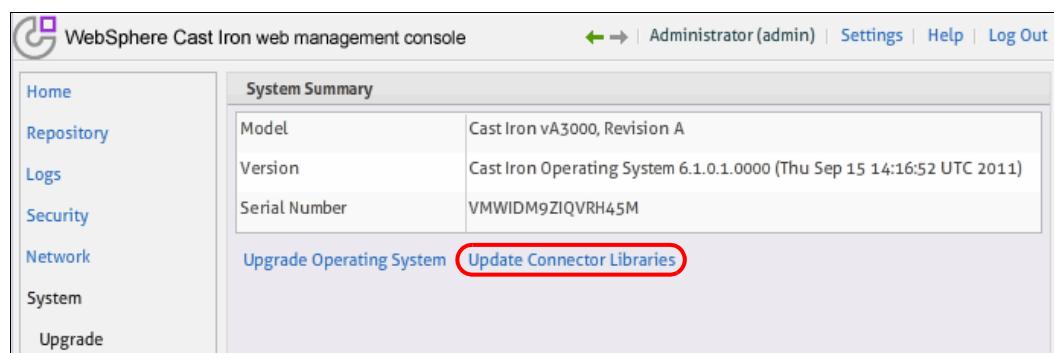


Figure 2-27 WMC System Summary

- In the Update connector Libraries panel, click the plus sign (+), as indicated in Figure 2-28.

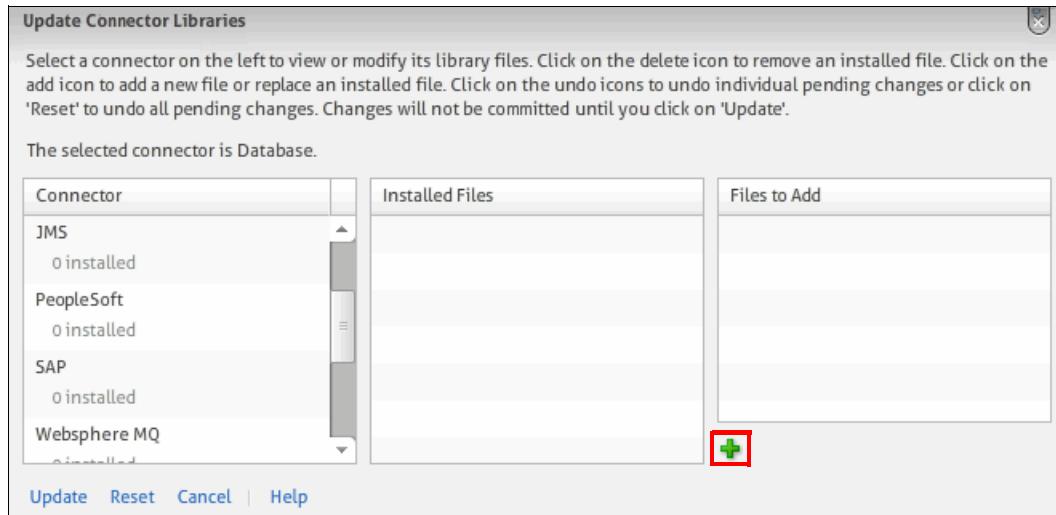


Figure 2-28 WMC connector library overview

- Add the two SAP JCo version 3.x.Java libraries, sapjco3.jar and sapidoc3.jar, and the native SAP library, libsapjco3.so. The panel now looks as shown in Figure 2-29.

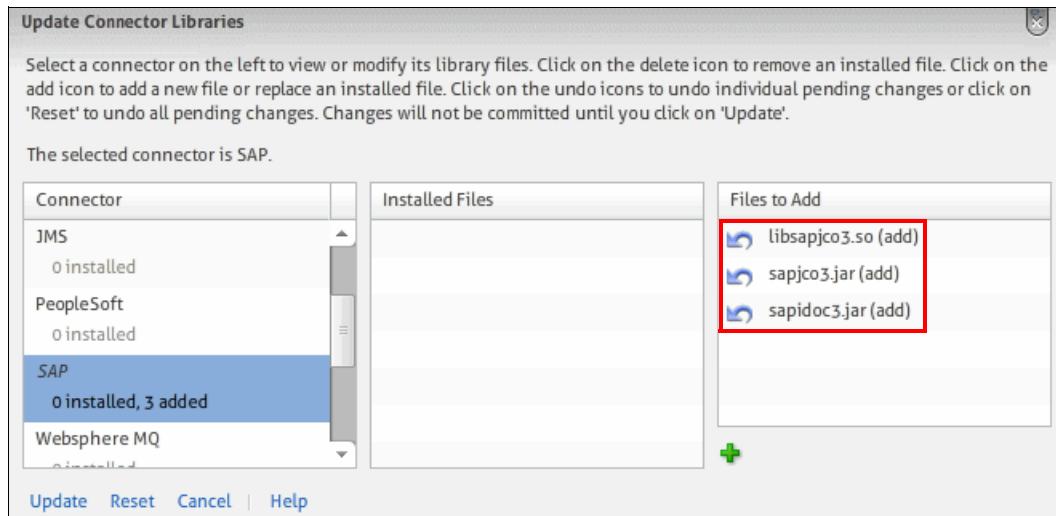


Figure 2-29 WMC install SAP libraries

- Click **Update** to upload the libraries. The appliance restarts after the upload. Now you can use the Integration Appliance with SAP.

2.6 Installing the WebSphere Cast Iron Live Secure Connector

The WebSphere Cast Iron Live Secure Connector is available for Cast Iron Live but is not available for the Integration Appliance. The Secure Connector provides a secure bidirectional data transfer between Cast Iron Live and the endpoints that are located behind a firewall.

The Secure Connector includes the following components:

- ▶ The Secure Connector component in Cast Iron Live

You must configure, in Cast Iron Live, the Secure Connectors that you want to use. Secure Connectors are defined in the scope of an environment.

- ▶ The Secure Connector component in the enterprise

You must install the Secure Connector program within your on-premise environment that you want to connect to Cast Iron Live through Secure Connector.

Figure 2-30 shows the architectural view of the Secure Connector.

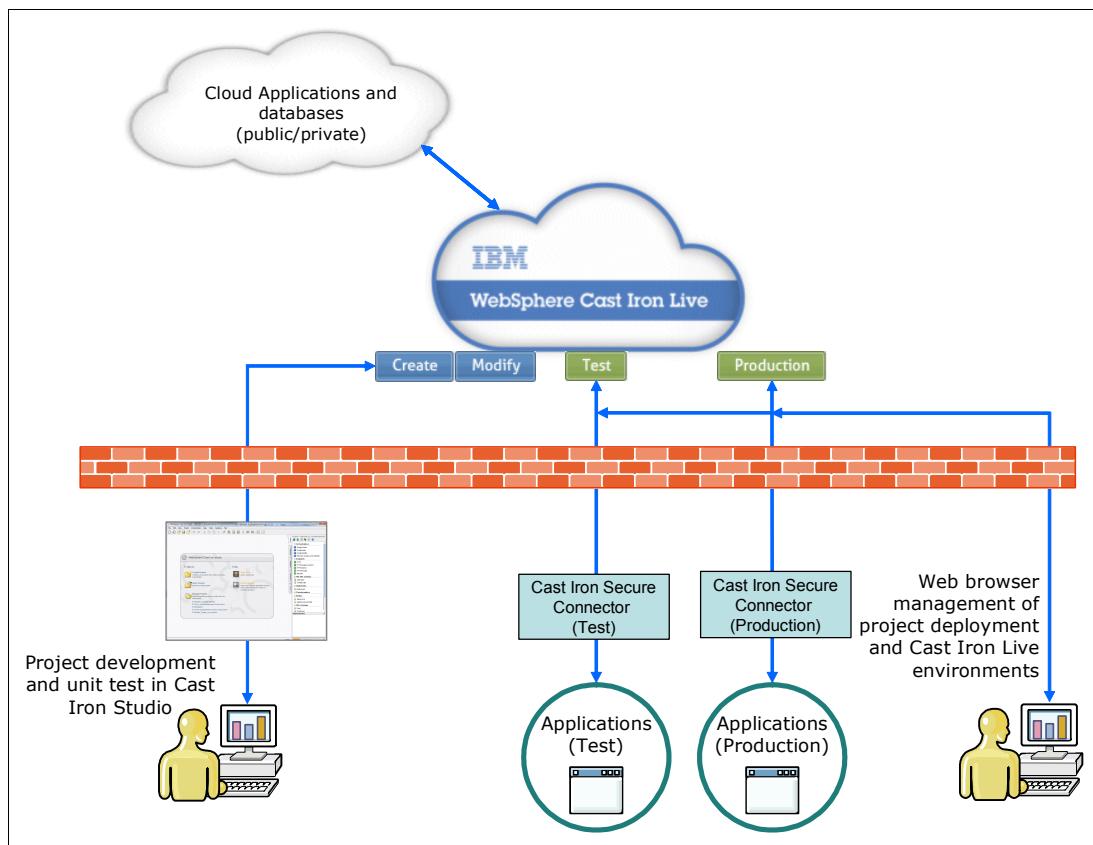


Figure 2-30 Secure Connector components

Tip: The machine running the Secure Connector must have access to the endpoint but does not have to run on the same machine as the endpoint.

Cast Iron Live uses the following environments and roles:

- ▶ The IBM WebSphere Cast Iron Cloud (the *Cloud*) supports multiple organizations (*tenants*). Each cloud tenant is managed by a tenant administrator.
- ▶ The Cloud supports multiple environments for each tenant, for example, you can have a Development, a Test, a Staging, and a Production environment.
- ▶ Although the Cloud administrator creates these environments for each tenant, it is the tenant administrator or the environment administrator who grants permissions to individual users or groups for each environment and who is allowed to install the Secure Connector.

To enable a Secure Connector, a tenant administrator or an environment administrator must complete the following tasks:

1. Create a Secure Connector in Cast Iron Live.
2. Download the Secure Connector configuration file and the installer.

Next, an administrator for the target system must complete these steps:

1. Install the Secure Connector into the target environment behind the firewall.
2. Install third-party libraries.
3. Start the Secure Connector and verify the communication.

The sections that follow describe how to set up the Secure Connector. For ease of documentation, we use one user for all steps.

2.6.1 Creating a Secure Connector in Cast Iron Live

A Secure Connector is defined for each cloud environment for security reasons. So, if you have multiple environments, (development, test, and production), you need to configure and install one Secure Connector per environment.

To create the Secure Connector in Cast Iron Live, you must have administrative rights for the target environment. Follow these steps in Cast Iron Live to set up a Secure Connector:

1. Log in as the administrator for the target environment.

When signing in to Cast Iron Live, four standard tabs display (the Home, Setup, Create, and Modify tabs.). Other tabs might display, depending upon your login settings.

This example uses the tenant administrator who is responsible for the Test, Development, and Production environments. The initial panel looks like Figure 2-31.



Figure 2-31 Cast Iron Live panel when logging in as the tenant administrator

2. Click the environment tab where you want to install the Secure Connector. In this example, WebSphere Cast Iron Secure Connector will be installed into the Test environment. So, click the Test tab.
3. Create a Secure Connector in Cast Iron Live:
 - a. On the Test tab, click **System** → **Secure Connectors**.
 - b. Click **New Secure Connector**, as indicated in Figure 2-32 on page 63.

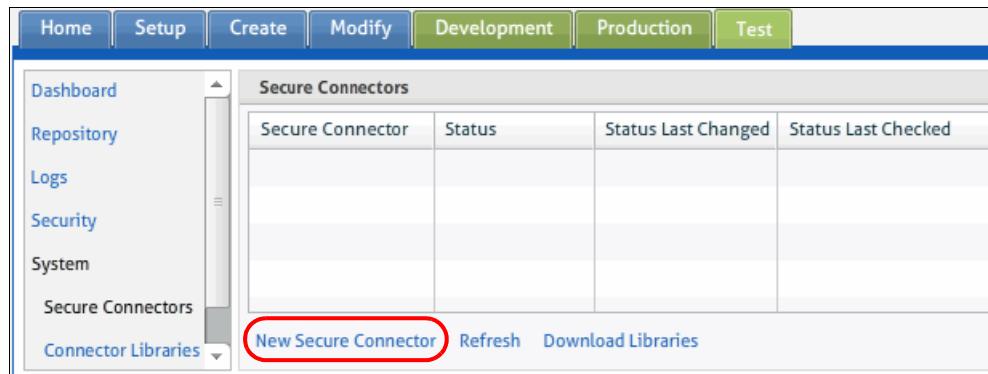


Figure 2-32 Cast Iron Live Secure Connector

- The Create Secure Connector dialog box displays. Specify a Secure Connector name (WinTestEnvironment), as shown in Figure 2-33, and optionally provide a brief description of the Secure Connector. Click **Save**.

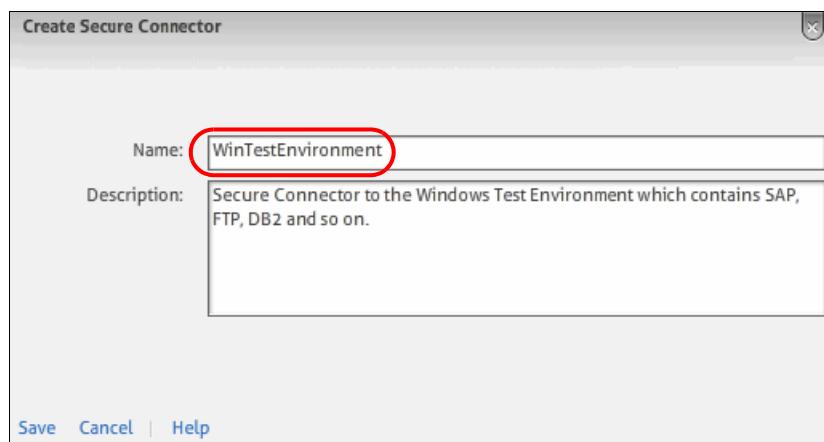


Figure 2-33 Cast Iron Live Create Secure Connector

2.6.2 Downloading the Secure Connector configuration and the installer files

Next, download the Secure Connector configuration file and the installer:

- From the Secure Connectors page, click the Secure Connector name. The Edit Secure Connector dialog box shows the authentication key.
- Click **Download Secure Connector Configuration**, as shown in Figure 2-34 on page 64.

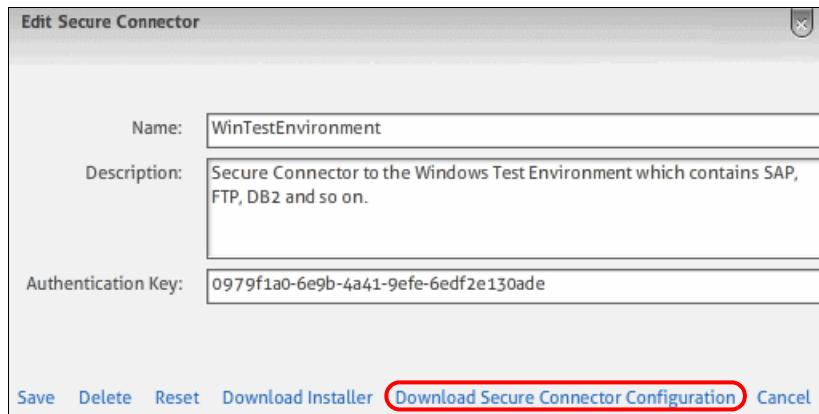


Figure 2-34 WebSphere Cast Iron Live Edit Secure Connector

3. Save the file on your local environment. Notice that the name of the file has the environment name and also the connector name (in this example SecureConnector-Test_WinTestEnvironment-Configuration), as shown in Figure 2-35.

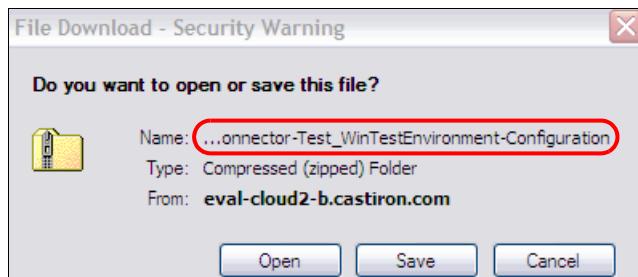


Figure 2-35 WebSphere Cast Iron Live downloading the configuration for Secure Connector

Click **OK** to continue.

4. Back in the Edit Secure Connector dialog box, click **Download Installer**. Choose either the Windows or Linux operating system (in this example, choose the Windows operating system). Save the file on your local environment. For a Windows operating system, this file is called windows-secure-connector-installer.exe, as shown in Figure 2-36.



Figure 2-36 WebSphere Cast Iron Live downloading the installer

2.6.3 Installing the Secure Connector on a Windows operating system

To install the Secure Connector into the target environment:

1. Copy the configuration and the installer files to the target environment.

Tip: Although you can place the installer file into a temporary directory, you must put the configuration file into a directory that is stable. During the installation, the configuration file is not copied. Instead, a reference to the file is created and used when the connector starts.

2. Click the installer file executable (`windows-secure-connector-installer.exe`) to start the installation wizard.
3. Depending on your security settings, a Security Warning panel might display. Click **Run** to allow the installer to continue.
4. The Welcome panel displays as shown in Figure 2-37. Click **Next** to continue.

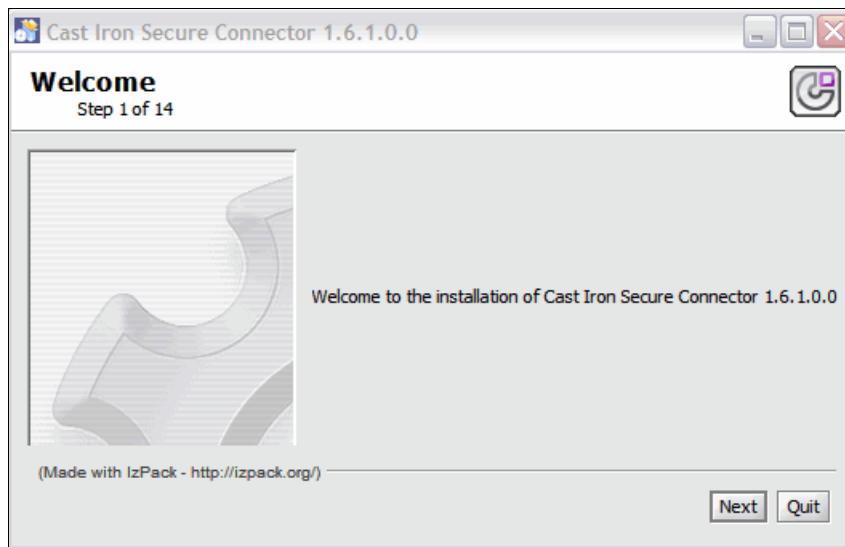


Figure 2-37 Cast Iron Secure Connector installation, Welcome panel

5. Read and accept the Licensing Agreement, and then click **Next**.
6. Specify the Target Path for the installation. The default path on a Windows operating system is:
`C:\Program Files\IBM\Secure_Connector_1.6.1.0.0`
Change this path to your path of choice. The examples in this book use the following path:
`C:\IBM\Secure_Connector_1.6.1.0.0`

7. You are informed that the target directory will be created, as shown in Figure 2-38. Click **OK**, and then click **Next** to continue.

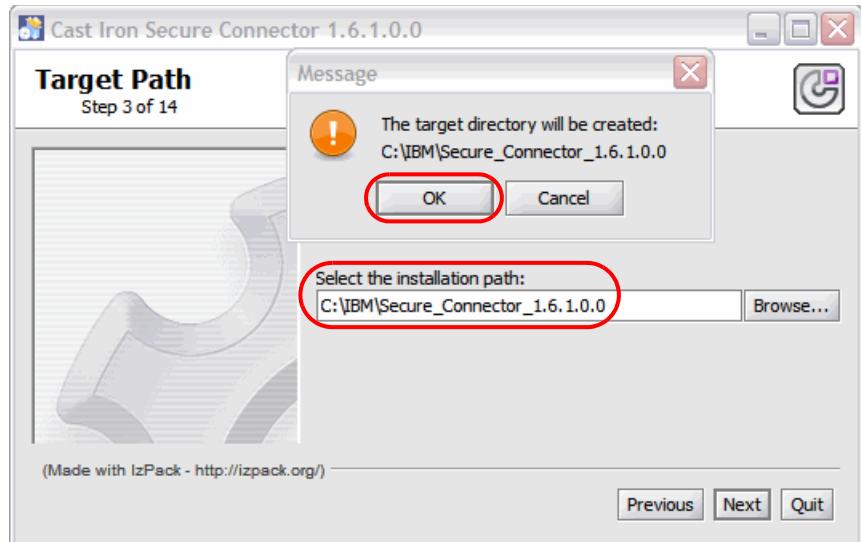


Figure 2-38 Cast Iron Secure Connector installation target path

8. Next, define the shortcuts. Accept the default settings, and then click **Next**.

Tip: We tested the installer in several environments, and there was no dialog box for step 5 of 14.

9. Now the installer starts the installation, as shown in Figure 2-39.

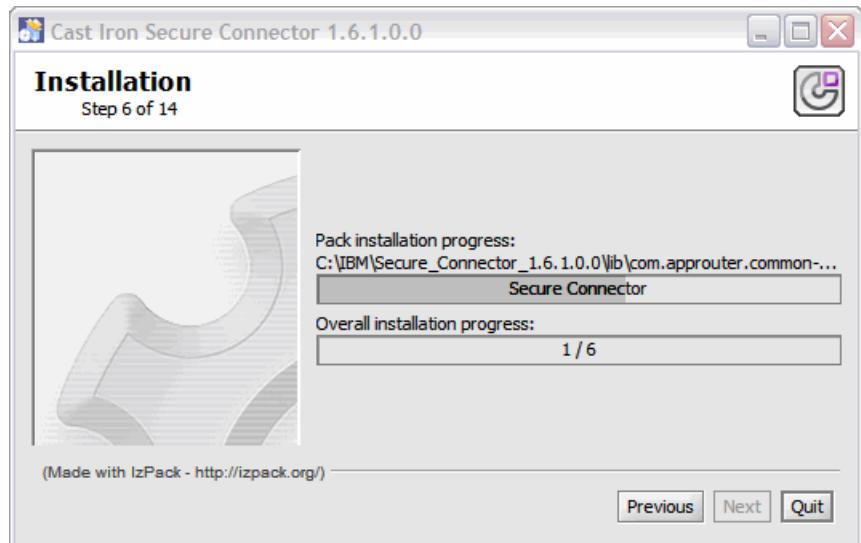


Figure 2-39 Cast Iron Secure Connector installation progress bar

It takes several minutes for the installation to complete. When finished, the progress bar looks as shown in Figure 2-40. Click **Next** to continue.

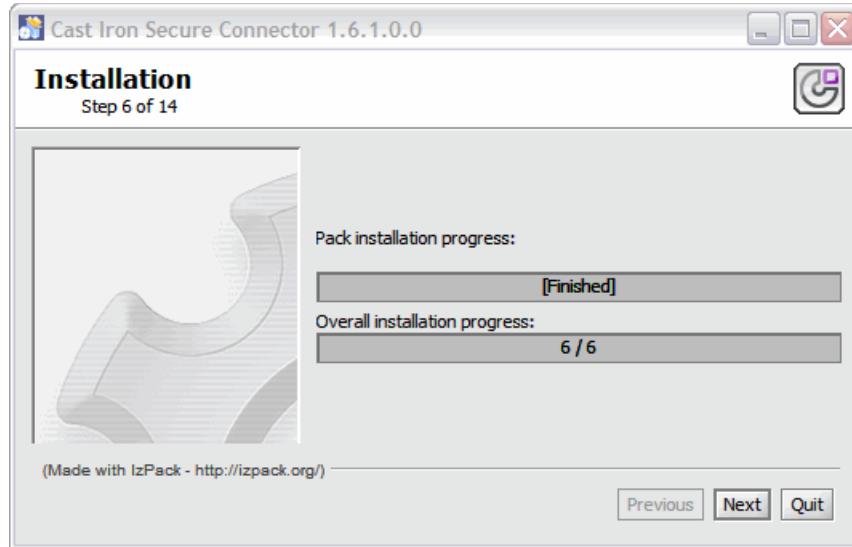


Figure 2-40 Cast Iron Secure Connector installation finished

10. Specify the location of the downloaded configuration file (SecureConnector-Test_WinTestEnvironment-Configuration) by clicking **Browse**, as indicated in Figure 2-41. Then, click **Next**.

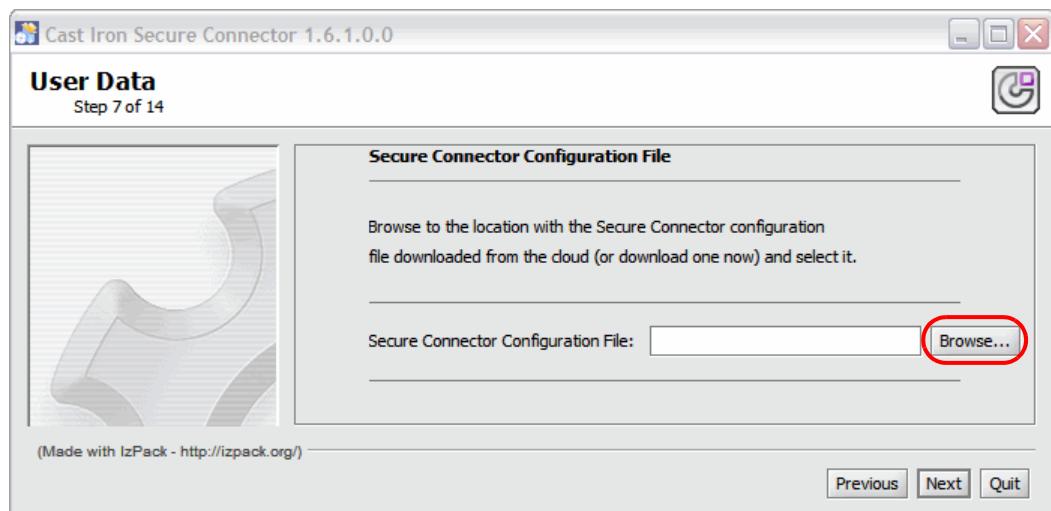


Figure 2-41 Cast Iron Secure Connector installation path to configuration file

11. The installation wizard reads the configuration file and displays the user data in step 9, as shown in Figure 2-42. Ensure that the listener port, which defaults to 2500, is not already in use in your system. If required, change the port to a port that is not in use. Click **Next**.

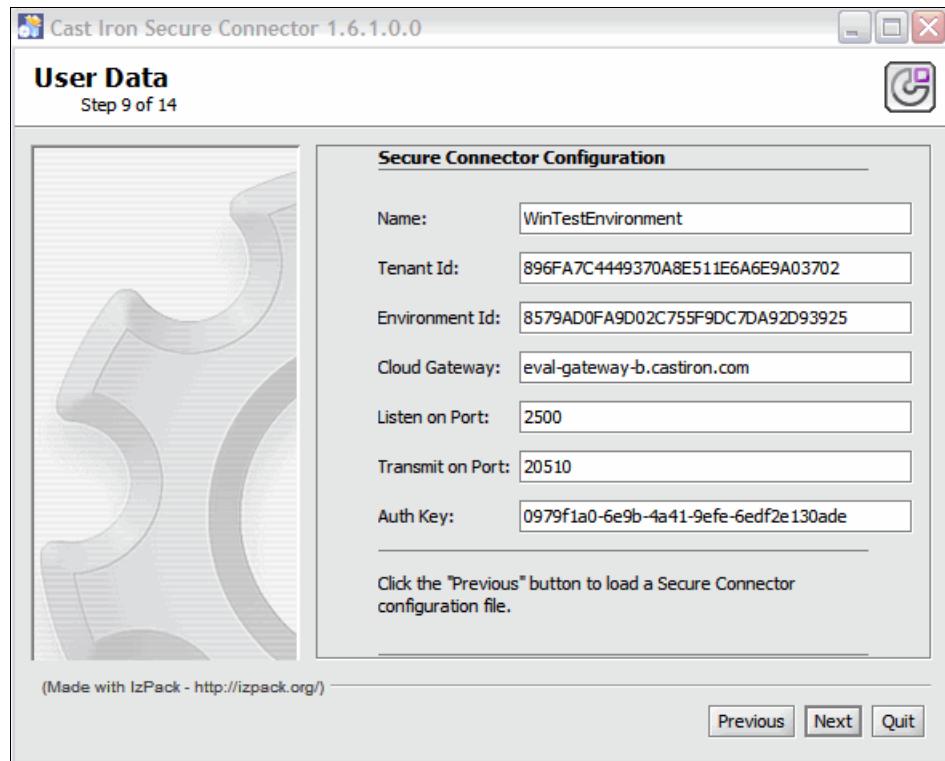


Figure 2-42 Cast Iron Secure Connector installation user data display

12. If your system cannot connect directly to the Internet but is using a proxy, you can specify a proxy, including password, as shown in Figure 2-43. (The environment for this book does not need to specify a proxy.) Click **Next** to continue.

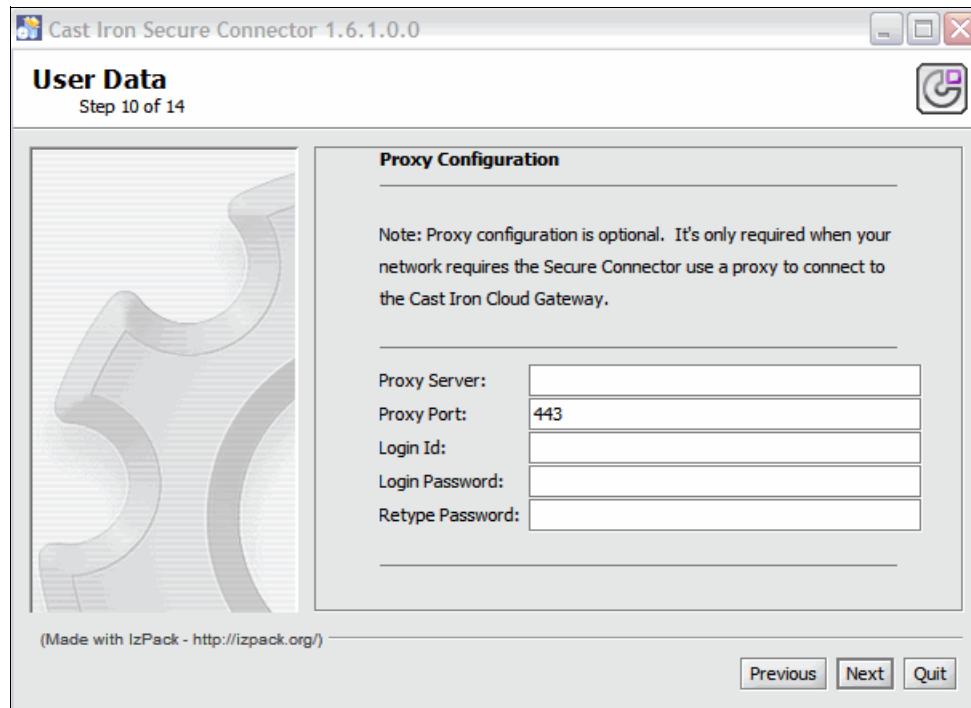


Figure 2-43 Cast Iron Secure Connector installation proxy configuration

13. If you want to run the Secure Connector as a Windows service, specify the details, as shown in Figure 2-44. This scenario starts the service manually, so do not enable this option. Click **Next** to continue.

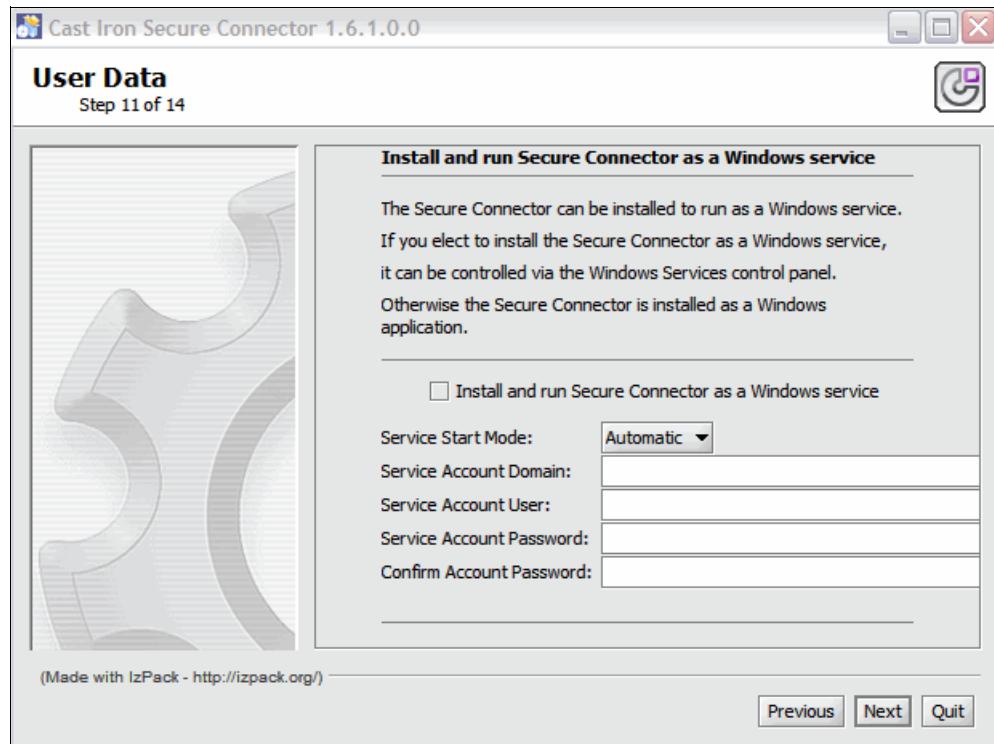


Figure 2-44 Cast Iron Secure Connector installation Windows service configuration

14. The Installation Finished panel displays, Figure 2-45, indicating that the installation finished successfully and that an uninstaller was created. Click **Done** to close the installation wizard.

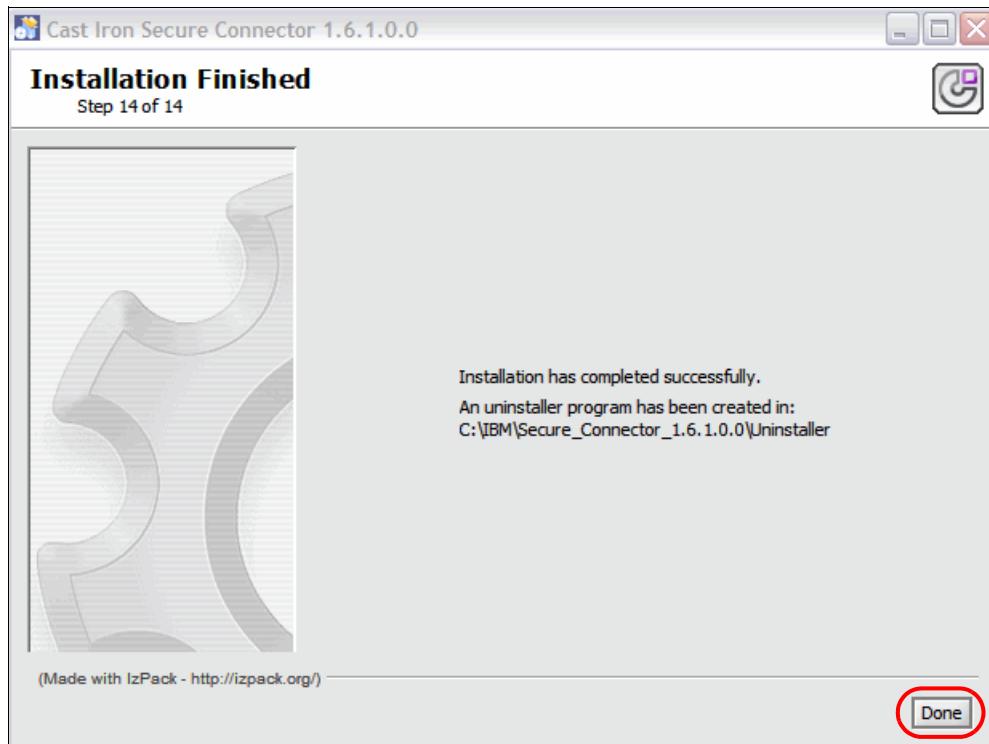


Figure 2-45 Cast Iron Secure Connector installation finished

The installer creates the following shortcuts, as shown in Figure 2-46 on page 72:

- ▶ The Secure Connector Configuration shortcut allows you to change the configuration. A wizard similar to the installation wizard starts, reads the current configuration, and allows you to change it. Use this shortcut also if you want to change the location of the configuration file.
- ▶ The Secure Connector Info shortcut displays the readme file with information about how to install the libraries and other components.
- ▶ The Start Secure Connector shortcut starts the Secure Connector.
You can also start the Secure Connector using the following command:
`C:\IBM\Secure_Connector_1.6.1.0.0\runclient osgi.bat start`
- ▶ The Stop Secure Connector shortcut stops the Secure Connector.
You can also stop the Secure Connector using the following command:
`C:\IBM\Secure_Connector_1.6.1.0.0\runclient osgi.bat stop`
- ▶ The Uninstall Secure Connector shortcut uninstalls the Secure Connector.
You can also uninstall the Secure Connector using the following command:
`C:\IBM\Secure_Connector_1.6.1.0.0\Uninstaller\uninstall.bat`



Figure 2-46 Cast Iron Secure Connector shortcuts

2.6.4 Starting the Secure Connector and verifying communication

To start the Secure Connection and verify communication:

1. Start the Secure Connector using either the shortcut or the command line. If you use the command line, issue the following command:

```
C:\IBM\Secure_Connector_1.6.1.0.0\runclient osgi.bat start
```

Example 2-13 shows the output of this command.

Example 2-13 Output for Secure Connector start on a Windows operating system

```
C:>C:\IBM\Secure_Connector_1.6.1.0.0\runclient osgi.bat start
  "Agent lib home C:\IBM\Secure_Connector_1.6.1.0.0\lib"
  "Picking localconf .\config\localConfig_ws.xml"
  "Picking the resource rules .\config\resourceRules.xml"
  "Flags -Dcom.approuter.agent.debugflag="
  -Dcom.approuter.agent.localconfigfile=
  .\config\localConfig_ws.xml
  -Dcom.approuter.agent.rulefile=.config\resourceRules.xml
  -Dcom.approuter.agent.log4jfile=.config\log4j.properties
  -Dorg.apache.commons.logging.Log=org.apache.commons.logging.impl.Jdk14Logger
  -Dcom.approuter.agent=true -Dcom.approuter.sysconf.agent=true
  -Djavax.net.ssl.keyStore=.etc\security\certs
  -Djavax.net.ssl.trustStore=.etc\security\cacerts
  -Djavax.xml.ws.spi.Provider=com.approuter.module.jws.ProviderImpl
  -Dorg.osgi.framework.bootdelegation=*
  -Dc3p0.debugUnreturnedConnectionStackTraces=true
  -Dc3p0.unreturnedConnectionTimeout=300 -Dc3p0.checkoutTimeout=30000
  -Dc3p0.maxIdleTimeExcessConnections=30 -Dc3p0.maxStatementsPerConnection=100
  -Dc3p0.testConnectionOnCheckin=true -Dc3p0.testConnectionOnCheckout=true
  -Dc3p0.numHelperThreads=10 ""JAVA Optional Flags -Xrs -Xmx768m -Xms512m
  -XX:+UseParallelGC -XX:+UseParallelOldGC -XX:+HeapDumpOnOutOfMemoryError
  -Xdump:stack -Xdump:java:defaults:file=logs\%Y-%m-%d-%H:%M:%S.jvm.pid%pid.dump"
Starting Cast Iron Secure Connector 1.6.1.0.0 (Build 20111011-0038_H3)
```

```
osgi> 2011-10-14 03:47:33 INFO
```

```
*****
*** 
***** STARTING SECURE CONNECTOR at 2011-10-14 03:47:33
***** Time zone is Central European Time or Europe/Berlin
*****
***
```

```
2011-10-14 03:47:38 WARNING Setting the algorithm to IbmX509
```

```

2011-10-14 03:47:42 INFO Finished loading connector modules
2011-10-14 03:47:42 INFO Secure connector name: WinTestEnvironment, Tenant ID:
8
96FA7C4449370A8E511E6A6E9A03702, Environment ID:
8579AD0FA9D02C755F9DC7DA92D93925
2011-10-14 03:47:45 INFO Secure Connector started successfully
2011-10-14 03:47:49 WARNING could not find the object:
com.approuter.assets.AssetPluginFactoryManager@10051005 in any directory!
2011-10-14 03:47:49 WARNING Extension Registry Could not locate the Directory
where the extensions may be located!

```

As the output shows, the Secure Connector started successfully but had some warnings.

- Look at the Cast Iron Live WMC, and check the status of the WinTestEnvironment Secure Connector. If there is no communication problem, the status changes from CREATED to RUNNING, as shown in Figure 2-47.

Secure Connectors			
Secure Connector	Status	Status Last Changed	Status Last Checked
WinTestEnvironment	RUNNING	10/14/2011 03:48:38 AM	10/14/2011 03:53:39 AM

Figure 2-47 Cast Iron Live WMC Secure Connector started successfully

- Use the **netstat -an** command to verify that the 2500 and 20510 ports are used.
- Stop the Secure Connector using the shortcut, and a message displays in the command line window, as shown in Example 2-14.

Example 2-14 Message when stopping the Secure Connector

```

osgi> Cast Iron Secure Connector 1.6.1.0.0 (Build 20111011-0038_H3) has
terminated
C:\IBM\Secure_Connector_1.6.1.0.0>

```

- Use the **netstat -an** command to verify that the 2500 and 20510 ports are released.
- Look again at the Cast Iron Live WMC (go to **System → Secure Connector**). The status of the WinTestEnvironment Secure Connector changed to STOPPED, as shown in Figure 2-48.

Secure Connectors			
Secure Connector	Status	Status Last Changed	Status Last Checked
WinTestEnvironment	STOPPED	10/14/2011 03:59:40 AM	10/14/2011 03:59:40 AM

Figure 2-48 Cast Iron Live WMC Secure Connector stopped successfully

Tip: You might have to click **Refresh** before the status change is displayed.

2.6.5 Installing third-party libraries

To communicate with third-party systems, such as SAP, specific libraries are required. You must install these libraries in WebSphere Cast Iron Live. If you want to communicate with these types of systems using the Secure Connector, you must also install these libraries on the system running the Secure Connector.

To install the libraries in WebSphere Cast Iron Live using the SAP libraries, log in as the environment administrator, and click the Test tab.

Installing the SAP libraries into the cloud

First, install the SAP libraries into the cloud:

1. Click **System → Connector Libraries** to open the Update Connector Libraries page.
2. In the Connector column, select the SAP connector. Verify that the following SAP JCo V3 or later Java libraries are installed:
 - sapjco3.jar and sapidoc3.jar
 - sapjco3.dll (if you use the Secure Connector that is based in a Windows operating system)
 - libsapjco3.so (if you use the Secure Connector that is based in a Linux operating system)

If one or more of these libraries are not installed, click the plus sign (+), and select the library file or files that you need to upload. Refer to Figure 2-49 for help.

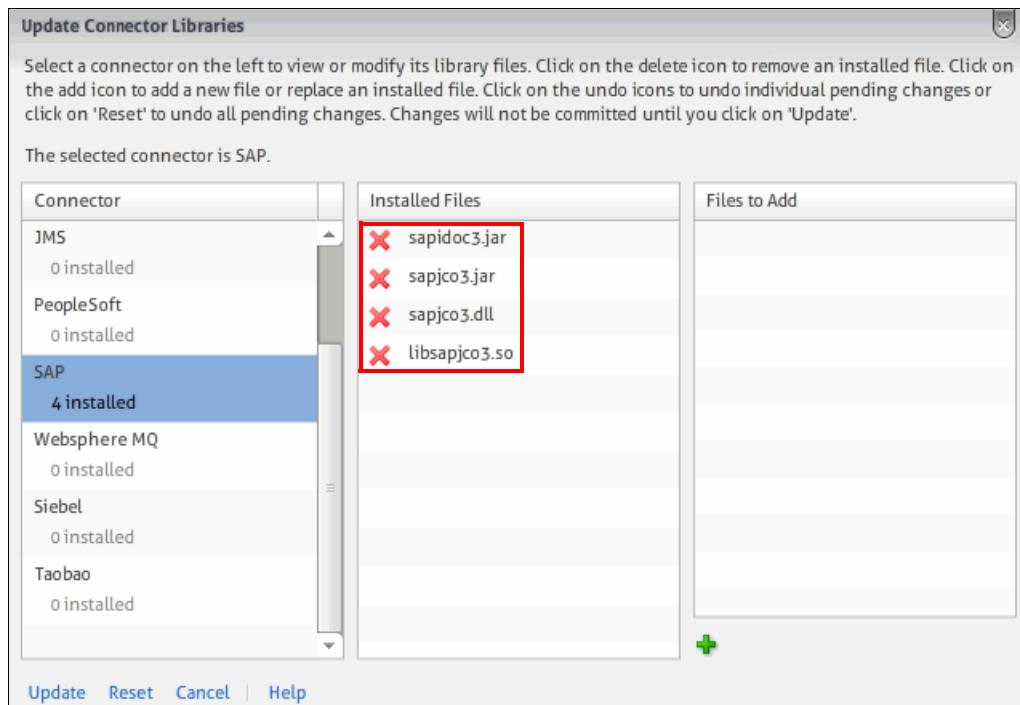


Figure 2-49 Cast Iron Live WMC displays SAP libraries

Downloading the SAP libraries for the Secure Connector

Next, download the SAP libraries for the Secure Connector:

1. Click **System → Secure Connector** to open the Secure Connector page.
2. Click **Download Libraries** to download an archive file containing the libraries. The file name depends on the environment (for the environment Test, it is named SecureConnector-Test-libraries.zip), as shown in Figure 2-50 on page 75.

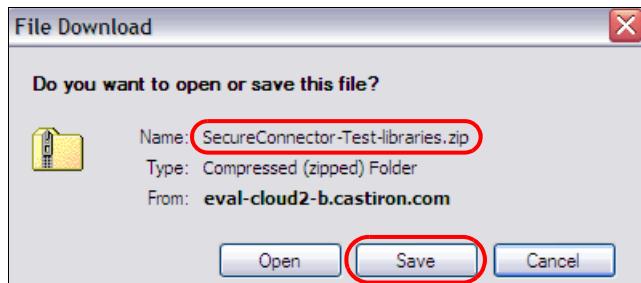


Figure 2-50 Cast Iron Live: Download libraries for the Secure Connector

3. Extract the downloaded libraries.zip file, which contains libraries for all connectors for which libraries were uploaded to WebSphere Cast Iron Live. So, in the environment for this example, with only SAP libraries installed, the .zip file contains only SAP libraries.
4. Use the *.jar files (in this example, com.approuter.module.jcaconnectors.sap.tp-1.0.0.jar) to replace the files in the Secure Connector /lib/plugins installation directory.

As shown in Figure 2-51, the downloaded file is much larger than the original file.

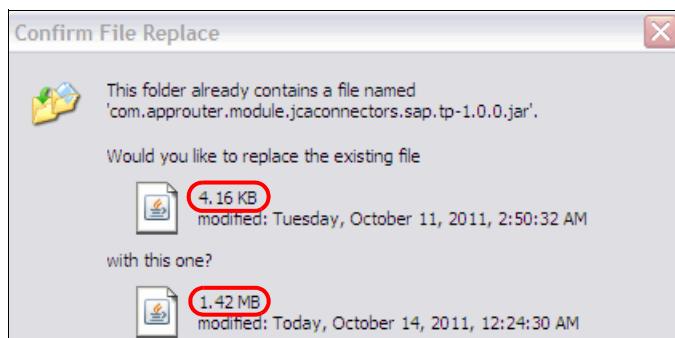


Figure 2-51 Cast Iron Live: Replace SAP connector libraries for the Secure Connector

Tip: If you want to keep the original files in the lib/plugins directory, move them to a different directory or rename them for example, *.jar_old. Otherwise, they are picked up and might cause problems.

5. Copy all the remaining files of the .zip file to the Secure Connector /lib/thirdparty installation directory.
In case of SAP, the remaining files are the sapjco3.dll and libsapjco3.so native libraries, which are located in the the native directory. These files do not exist in the thirdparty directory. (For Windows operating systems, you need only the DLL and not the .so file.)
6. Start the Secure Connector again. The messages is the same as in Example 2-13 on page 72, and in the Cast Iron Live WMC the status of the Secure Connector WinTestEnvironment changes back to RUNNING. (Take a look at step 6 on page 73 for help).

2.6.6 Installing the Secure Connector on Linux

The steps to install the Secure Connector on a Linux operating system are similar to the process for installing on a Windows operating system:

1. Create a Secure Connector in WebSphere Cast Iron Live.

This process is the same as for a Windows operating system, as described in 2.6.1, “Creating a Secure Connector in Cast Iron Live” on page 62, because the definition of the connector is platform independent.

2. Download the configuration and the installer.

This process is the same as described in 2.6.2, “Downloading the Secure Connector configuration and the installer files” on page 63. Specify Linux as the platform. In addition, instead of the Windows operating system executable, use the `linux-secure-connector-installer.sh` shell script.

3. Install the Secure Connector on the Linux operating system.

The installation requires a graphical user interface (GUI) such as GNOME or KDE.

The wizard itself is similar to the wizard for the Windows operating system, as described in 2.6.3, “Installing the Secure Connector on a Windows operating system” on page 64. For further details, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.live.doc%2FSecure_Connector%2FinstallingSecureConnectors.htm

4. Start the Secure Connector.

This process is similar to the process for the Windows operating system. Start a shell script (.sh) instead of a batch file (.bat). For more details, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.live.doc%2FSecure_Connector%2Fstarting_stopping_secure_connectors_linux.htm

5. Download and install the third-party libraries.

This process is the same as the process for Windows operating systems as described in 2.6.5, “Installing third-party libraries” on page 73.

For SAP, copy the .so file and not the DLL file.

2.7 Upgrading from WebSphere Cast Iron Cloud Integration V6.0 to V6.1

This section describes how to upgrade from the WebSphere Cast Iron Cloud Integration V6.0 to the WebSphere Cast Iron Cloud Integration V6.1 and later. Upgrades to V6.0 and later are available either as a fix pack or as a full installation. For more information, go to the IBM Support Portal at:

<http://www-933.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm^WebSphere&product=ibm/WebSphere/WebSphere+Cast+Iron+Cloud+Integration&release>All&platform>All&function=all>

Figure 2-52 shows a list of the fixes that are available at the writing of this book.

1. fixpack: 6.0.0.4-WS-WCI-20110928-1725_H2.studio.exe 6.0.0.4-WS-WCI-20110928-1725_H2.studio.exe	Oct 3, 2011
2. fixpack: 6.0.0.4-WS-WCI-20110928-1717.vcrypt2 6.0.0.4-WS-WCI-20110928-1717.vcrypt2	Oct 3, 2011
3. fixpack: 6.0.0.4-WS-WCI-20110928-1717.scrypt2 6.0.0.4-WS-WCI-20110928-1717.scrypt2	Oct 3, 2011
4. fixpack: 6.0.0.4-WS-WCI-20110928-1717.ova 6.0.0.4-WS-WCI-20110928-1717.ova	Oct 3, 2011
5. fixpack: 6.1.0.1-WS-WCI-20110916-1648_H2.exe 6.1.0.1-WS-WCI-20110916-1648_H2.exe	Sep 19, 2011
6. fixpack: 6.1.0.1-WS-WCI-20110915-1504_H2.vcrypt2 6.1.0.1-WS-WCI-20110915-1504_H2.vcrypt2	Sep 19, 2011
7. interim fix: 6.1.0.1-WS-WCI-20110915-1504_H2.tar 6.1.0.1-WS-WCI-20110915-1504_H2.tar	Sep 19, 2011
8. fixpack: 6.1.0.1-WS-WCI-20110915-1504_H2.scrypt2 6.1.0.1-WS-WCI-20110915-1504_H2.scrypt2	Sep 19, 2011
9. fixpack: 6.1.0.1-WS-WCI-20110915-1504_H2.ova 6.1.0.1-WS-WCI-20110915-1504_H2.ova	Sep 19, 2011

Figure 2-52 Fixes available for WebSphere Cast Iron Cloud Integration

2.7.1 Upgrading WebSphere Cast Iron Studio V6.0

The fix pack for WebSphere Cast Iron Studio is a full installation. To upgrade from Studio V6.0 to V6.1, you only need to install V6.1. Although the binaries are replaced, configuration settings, such as the global configuration properties, are kept and are available after the upgrade.

2.7.2 Upgrading the physical appliance

To upgrade the physical appliance to WebSphere Cast Iron V6.1.0.1, download the 6.1.0.1-WS-WCI-20110915-1504_H2.scrypt2 or later fix pack. This fix pack is an encrypted file to upgrade the WebSphere DataPower Cast Iron appliance from V6.0 and later to V6.1.0.1. Use the WMC or the CLI to apply the fix pack.

During the upgrade, the appliance checks the integrity of the fix pack before applying it. If the newer version supports newer third-party libraries (as it is for SAP), replace the older libraries with newer libraries. For details about the upgrade, refer to 4.9, “Appliance and connector upgrades” on page 227.

If you configured your WebSphere DataPower Cast Iron XH40 appliance for high availability and have the appliances running in active/standby mode, apply the fix pack to the primary appliance, and it is applied automatically to the standby appliance as well.

2.7.3 Upgrading the virtual appliance

To upgrade the virtual appliance to WebSphere Cast Iron V6.1.0.1, download the fix pack 6.1.0.1-WS-WCI-20110915-1504_H2.vcrypt2 or later. This file is an encrypted file to upgrade WebSphere Cast Iron Hypervisor Edition from V6.0 or later to V6.1.0.1. Use the WMC or the CLI to upgrade the appliance.

During the upgrade, the appliance checks the integrity of the fix pack before applying it. If the new version supports newer third-party libraries (as is the case for SAP), replace the older libraries with newer ones. For details about the upgrade, refer to 4.9, “Appliance and connector upgrades” on page 227.

Alternatively, you can download the new version of the WebSphere Cast Iron Hypervisor Edition and deploy it to the VMware hypervisor. Then, export the configuration of your existing appliance and import it into the new one. You can decide to import just the projects, just the user settings, or both. User setting information includes network configuration, users and groups, licenses, job log parameters, log levels, notifications, downtime rules, and passwords. For details about the export and import, refer to 4.2.14, “Importing and exporting repository configuration” on page 198.

2.7.4 Upgrading the cloud

Upgrading WebSphere Cast Iron Live is not done by individual users; instead, it is done by IBM. If the new version supports newer third-party libraries (as is the case for SAP), replace the older libraries with newer ones.

If you use the WebSphere Cast Iron Secure Connector, upgrade the connector and, if required, the third-party libraries as described at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.live.doc%2FSecure_Connector%2Fupgradingsecureconnectors.htm



Developing and unit testing with WebSphere Cast Iron Studio

This chapter introduces the WebSphere Cast Iron Studio development tool (referred to in this book as *Studio*). It describes the development process and discusses the development components used in WebSphere Cast Iron Cloud Integration.

The chapter includes the following topics:

- ▶ Overview of Cast Iron development
- ▶ Overview of WebSphere Cast Iron Studio
- ▶ Projects
- ▶ Orchestrations
- ▶ Connectors and endpoints
- ▶ Activities
- ▶ Using HTTP and web service starter activities with Cast Iron Live
- ▶ Functions and lookup tables
- ▶ Maps
- ▶ Parsing data
- ▶ Exception handling
- ▶ Unit testing
- ▶ Exporting and publishing

3.1 Overview of Cast Iron development

A Cast Iron Integration Solution consists of an *orchestration* that contains a set of *activities* performed in order. Each of these activities performs a predetermined operation. Some activities use external systems. You configure access to these external systems using *endpoints* that can be shared by the orchestrations in a project. Orchestrations are created using a graphical orchestration editor in Studio, which allows you to configure and add activities and endpoints to the orchestration.

Tip: Use consistent naming conventions for all projects, orchestrations, activities, endpoints, and variables. Although there are no recommended naming conventions for Cast Iron, establish early on within your team the naming conventions that you will use and how you enforce those naming conventions.

A *project* contains a number of orchestrations and endpoints. After you build an orchestration, you can test it in Studio. You can also test individual activities. Test an orchestration regularly while it is being written to make sure that it matches the requirements of the integration solution.

When you complete a project, you publish it to a Cast Iron runtime. All orchestrations in a project are published when the project is published. The project is then deployed and started using the Web Management Console (referred to in this book as the WMC).

Figure 3-1 shows this overview.

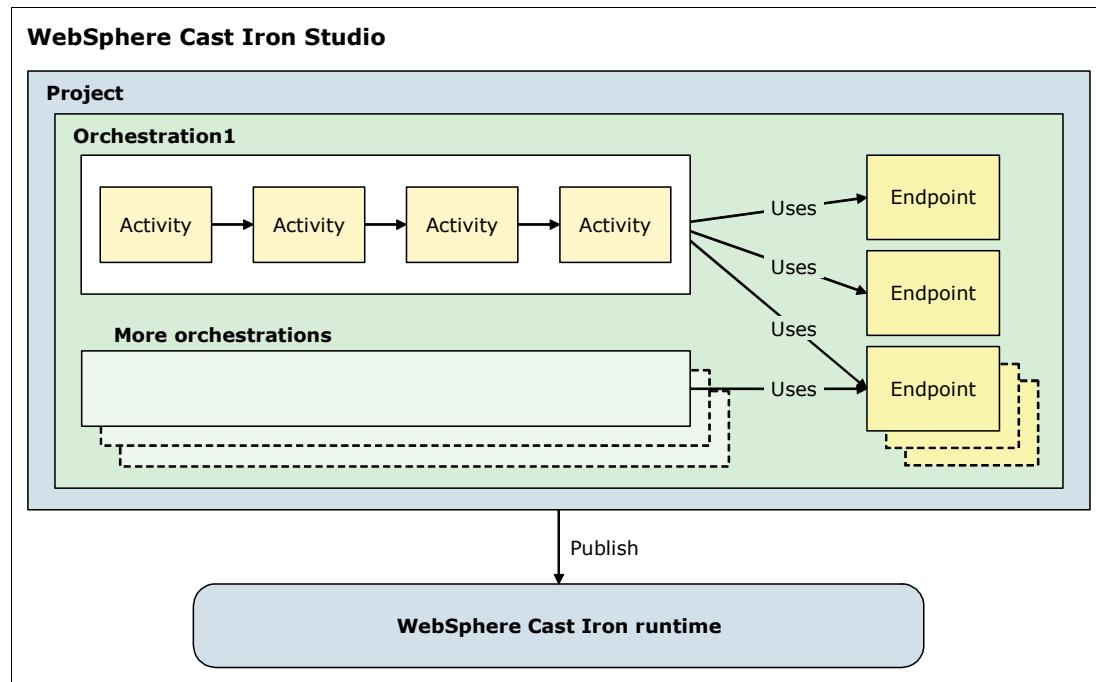


Figure 3-1 An overview of developing with WebSphere Cast Iron Studio

The remainder of this chapter describes the process of developing Cast Iron Integration Solutions using Studio and the components that are involved in this process.

3.2 Overview of WebSphere Cast Iron Studio

WebSphere Cast Iron Cloud Integration provides a development tool called WebSphere Cast Iron Studio, which we refer to in this book as *Studio*. Use this tool to develop for both Cast Iron Live and Cast Iron Integration Appliances in the same manner. The few differences are described in the sections in this chapter where they apply.

For Cast Iron Live, Studio is a dynamically downloaded Java application that is started from the modify tab inside Cast Iron Live. Click the pencil icon under Actions, and then select **Edit Project in Designer**, as shown in Figure 3-2.

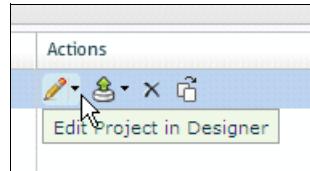


Figure 3-2 Launching Studio from the Web Management Console (WMC) of Cast Iron Live

Tip for Cast Iron Live: If the WMC that launches Studio times out because of inactivity, you cannot save the project that you are developing in Studio back to Cast Iron Live. If this situation occurs, save the project locally, and then close Studio. Log in to the WMC again, and restart Studio. Finally reopen the local project, and save it back to Cast Iron Live.

To avoid Cast Iron Live timing out, change the Inactivity Timeout from the WMC Settings pane, which is described in 4.1.2, “Web Management Console” on page 175.

For Cast Iron Integration Appliances, Studio is a local application that must be installed and run on a Windows operating system.

The installation process for Studio when using Cast Iron Integration Appliances and the prerequisites for using either version of Studio are described in Chapter 2, “Installing and setting up WebSphere Cast Iron integration” on page 27.

When you launch Studio for the first time, it starts with the introduction pane shown in Figure 3-3 on page 82. The menus and toolbar icons are covered later in this chapter, and projects are covered fully in 3.3, “Projects” on page 84.

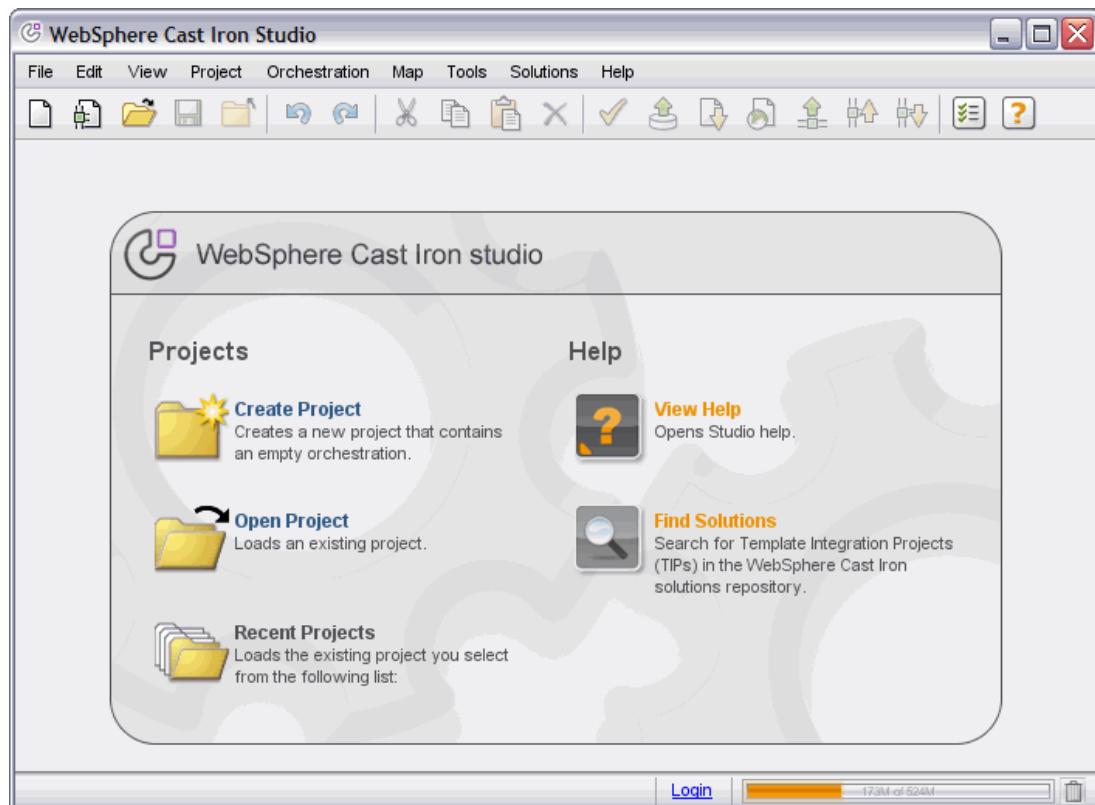


Figure 3-3 Studio introduction pane

The moving orange bar at the bottom right of the window, Figure 3-3, indicates the amount of memory that Studio is currently using. To the right of the bar is a trash can icon. Clicking the trash can icon requests that Studio run garbage collection to attempt to reduce the amount of memory currently being used by Studio.

You can hide the bar and trash can icon by selecting **Edit → Preferences → Project** and clearing the option. You can also set Studio to reopen the last project when Studio starts using the same preference window.

Studio preferences include a number of other useful settings, all of which are fully documented in the information center:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.doc%2Fbasis_projectpreferences.html

You can start a web browser through Studio to go to the information center website using the Help menu. From the Help menu, you can also start a web browser to gain access to the WebSphere Cast Iron Community. This community provides support for Cast Iron development. The community also gives access to the Cast Iron Solutions Repository, which is discussed in Chapter 7, “Reusability with Template Integration Projects” on page 293, or you can access this repository directly in Studio using the Solutions menu.

Figure 3-4 on page 83 shows an open project in Studio. The Studio main window includes the following sections, which are highlighted in Figure 3-4 on page 83:

Workspace The area for creating and editing orchestrations

Configuration Pane The area to set properties for the orchestration and its activities as well as to performing mapping

Toolbox

A set of features that can be used by an orchestration

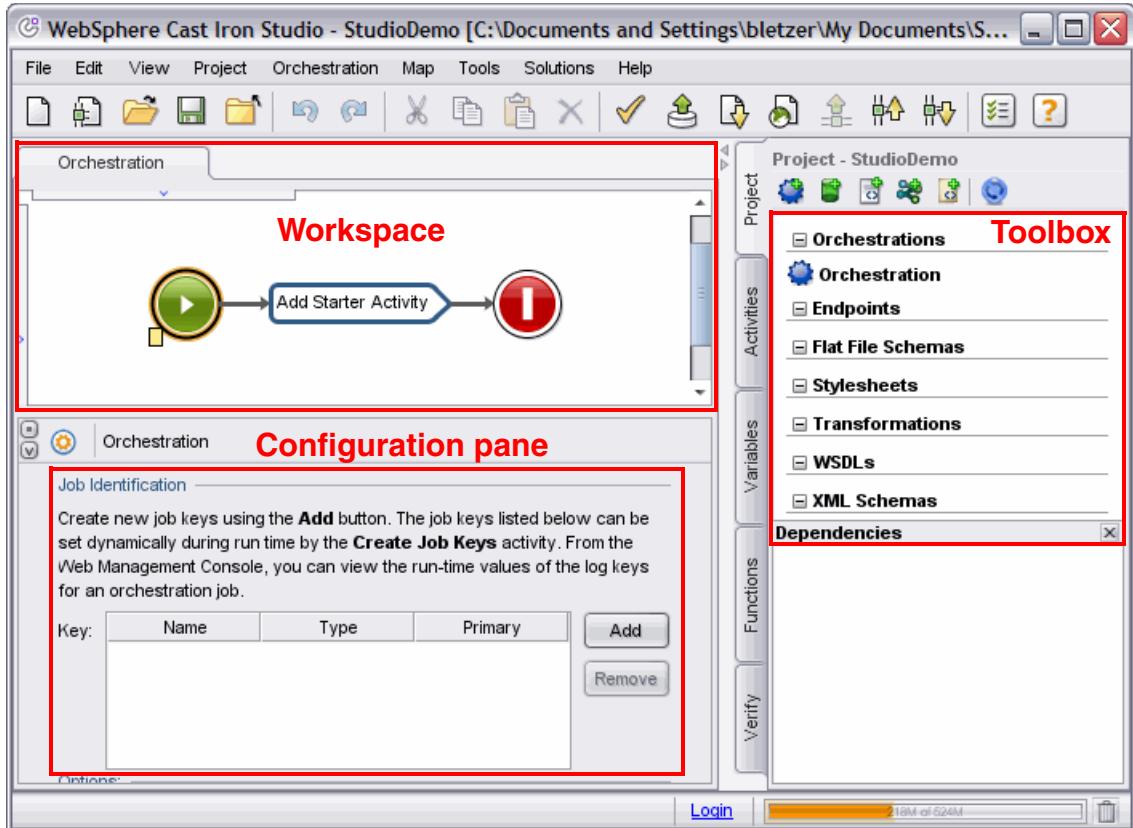


Figure 3-4 Studio sections

Multiple orchestrations, endpoints, and other project entities can be opened in the workspace simultaneously. Each opened entity has an entity tab at the top of the workspace with the name of the entity displayed on it, as shown in Figure 3-5. Click an entity tab in the workspace to select that entity. Click the cross on an entity tab to close that entity.

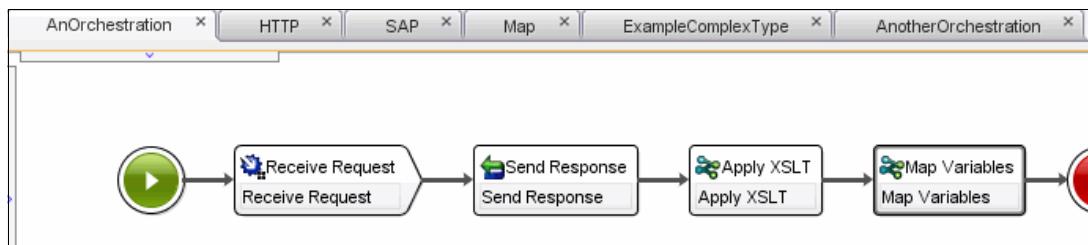


Figure 3-5 Examples of entity tabs

The Toolbox section has the following tabs, as shown in Figure 3-6 on page 84:

Project

Selected in Figure 3-4 and discussed in 3.3.1, “Project toolbox tab” on page 87. This tab provides access to orchestrations, endpoints, flat file schemas, stylesheets, transformations, WSDL files, and XML schemas.

Activities

Selected in Figure 3-6 on page 84 and discussed in 3.6, “Activities” on page 107. This tab provides access to the activities you can use to create an orchestration.

Variables	Discussed in 3.7, “Variables” on page 117. This tab allows you to create and configure the variables that hold data in the orchestration.
Functions	Discussed in 3.8, “Functions and lookup tables” on page 123. Provides access to built-in and custom functions that can be used to alter data in mapping tables.
Verify	Discussed in 3.12, “Unit testing” on page 159. This tab provides a venue for testing orchestrations.

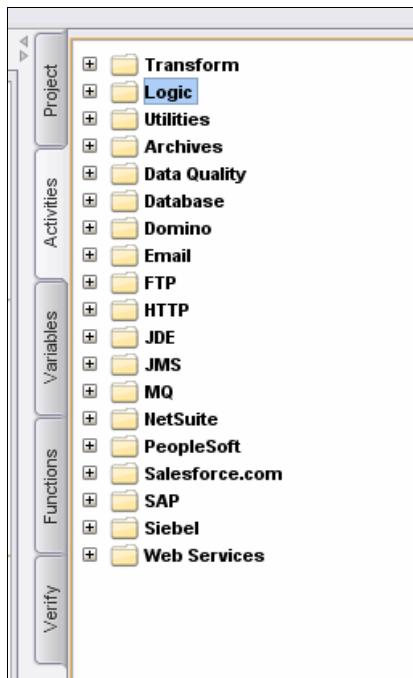


Figure 3-6 Toolbox tabs

3.3 Projects

A Cast Iron Integration Solution is created in an integration project. In addition, there are also connector projects, which are discussed in Chapter 8, “Building custom plugin connectors” on page 321. *Projects* are a container for the components of an integration solution, such as orchestrations, endpoints, and transformations. You must create a project before you can create an integration solution, and all integration components must be held inside a project.

To create a project inside Studio, select **File** → **New** → **New Project**, or click the New Project toolbar icon shown in Figure 3-7. Other toolbar icons are available in Studio. A description of each toolbar icon displays when you move the mouse pointer over the icon.

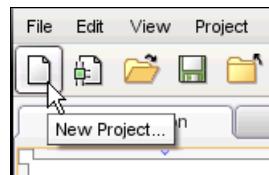


Figure 3-7 The New Project toolbar icon

Enter a name for the project and select a directory in which to store the project files. For a description of valid names, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.doc%2Fref_About_Valid_Names.html

For Cast Iron Live, you can create a project in the WMC using the Create tab. You can search for a project template in the Cast Iron Solutions Repository or you can select the **create one from scratch** link, as shown in Figure 3-8.

Name	Source	Target	Certified
T0060 - SAP Materials To NetSuite Inventory Items	SAP	NetSuite	
T0091 - SAP Contact To NetSuite Contact	SAP	NetSuite	
T0090 - SAP Customers To NetSuite Customers	SAP	NetSuite	

Figure 3-8 Searching for template projects in Cast Iron Live or creating a project from scratch

Tip: Because multiple projects can be published by multiple people to one Cast Iron runtime, it is important to give the project a clear, descriptive, and unique name. A project replaces another project of the same name and version if they are both published to the same Cast Iron runtime. Different versions of the same project can coexist within a Cast Iron runtime

To open an existing project inside Studio, select **File → Open Project**, and then select a Cast Iron project properties file from the local file system. The file has an extension of .sp3.

To open an existing project from Cast Iron Live, select the Modify tab, click the pencil icon next to the project that you want to open, and then select **Edit Project in Designer**, as shown in Figure 3-2 on page 81. If the Cast Iron Live project was imported using the WMC and was not created in Cast Iron Live, the project does not appear on the Modify tab. To copy the source code for this published project to the Modify tab, select the environment tab where the project was published, and click the **Create Source Project** icon shown in Figure 3-9 on page 86 for the project that you want to create.

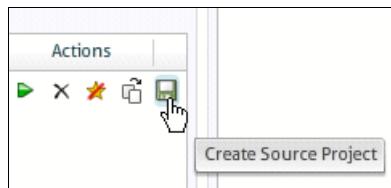


Figure 3-9 Creating a source project in Cast Iron Live

You can also delete a project from Cast Iron Live using the Modify tab and the cross icon, as shown in Figure 3-10.

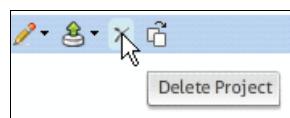


Figure 3-10 Deleting a project in Cast Iron Live

Studio opens only one project at a time. If a project is currently open in Studio and a new project is created or an existing project is opened, Studio closes the currently open project. If the currently open project was not saved, Studio prompts you to save changes before it closes.

Save a project at any time using the **File** menu. For Cast Iron Live, you can save a project either to the local machine or to Cast Iron Live. The project must be stored to Cast Iron Live before it is published because you cannot publish to the Cast Iron Live environment directly from Studio. You can also rename a project by saving it with a different name, which creates a copy of the project with the new name, allowing you to use it as the starting point for a new integration solution.

To promote an integration solution on to a Cast Iron runtime, the project that contains the solution is published to the runtime. All components inside the project are published with the project. The project then becomes a *project configuration* on the runtime. This project configuration holds the properties associated with the entities in the project. These properties can be changed in the WMC. This is described in 3.3.2, “Configuration properties” on page 88.

When the project configuration starts on the runtime all enabled orchestrations inside it are started. Publishing a project is described in 3.13, “Exporting and publishing” on page 170, and starting a project configuration is discussed in 4.2.2, “Starting a configuration” on page 181.

Tip: All orchestrations inside a project configuration are stopped if the project configuration stops. Therefore it is good practice to only place orchestrations that are directly related to the same integration solution, and all need to be stopped if any are stopped in to the same project.

3.3.1 Project toolbox tab

When a project is opened in Studio, the Project toolbox tab shows the contents of the project, as shown in Figure 3-11.

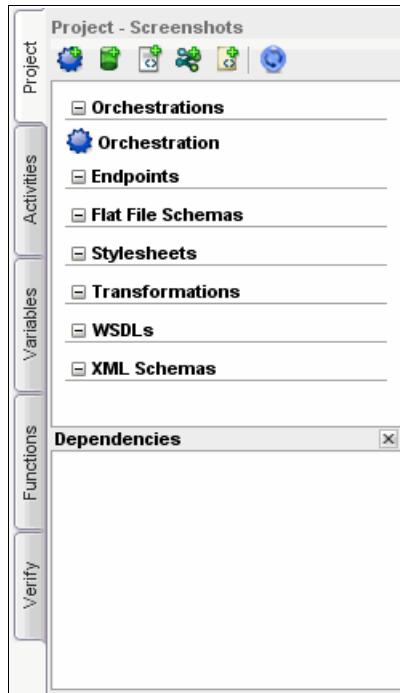


Figure 3-11 The Project toolbox tab

Figure 3-12 shows a view of the icons at the top of this tab. The four icons at the top left of the tab allow the creation of new orchestrations (1), endpoints (2), flat file schemas (3), and standalone maps (or transformations) (4) respectively. The fifth icon is Add Document (5) and gives options to import flat file schemas, WSDLs, XML schemas, and stylesheets (XSLTs) into the project. Finally, the right icon (6) revalidates the project. Project validation is discussed in 3.12, “Unit testing” on page 159.



Figure 3-12 A magnified view of the Project toolbox tab icons

The bottom of the Project toolbox tab contains a Dependencies pane. When a component is selected in the Project toolbox tab this dependencies pane shows any other components that are dependent on the selected component.

The items in the Project toolbox are discussed in detail later. Orchestrations are discussed in 3.4, “Orchestrations” on page 95. Endpoints are discussed in 3.5, “Connectors and endpoints” on page 102. Flat File Schemas and XML Schemas are covered in 3.10, “Parsing data” on page 149. Stylesheets and Transformations are covered in 3.9, “Maps” on page 128. Finally, WSDLs are used in 3.6, “Activities” on page 107.

3.3.2 Configuration properties

When a project is published to a Cast Iron runtime it becomes a project configuration. This project configuration contains all of the components of the integration solution and holds values for properties needed by the components of the integration solution such as:

- ▶ The name of a database used by a Database endpoint
- ▶ The URL used by an HTTP Receive Request activity
- ▶ The subject of an email sent from a Send Email activity

The values for these properties can be hard coded in to the project configuration or can be held inside *configuration properties* for the project configuration. The configuration properties contain values that are used by components in an integration solution, which can be changed in the WMC without using Studio or republishing the project. Thus the settings of the integration solution can be altered by an administrator as the need arises. For example, if the password used to connect to Salesforce.com changes, or a database is moved to another server, these changes can be made using the WMC.

It is even possible to create a clone of the project configuration for the integration solution after it is published to a Cast Iron runtime. The WMC is used to clone a project configuration. This clone can have several values for its configuration properties that allows the clone to perform differently to the original project configuration, as shown in Figure 3-13.

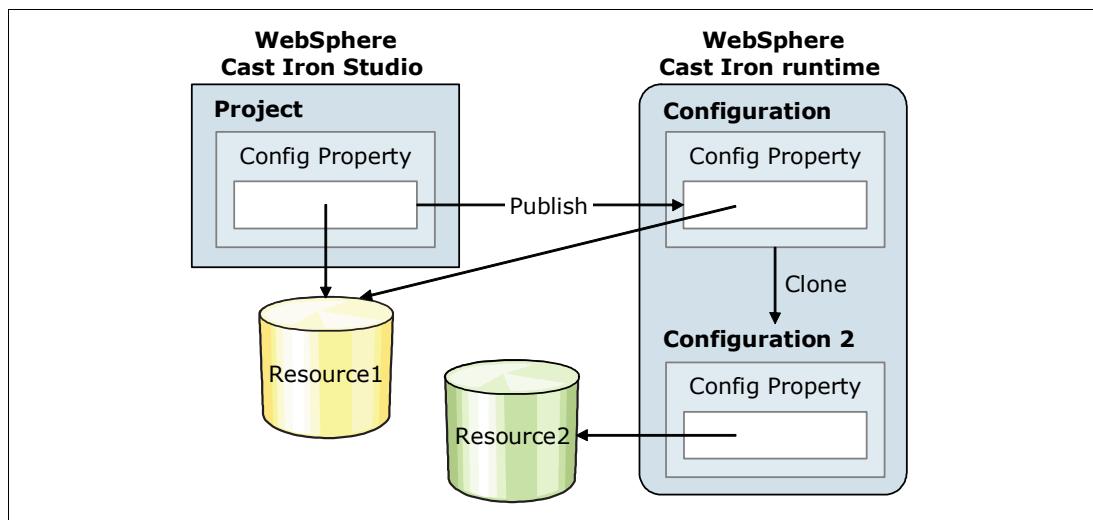


Figure 3-13 A cloned project configuration with different values for its configuration properties

Changing configuration properties is described in 4.2.6, “Changing configuration properties” on page 185. Cloning a project configuration is described in 4.2.8, “Cloning configurations” on page 188.

Tip: It is recommended that configuration properties are used for all endpoint values. Establish a consistent naming convention for configuration properties that avoids abbreviations in the property name to avoid confusion. Configuration properties are listed alphabetically in the WMC so start settings for each endpoint with the same string to group them together, for example, DB2Password, DB2Username, and so on.

Creating configuration properties

Configuration properties are created in Studio. The properties are given default values in Studio, but you can change these properties at runtime in the WMC. You can create

configuration properties using the **Project → Configuration Properties** menu to open the window shown in Figure 3-14. Each property is given a name, data type, and default value.



Figure 3-14 The configuration properties window

The property can then be used in any field in a component that has the T drop-down icon next to it. The example in Figure 3-15 shows the User Name and Password fields of a Salesforce.com endpoint. Both of these fields have the T drop-down icon next to them and thus can use a configuration property.

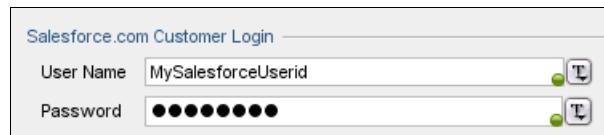


Figure 3-15 Salesforce.com fields that can use configuration properties

A configuration property can also be created directly from a field with the T drop-down icon next to it. In Figure 3-15, there is a green dot at the end of the field. This dot indicates that the value in the field can be turned into a configuration property. If you click the green dot, a dialog box is displayed that asks for the name of the property, as shown in Figure 3-16.

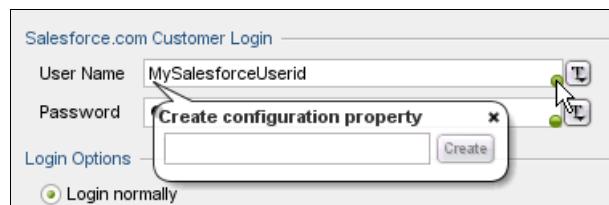


Figure 3-16 Creating a configuration property by pressing the green dot

When you select the configuration property for a field, the name of the property is displayed in the field and not the value of that property. The name of the field is in italics to indicate that it is a configuration property, as shown in Figure 3-17 on page 90.



Figure 3-17 The User Name and Password fields are set using configuration properties

Global configuration properties

Collections of configuration properties can be defined in Studio so that they can be imported in to any project that is being developed in that copy of Studio. These properties are called Global Configuration Properties. To create them, select **Edit → Manage Global Configuration Properties**. The Global Configuration Properties window shown in Figure 3-18 opens.

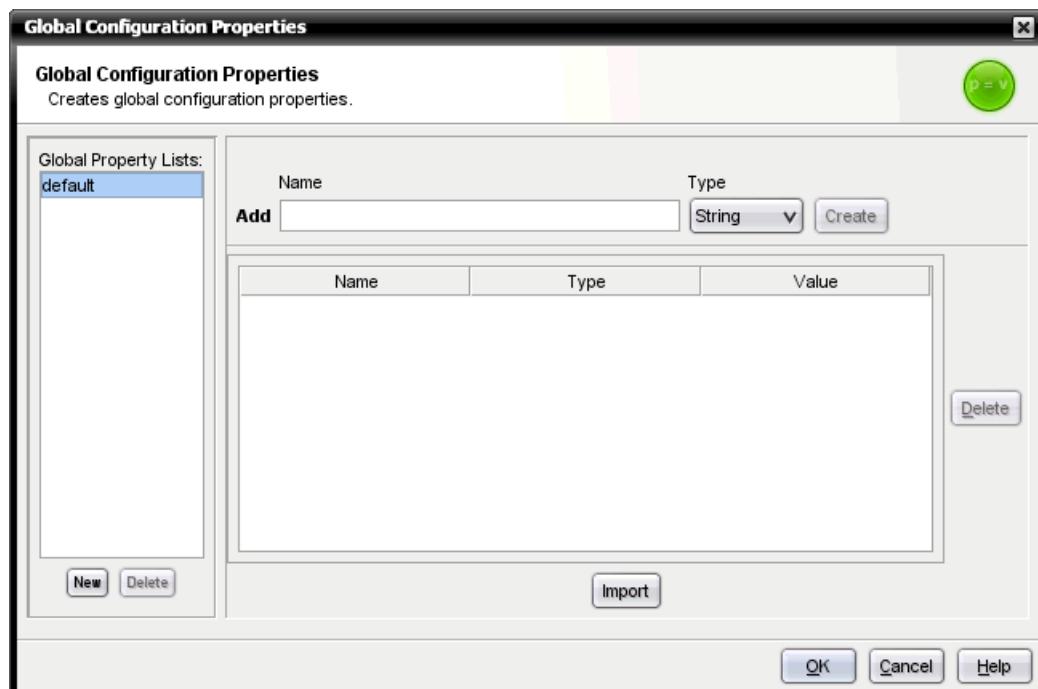


Figure 3-18 The Global Configuration Properties window

Global Configuration Properties are created in Global Property Lists. The default list is created automatically. To create a new list, click **New** at the bottom left of the Global Configuration Properties window.

To create a new property:

1. Select the Global Property List to which you want to add the property.
2. Type the name of the property in to the Add field, and click **Create**.
3. Set the data type and value for the property, and click **OK**. The window closes.

To use a Global Configuration Property in a project, click **Import** on the project Configuration Properties window shown in Figure 3-14 on page 89. The Import Properties window shown in Figure 3-19 on page 91 opens.

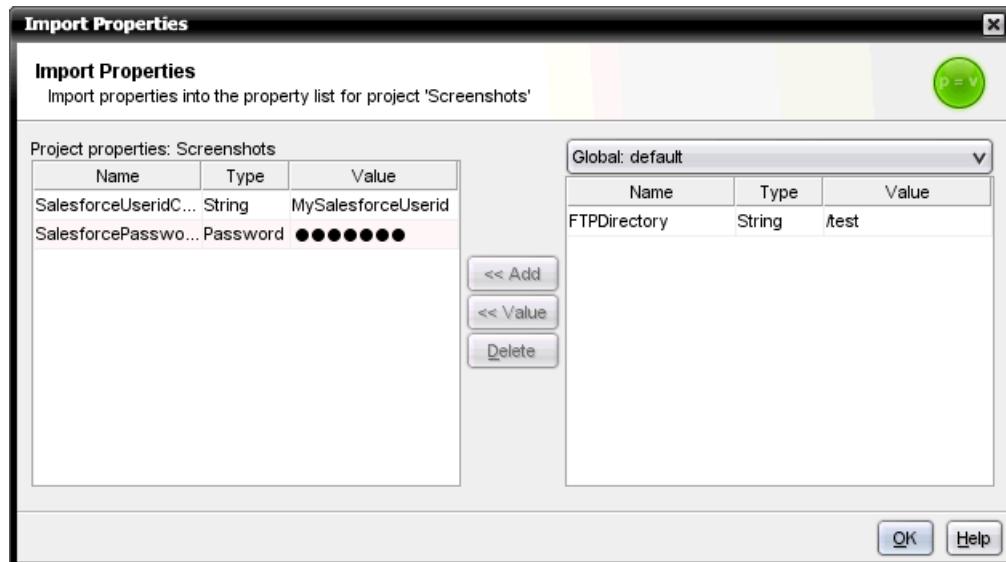


Figure 3-19 The Import Properties window to import Global Configuration Properties

Select the Global Property List that contains the property you want to import from the drop-down list on the right pane of the window. Then select the property that you want to import, and click **Add**. If a property of the same name already exists in the project, a dialog window opens asking whether you want to replace the existing configuration property.

You can copy just the value of a Global Configuration Property to an existing project Configuration Property of the same data type by selecting both the Global Configuration Property and the project Configuration Property and clicking **Value**.

Project Configuration Properties can be imported in to a Global Property List by clicking **Import** on the Global Configuration Properties window shown in Figure 3-18 on page 90 and following the same steps that you followed to import Global Configuration Properties.

3.3.3 Project directory structure

Projects on the local machine are stored in a directory containing a project properties file with an extension of .sp3, some metadata files, and subdirectories for the components in the project. This directory holds all of the files that the project needs and can be backed up or moved to another machine if necessary. Direct editing of any of the files in this directory is not recommended. Use Studio for all editing of integration solution files.

Start Studio and open a project by double-clicking the .sp3 file in the project directory. This .sp3 file contains the name of the project, the version of the project, and the version of Studio that created the project.

Projects created in Cast Iron Live are stored within Cast Iron Live and, therefore, have no directory structure on the local machine.

WebSphere Cast Iron Studio Version 6.1 can open projects created in earlier versions of Studio. It is recommended that projects created in earlier versions are opened in Studio before being published to a Cast Iron Version 6.1 runtime.

3.3.4 Versioning projects

Studio does not contain any development version control functionality. However, the Cast Iron runtime does recognize different versions of a project configuration and allows different versions to be published simultaneously.

Set a version number for a project in Studio by clicking **Project** → **Project Settings**, as shown in Figure 3-20. You can also store a description and comments with the project.

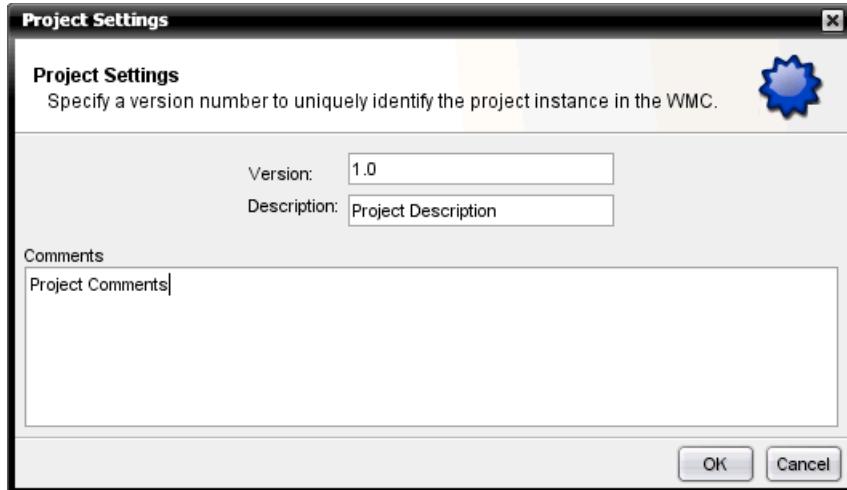


Figure 3-20 The Project Settings window

The version number is a string property stored with the project. A copy of the project is not created when the version number is changed. If you need to keep previous versions of a local project, click **File** → **Save As** to store the old version as another name. Alternatively, the whole of the project directory or an export of the project can be stored in an external version control repository.

Two or more versions of a project can be published, deployed, and started on the same Cast Iron runtime at the same time. Project configurations for all versions of the project are created. Figure 3-21 on page 93 shows an example of why two versions of a project might be running at the same time. A number of departments in a company are using one integration solution to move data from a database to Salesforce.com. New columns were added to the database to support new requirements for the company. The validation and mapping within the orchestration must be changed to reflect the additional fields. However, the new columns are only being introduced to departments one at a time. Therefore it is necessary to keep both versions of the orchestration running for a period of time.

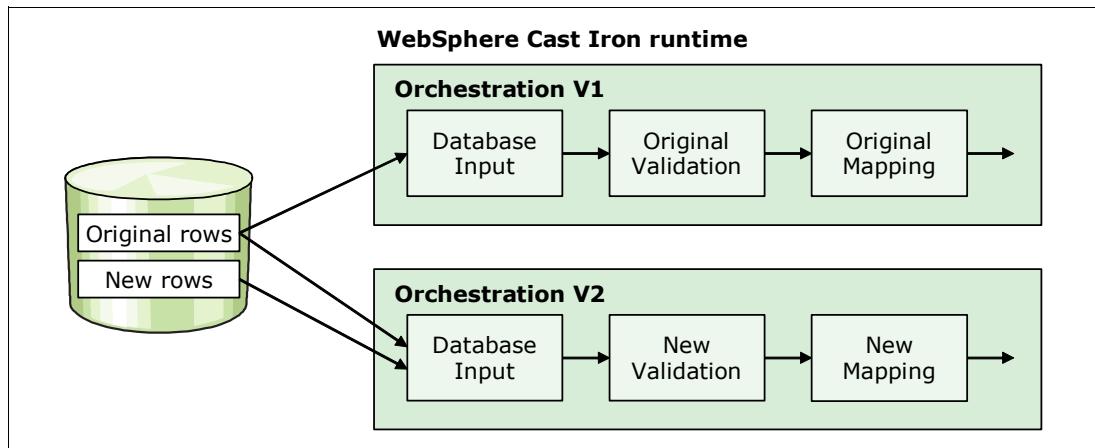


Figure 3-21 Two versions of an orchestration running at the same time

Tip: Publishing a new version of a project while the old version is still running can make it quicker to move from one version to another in the WMC. After the new version of the project is published and configured, you can stop the old version and start the new version quickly to take its place.

When a project is published to a Cast Iron runtime the name of the project configuration created uses the version number of the project. Thus, if a project is published more than once with different version numbers, multiple separate project configurations are created, as shown in Figure 3-22. Notice that version 1.0 of a project does not show a version number.

Project Configurations						
Configuration	Running	Completed	Errored	Total	Actions	
▶ Screenshots (Undeployed)	0	0	0	0	0	▶ X ✘
▶ Screenshots {1.1} (Undeployed)	0	0	0	0	0	▶ X ✘
▶ Screenshots {1.2} (Undeployed)	0	0	0	0	0	▶ X ✘

Figure 3-22 Multiple versions of a project published to the same Cast Iron runtime

For projects created in Cast Iron Live, it is possible to create a new version of the project from the Modify tab by clicking **Create New Version**, as shown in Figure 3-23.



Figure 3-23 Creating a new version of a project in Cast Iron Live

3.3.5 Project permissions

You can protect projects by giving them a password. To set a password in Studio, click **Project → Permissions → Protect**. Next, enter a password in the dialog box shown in Figure 3-24.

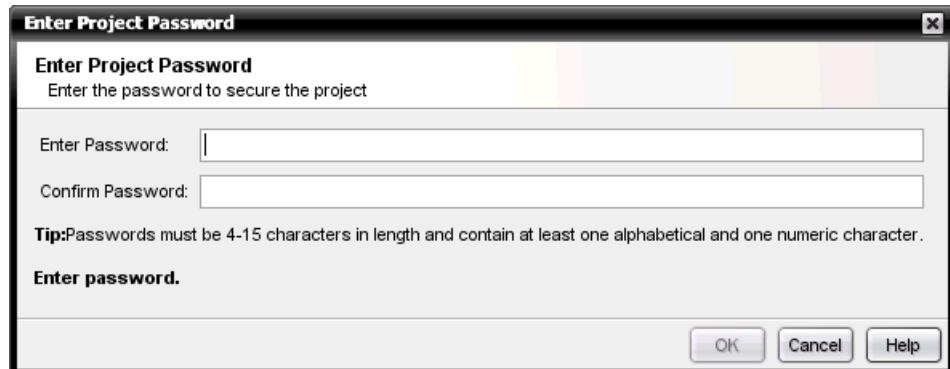


Figure 3-24 The protect project window

A project that is protected cannot be opened in Studio without entering the password. You can change the password and the project can be unprotected using the Project menu.

The password is held only inside the project. When the project is published to a Cast Iron runtime, the password is not included in the project configuration and has no effect on the integration solution.

3.3.6 Project documentation

Studio can produce an HTML file that contains documentation for a project. To produce an HTML file, click **File → Generate Project Documentation**. The file is named using the project name, with a suffix of Doc.htm1. The file lists details of the following items, if they are included in the project:

- ▶ Configuration Properties
- ▶ Endpoints
- ▶ Orchestrations
- ▶ Flat File Schemas
- ▶ Stylesheets
- ▶ Standalone Maps
- ▶ XML Schemas
- ▶ WSDLs

Tip: Any passwords documented for the Configuration Properties and Endpoints are masked.

The documentation for an orchestration in the project documentation HTML file gives a summary of the structure of each orchestration and the activities contained in the orchestration, as shown in Figure 3-25 on page 95. Each activity is documented using the Activity Name field set for the activity.

FTP_DB2_to_Domino_sync_Orders	
details	
<i>Number of Activities:</i> 13	
comments	
Any comments for the whole orchestration display here.	
activities	
Read Orders File over FTP (FTP) This activity polls the selected FTP endpoint for the specified file(s)	
comments	
If a comment has been added for an activity it displays next to the activity here.	
Parse Orders File (Transform) Parse Flat File	
Transform Orders to List of Items (Transform) Custom XSLT	
Try (Logic) Try	
Lookup Product Data in Database (Data Quality) This activity gets rows of data from the selected database endpoint based on a given SQL select query	
Merge Good Data and Error Data (Data Quality) This activity merges two sorted inputs	
Loop through Items (Logic) ForEach Activity	
Create Invoice Documents in Domino (Domino) This activity creates new documents on the Domino server	
CatchAll (Logic) Catch All Activity	
Log Error (Web Services) This activity invokes a web service on the selected Web Service endpoint	
Record File Name and Total Orders (Utilities) Log Keys Activity	
CatchAll (Logic) Catch All Activity	
Log Error (Web Services) This activity invokes a web service on the selected Web Service endpoint	

Figure 3-25 The orchestration summary contained inside a project documentation file

Orchestration and activity comments and saving an image of an orchestration are described in 3.4.4, “Orchestration documentation” on page 100.

An XML file is also created that contains the settings for all components of the project, including configuration properties.

3.4 Orchestrations

Orchestrations are the main component of a Cast Iron Integration Solution. All functionality in an integration solution is controlled by orchestrations. All projects must have at least one orchestration, but projects can contain more than one orchestration.

Take care when designing orchestrations. Poorly designed orchestrations can be difficult to maintain, and if an orchestration is designed badly, it can take a long time to complete execution or it does not complete at all. Additionally, the Cast Iron runtime executes multiple orchestrations simultaneously. Thus, a badly designed orchestration can slow down other orchestrations running on the same Cast Iron runtime.

You can create orchestrations from either the Project menu or the Project toolbox tab by clicking **New Orchestration**. The new orchestration is given a default name that you must change to a name that is unique within the project and that describes the functionality of the orchestration.

For a description of valid orchestration names, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.doc%2Fref_About_Valid_Names.html

Orchestrations contain a set of activities that are performed in an order that is defined within the orchestration editor or workspace. Activities are discussed in 3.6, “Activities” on page 107.

3.4.1 The orchestration editor

To open an orchestration in the orchestration editor or workspace, double-click the orchestration in the orchestrations section of the Project toolbox tab. The workspace opens and shows the current content of the orchestration. When a new orchestration is created, that orchestration automatically opens in the workspace. A new orchestration has no activities or endpoints and displays in the workspace, as shown in Figure 3-26.

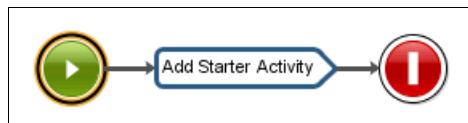


Figure 3-26 A new orchestration

Below the workspace is the configuration pane. Property values are set in this pane to control the behavior of an orchestration and its components. When a new orchestration is opened in the workspace the orchestration properties are shown in the configuration pane.

Orchestration properties are described in 3.4.3, “Orchestration properties” on page 99.

The workspace has three sections. The middle section shows the set of activities in the orchestration and the order in which they are performed. The orchestration starts from the green play icon and ends at the red stop icon. These icons indicate only the start and end of the orchestration and do not contain any functionality. The top and bottom sections of the workspace show any endpoints that are used by activities in the orchestration. Inbound activities are linked to endpoints in the top section and outbound activities are linked to endpoints in the bottom section.

Figure 3-27 shows an orchestration with some activities and endpoints.

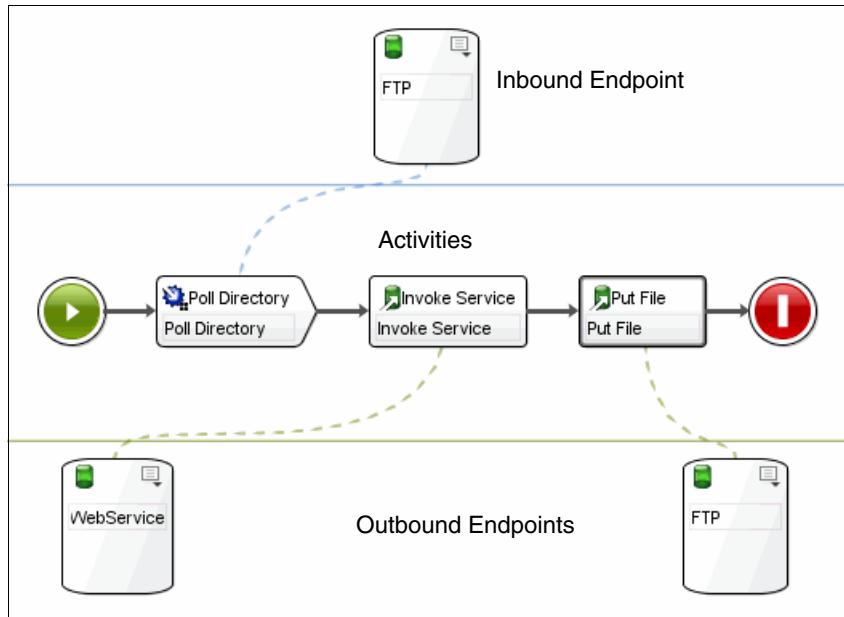


Figure 3-27 The workspace for an orchestration with activities and endpoints

Tip: If an endpoint is used as both inbound and outbound, it can be displayed in both the top section and the bottom section of the workspace. In Figure 3-27, the FTP endpoint is used to both get files and to put files using FTP.

You drag activities from the Activities toolbox tab and place them on the orchestration at the point where they will be performed, as shown in Figure 3-28.

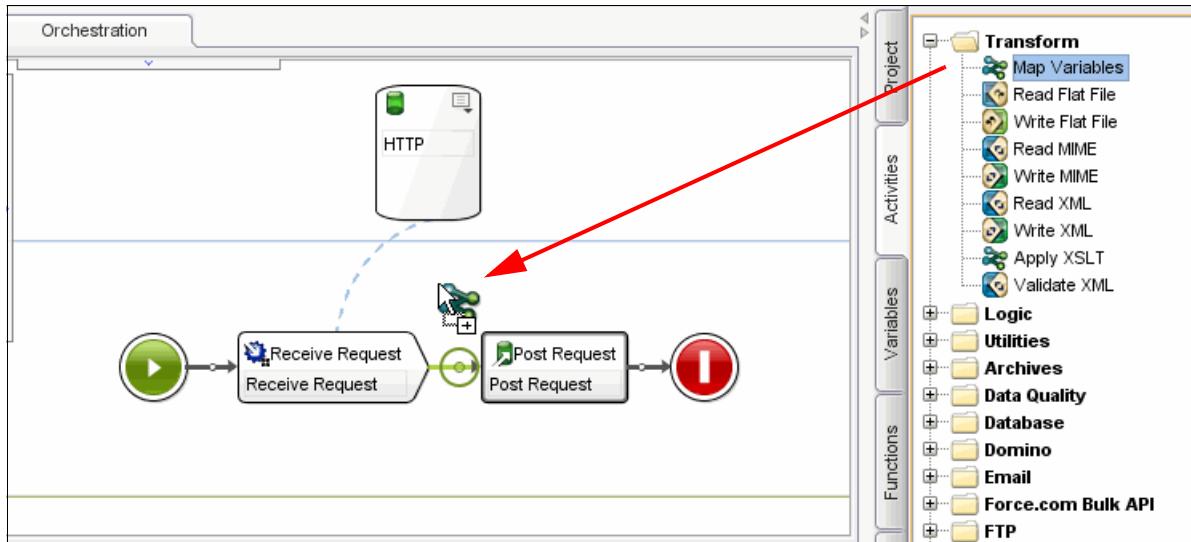


Figure 3-28 Adding a Map Variables activity to the middle of an orchestration

You can reorder activities at any time by dragging the existing activity in the orchestration to its new position in the orchestration. You can delete activities by right-clicking the activity and then selecting delete.

You control the appearance of the workspace using the orchestration view icons at the top left of the workspace, as shown in Figure 3-29.

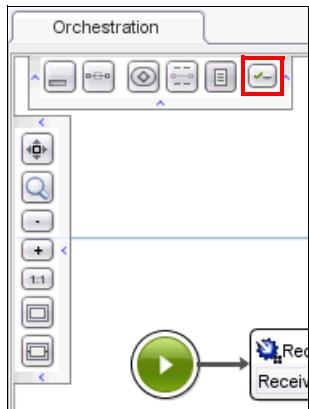


Figure 3-29 The orchestration view icons

The top right icon highlighted in Figure 3-29 provides orchestration layout preferences. Clicking the icon opens the window shown in Figure 3-30, which you can use to set preferences on the layout of the orchestration.

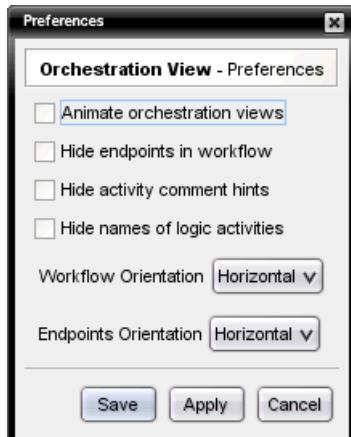


Figure 3-30 The orchestration layout preferences window

Each icon provides a description when you move the mouse pointer over it. You can find full descriptions of each icon in the information center at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.doc%2Forch_Viewing_an_Orchestration.html

3.4.2 What makes a valid orchestration

An orchestration must be valid before it can be tested and published. There are a number of rules that must be followed to make an orchestration valid:

- ▶ Every orchestration must begin with either a Starter Activity or a Pick Activity:
 - If the orchestration begins with a Pick Activity, the first activity in every branch of the Pick Activity must be a Starter Activity.
 - Activities including Starter Activities are discussed in 3.6, “Activities” on page 107.

- ▶ All variables must be initialized before they are used:
 - Initializing variables is discussed in 3.6.8, “Using HTTP and web service starter activities with Cast Iron Live” on page 116.
- ▶ All mandatory properties of all activities in the orchestration must be populated with valid values.

If an orchestration is not valid, a yellow hazard symbol (which includes an exclamation mark) displays next to the orchestration in the Project toolbox tab, as highlighted in Figure 3-31.



Figure 3-31 The yellow hazard symbol showing an invalid orchestration

3.4.3 Orchestration properties

An orchestration holds properties that refer to the whole orchestration. Set these orchestration properties in the configuration pane shown in Figure 3-32. To access the orchestration properties, click **Orchestration → Properties**. You can also access the properties by clicking the green play icon at the start of the orchestration.

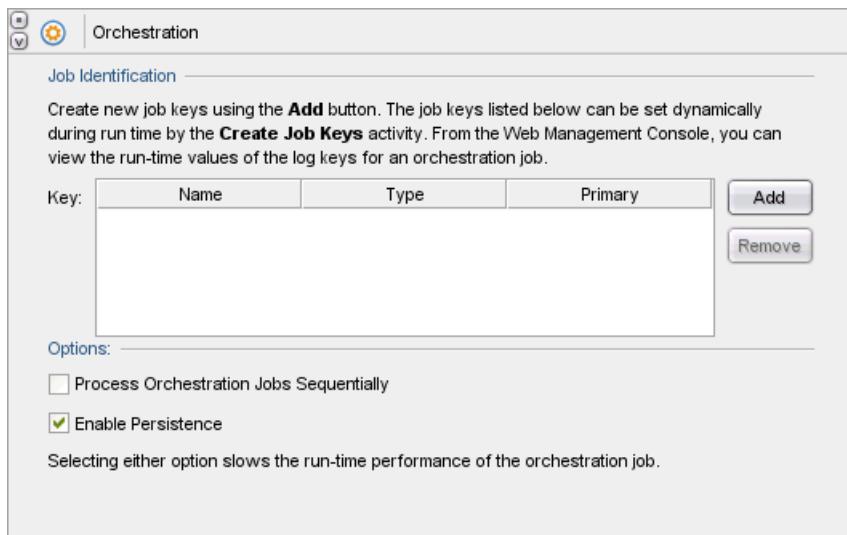


Figure 3-32 The orchestration properties in the configuration pane

Tip: If the **Orchestration → Properties** option is not available, there are a couple of reasons this can happen:

- ▶ The Orchestration Properties are already visible at the bottom of the screen.
- ▶ The Orchestration does not have focus, which can happen if you are using the Variables Toolbox Tab to edit a variable, for example.

With the orchestration open, click in the workspace, then try again.

You can set the following properties for the orchestration:

- ▶ Job Keys are created in the Job Identification section of the orchestration properties. The benefit of Job Keys and how they are used is discussed in 3.7.4, “The JobInfo variable and Job Keys” on page 122.
- ▶ By default, multiple jobs run through each orchestration in parallel on a Cast Iron runtime to improve performance. In some cases, it is important that only one job runs at any one time. For example, two jobs can conflict with each other when trying to access one resource, such as a database table where one job can delete a table row that the other job is trying to use.

To force the orchestration to run with only one job at a time, select the **Process Orchestration Jobs Sequentially** option shown in Figure 3-32 on page 99.

Tip: If any orchestration variables are defined as shared, the orchestration runs all jobs sequentially automatically. Shared variables are described in 3.7.2, “Shared variables” on page 121.

In this case, the orchestration properties view displays, as shown in Figure 3-33. The option is selected and disabled so that it cannot be changed.

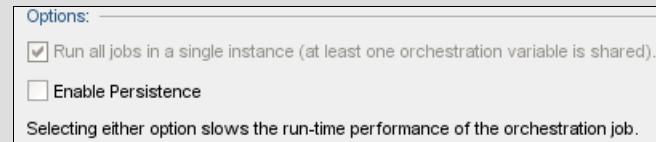


Figure 3-33 The orchestration properties when a shared variable is defined

- ▶ By default the Cast Iron runtime persists the current state of a job running through an orchestration along with its current data. Thus if a Cast Iron runtime failure occurs that causes the runtime to stop before the job runs to completion, the Cast Iron runtime remembers the state of the job, and when the runtime restarts the job restarts from the last persisted point.

However, persisting the state of each job impacts performance. Persistence is disabled for an orchestration by clearing the Enable Persistence option, as shown in Figure 3-32 on page 99. You can find further details about Cast Iron recovery in Chapter 6, “Availability and scalability planning” on page 277.

3.4.4 Orchestration documentation

You must always document an orchestration. You can add comments to activities within an orchestration by selecting each activity and clicking the yellow box at the bottom left of the activity, as shown in Figure 3-34 on page 101.

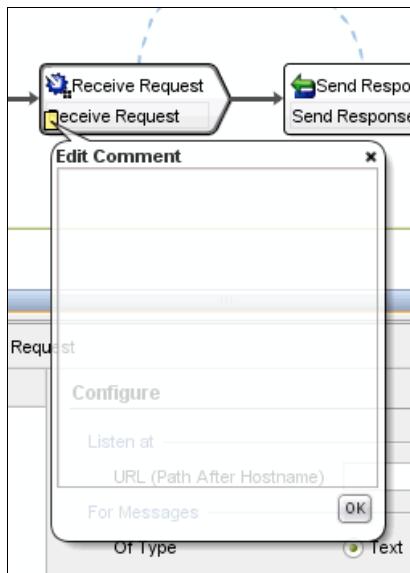


Figure 3-34 Entering a comment for an activity

A window opens to allow you to add comments. After you add comments for an activity, the yellow box for the activity is filled in with lines, as shown in Figure 3-35.



Figure 3-35 An activity with completed comments

You can also enter comments for the green start icon by clicking the icon in the same manner. Finally, you can enter comments for the whole orchestration by right-clicking the orchestration in the Project toolbox tab and then clicking **Comment**. After you add comments for the orchestration, a yellow box displays at the top left of the orchestration icon for that orchestration in the Project toolbox tab, as shown in Figure 3-36.



Figure 3-36 An orchestration with completed comments

Comments entered for the orchestration and its activities display in the project documentation files, as described in 3.3.6, “Project documentation” on page 94. However, comments that are entered for the green start icon do not display in the project documentation files.

Tip: You can also add comments to the other components of a project, such as an endpoint or an XML schema, by right-clicking the component in the Project toolbox tab and selecting **Comment**. Comments for the project are set by selecting **Project → Project Settings**. Comments for the project and all of its components display in the project documentation files.

You can save orchestrations as an image to provide further documentation alongside the project documentation. To save an image of an orchestration, open the orchestration in the workspace, and then select **File** → **Save as Image**. An image is created with the same name as the orchestration, and the image shows all the activities and endpoints in an orchestration. You can also select **File** → **Print** to print an image of the orchestration.

Figure 3-37 is an example of an image file that was produced by selecting **File** → **Save as Image**. Notice that no comments or properties display in the image. Comments and properties display in the project documentation files.

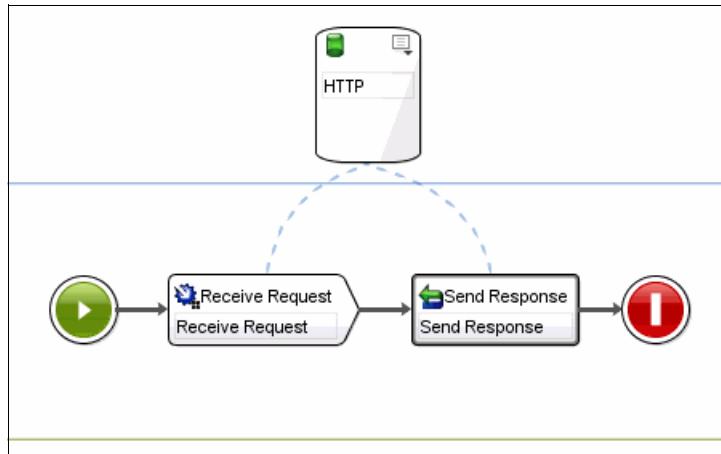


Figure 3-37 An image file showing the orchestration

3.5 Connectors and endpoints

This section discusses the concepts of connectors and endpoints. It also explains how to install a module provider, which some connector types, such as SAP and MQ, need to access the external system.

3.5.1 Connectors

A Cast Iron Integration Solution uses connectors to access external systems. A *connector* is a component installed on the Cast Iron runtime that provides the functionality to connect to a specific external system.

Cast Iron contains a number of built in connectors that fall in to two categories:

- ▶ *Application connectors* connect to specific applications, such as Salesforce.com, SAP, PeopleSoft, and so forth.
- ▶ *Technology connectors* connect to systems without a specific application endpoint, such as FTP, databases, web services, and so forth.

You can create new connectors using the Connector Development Kit described in Chapter 8, “Building custom plugin connectors” on page 321. You can add a connector to Studio by selecting either **File** → **Import Connector** to import a local connector or **Solutions** → **Plugin Connectors** to import a connector that was shared with the Cast Iron community.

3.5.2 Endpoints

An *endpoint* is defined within a project to hold the connection properties used by a connector. Each endpoint is associated with a specific fixed type of connector and contains properties for that specific type of connector. For example, an endpoint created for a database connector contains database connection properties, such as the database name, while an endpoint created for an SAP connector contains properties, such as the SAP system number.

By holding these connection properties in an endpoint component rather than in each activity that accesses the external system, the connection properties are defined only once for each external system that is used in a project, as shown in Figure 3-38. In addition, if the connection properties for the external system change, the endpoint can be updated, and the new property values are used by all activities that are associated with this endpoint.

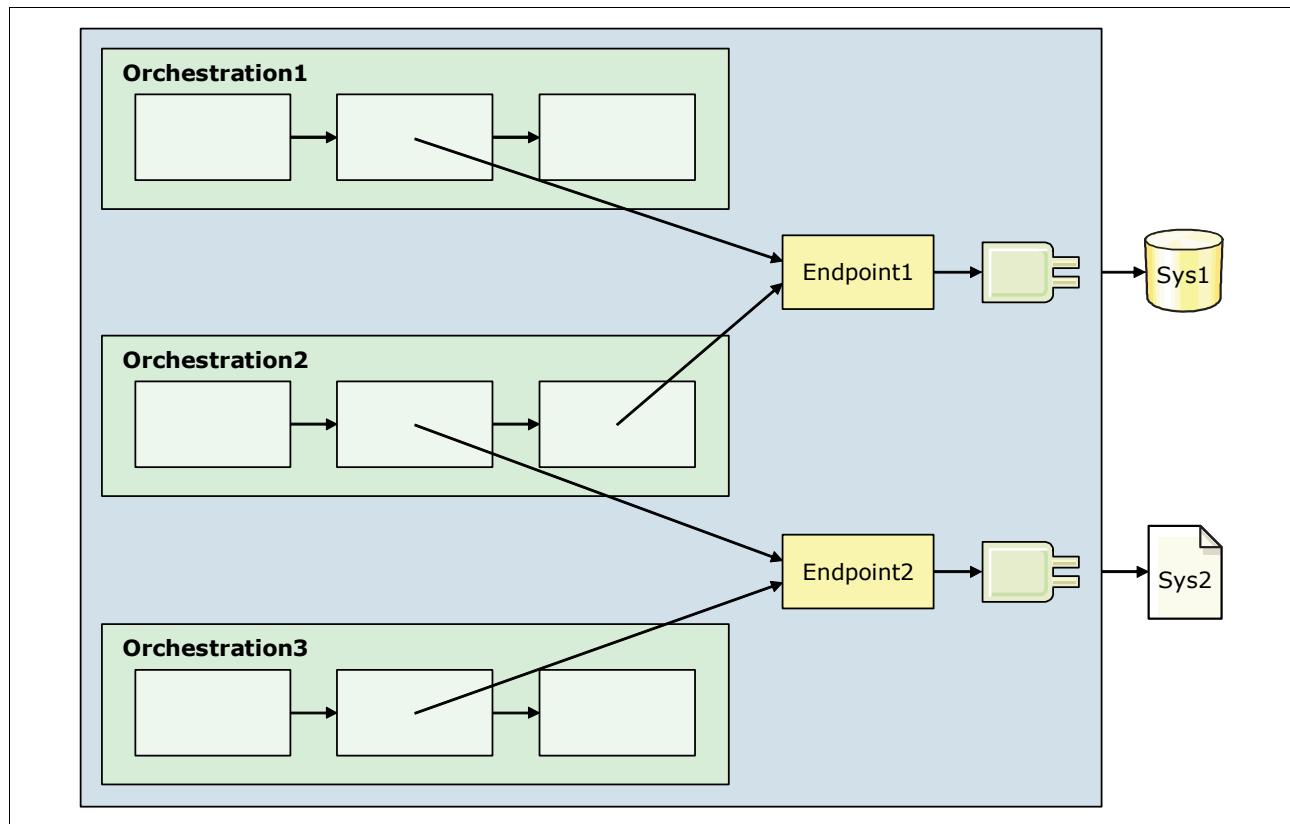


Figure 3-38 Multiple orchestrations reusing endpoints in a project

Endpoints must use configuration properties to hold the values of the connection properties. Configuration properties allow the values of these properties to be changed in the WMC at run time. Configuration properties are described in 3.3.2, “Configuration properties” on page 88.

You can create endpoints from the Project toolbox tab by clicking **New Endpoint** or right-clicking Endpoints and selecting **Create Endpoint**. A dialog box opens that shows all connectors in this copy of Studio, as shown in Figure 3-39 on page 104. Select the type of endpoint that you want to create. You cannot change the type of an endpoint after it is created. However, you can create a new endpoint with different properties and change activities to use this new endpoint. You can then delete any endpoint that is not used by the activities in an orchestration.

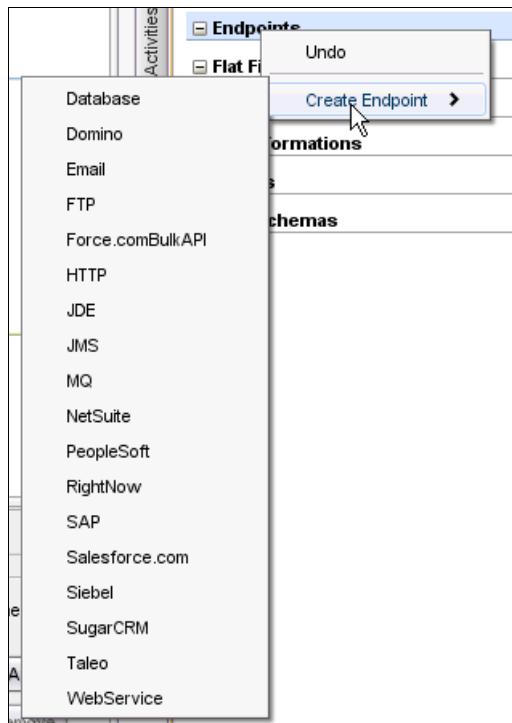


Figure 3-39 Creating a new endpoint

The new endpoint is given a default name. Change the default name to a name that is unique within the project and that describes the functionality of the endpoint. For a description of valid names, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.doc%2Fref_About_Valid_Names.html

To set the properties of an endpoint, double-click the endpoint in the endpoints section of the Project toolbox tab. An entity tab opens that shows the specific settings for the connector that is associated with this endpoint. Figure 3-40 on page 105 shows an endpoint for an SAP connector with no values entered. The yellow hazard symbol indicates a required property that has not yet had a valid value entered.

The screenshot shows the configuration dialog for a new SAP endpoint. The interface is organized into several sections:

- SAP Application Server**: Contains fields for Host Name, System Number, and a warning icon.
- Client Information**: Contains fields for SAP Client, User Name, Password, and Language (set to EN).
- Connection Pool Options**: Contains Maximum Connections (set to 25) and Maximum Idle Time (set to 120 minute(s)).
- Inbound Gateway**: Contains checkboxes for 'Enable inbound gateway' and 'Required only for the Receive IDOC activity'. It also includes fields for Host, Server Instances, Program ID, Service, and Retry Interval (set to second(s)).
- Remote Endpoint Configuration**: Contains a checkbox for 'Endpoint Runs Behind Firewall' and a field for 'Secure Connector Name'.
- Buttons**: At the bottom left is a 'Test Connection' button.

Figure 3-40 A new SAP endpoint with no values entered

Notice the Test Connection button at the bottom of the SAP endpoint in Figure 3-40. Some endpoint types can issue a connection to the external system from Studio to test that the connection properties are entered correctly. Test connection is described in 3.12.1, “Endpoint Test Connection” on page 159.

The SAP endpoint in Figure 3-40 can be used by activities to run SAP functions that read in data from the SAP system and write out data to the SAP system. However, some endpoints are one way, which means that the endpoint can only be used to read in or to write out. An HTTP endpoint is an example of a one way endpoint. Figure 3-41 on page 106 shows a section of an HTTP endpoint that was configured to receive HTTP requests. To change the HTTP endpoint to send out HTTP requests, select the **Remote Server** option and enter an HTTP host name and port.

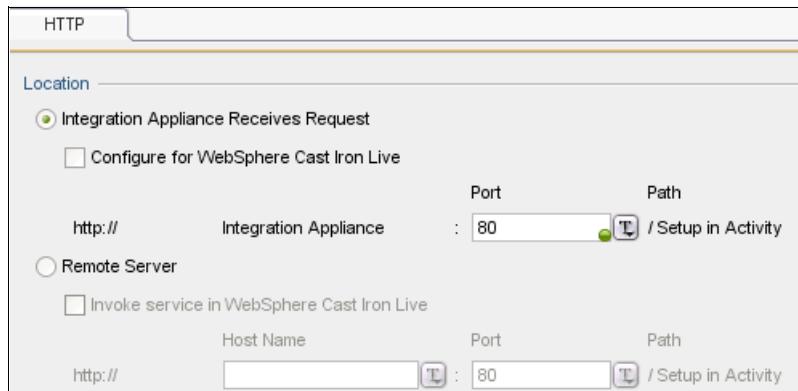


Figure 3-41 An HTTP endpoint configured to receive requests

Notice that the HTTP host name in Figure 3-41 cannot be set for an HTTP endpoint that is configured to receive requests because the integration solution that is using the HTTP endpoint will be published to a specific Cast Iron runtime, and the HTTP host name of this runtime will be used.

Every endpoint type contains detailed properties that are specific to the connector with which it is associated. More information about each of the endpoint types and the properties that can be set can be found in the WebSphere Cast Iron information center.

Endpoints: Endpoints are used at development time to interrogate external systems to discover the structure of the systems. This is used to generate the queries that are run by activities and to pre-populate the list of input and output parameters required by activities.

3.5.3 Installing module providers

Some connector types, such as generic JDBC drivers and SAP, need module providers or drivers to be able to access the external system. These module providers must be installed on every copy of Studio that needs to use the connector. The module provider is held for Studio and not for each project. Thus, you must install the module provider only once for each copy of Studio.

For Cast Iron Live, Studio uses the connector libraries installed on Cast Iron Live for the tenant that is starting Studio as described in 4.9.3, “Update connector libraries” on page 229. These libraries are installed individually for each tenant environment. You use the WMC settings to set which environment is used to find the connector libraries by Studio, as described in 4.1.2, “Web Management Console” on page 175.

To install a module provider in Studio:

1. Click **Tools** → **Install Module Providers**. The window shown in Figure 3-42 opens.



Figure 3-42 The install module providers window

2. Click the plus icon for the type of provider that you want to add, and then click **Browse**.
3. Navigate to the file that contains the module provider for the external system that you want to access and select it.
4. Click **OK**.
5. To install the module provider, Studio must close and reopen. Thus, a confirmation box opens telling you that Studio will close. Click **Yes** to close Studio. Then, restart Studio manually.

You must also install module providers on the Cast Iron runtime where the orchestration that is using the connector will run. For more information about this topic, refer to 4.9.3, “Update connector libraries” on page 229.

3.6 Activities

Activities are pre-written functional components that are placed in an orchestration and configured to create an integration solution. There are many different activities built into Studio. These activities provide a full range of functionality, but the development process to use them within Studio is the same.

3.6.1 Configuring and using an activity

Activities are selected from the Activities toolbox tab. They are grouped in to categories. To use an activity:

1. Open the orchestration where you want to use the activity.
2. Drag the activity from the Activities toolbox tab to the place in the orchestration where you want it to execute.

All activities have an activity name that displays on the activity icon in the workspace. Change the name of an activity to indicate the role of the activity in the orchestration as clearly as possible. The activities in the orchestration are then self-documenting. Change the name of the activity to indicate how the activity is being used by right-clicking the activity and then selecting **Rename**.

Tip: There is no maximum length for an activity name in Studio. However, the WMC displays only the first 30 characters of an activity name.

Most activities require some configuration. To set an activity's properties, click the activity after it is added to an orchestration. The configuration pane shows the properties for the selected activity, as shown in Figure 3-43.

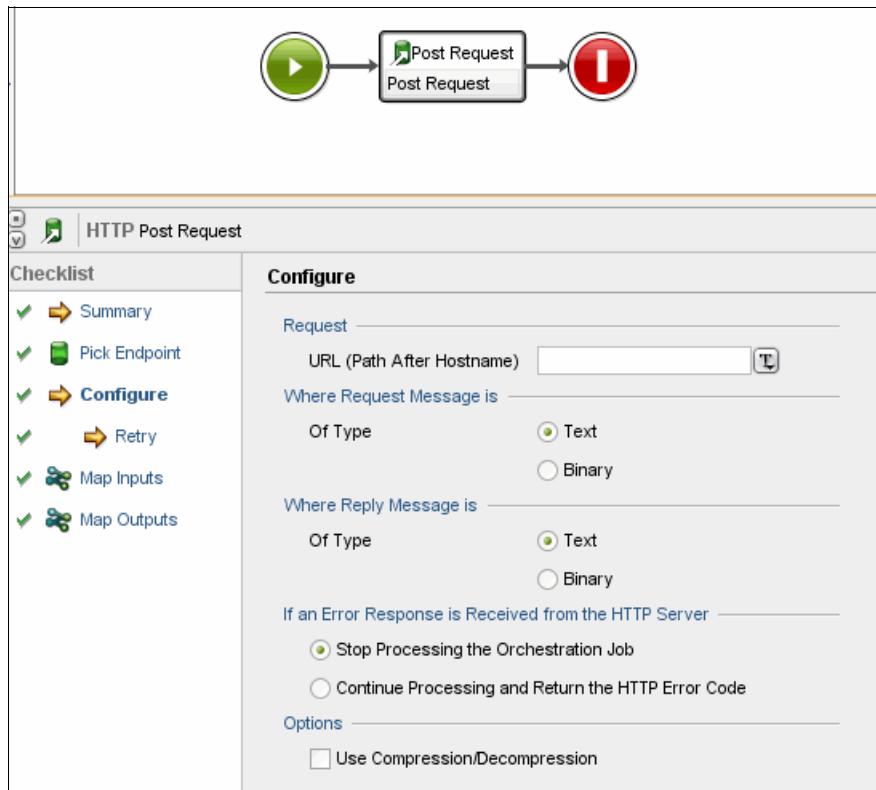


Figure 3-43 The configuration pane for an HTTP Post Request activity

The configuration pane provides a checklist of tasks for the activity that must be completed. When a task is completed with valid values, it shows a green check mark next to this task. A yellow hazard symbol (which includes an exclamation point) displays next to any task that is not completed.

Figure 3-44 on page 109 shows a checklist with both complete and incomplete tasks.

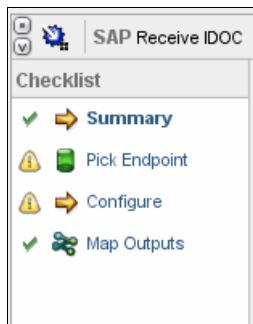


Figure 3-44 An incomplete checklist for an SAP Receive IDOC activity

You can change properties for the activity in Studio by selecting the activity and clicking the checklist task that contains the property.

Some properties fields are check boxes or drop-down menus. Other fields allow text entry. Some fields are set by selecting specific values that are generated by Studio. You set fields with a Browse button or an ellipsis button (...) next to them, as highlighted in Figure 3-45, by clicking the button and then selecting the appropriate value. Use the Browse or ellipsis buttons where they are available, even if the field allows text entry, to ensure that the syntax is correct.

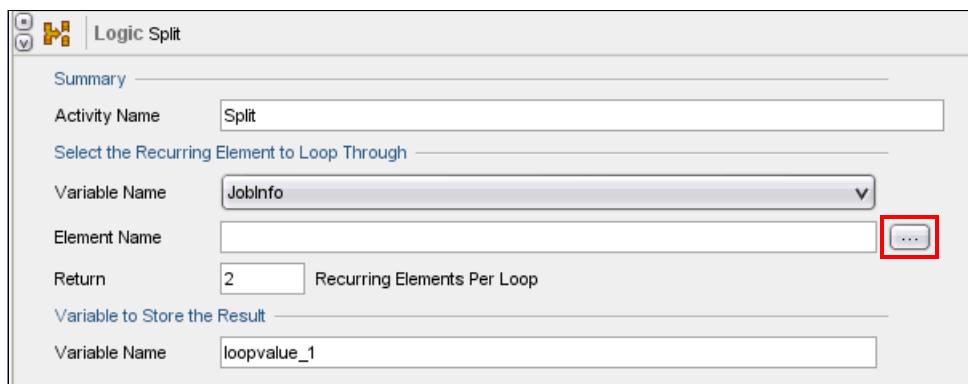


Figure 3-45 An entry field with an ellipsis button

Activities can contain the following tasks in their checklist:

Summary	A description of the functionality of the activity.
Pick Endpoint	For activities that use external systems, this task sets the endpoint that holds connection properties to the external system.
Configure tasks	A set of tasks to change how the activity behaves.
Map Inputs	A mapping editor that defines how the input parameters for the activity are set.
Map Outputs	A mapping editor that defines where the output parameters from the activity are written.

The mapping editor is discussed in 3.9, “Maps” on page 128.

The checklist for each activity depends heavily on the functionality that the activity provides. Some activities require little configuration. Other activities have a lot of checklist tasks. Activities, such as the Apply XSLT and the Web Services Invoke Service activities, require additional files, such as stylesheets, WSDL files, or XML schemas, to be imported in to Studio

to configure the activity. These files are imported as one of the tasks in the checklist for each activity.

Tip: If a new version of an XML schema or a WSDL file needs to be imported in to Studio, right-click the name of the file in the Project toolbox tab, and then select **Update**. A window opens to select the new version of the file. Studio then imports the file and updates all references to the file to reflect the new structure of the XML schema or WSDL.

Some activities that run using external systems must access these external systems directly at development time in Studio. This access allows Studio to interrogate the external system to discover how the system was set up. For example, the database Insert Rows activity shown in Figure 3-46 is configured by connecting to the database and discovering the tables and the list of columns in a selected table. The input for the activity is set to the selected columns. This task is a required task. Therefore, access to the database from Studio is required at development time.

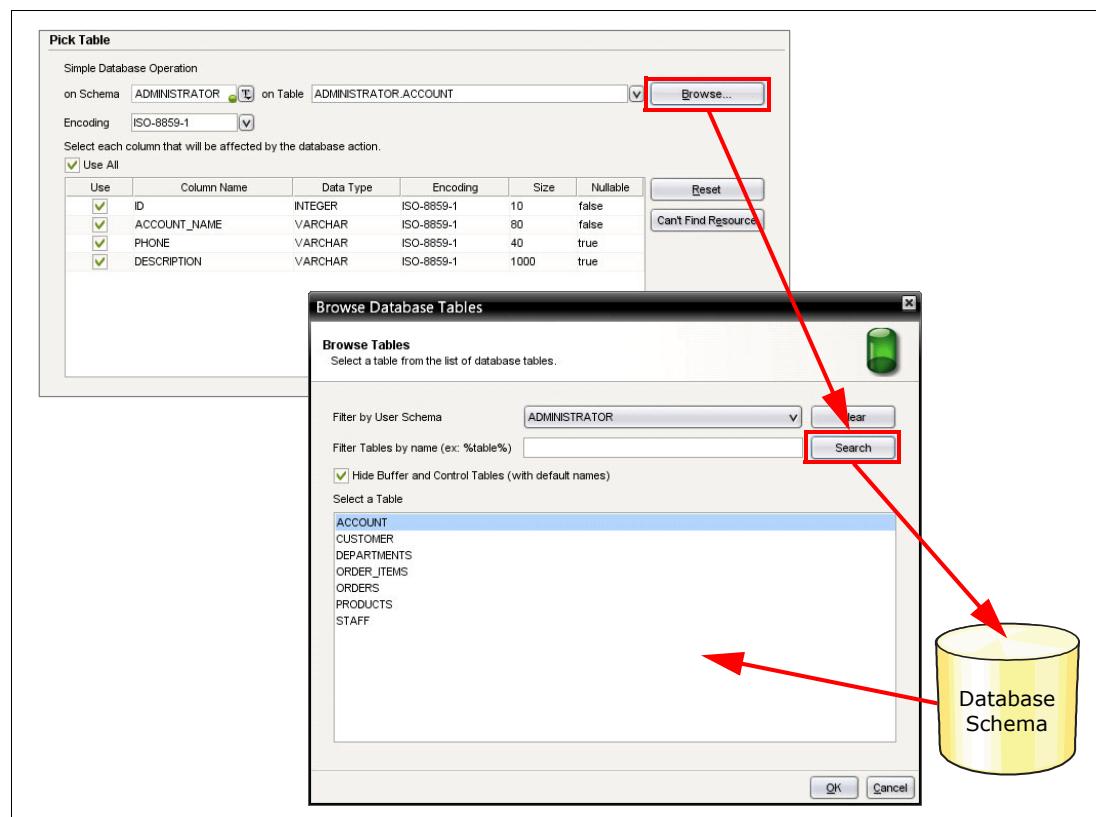


Figure 3-46 The database Insert Rows activity needs access to the database at development time

It is important, therefore, to provide a direct connection from Studio at development time to any external systems that are used by activities that interrogate these external systems. Without access to these systems from Studio at development time, some activities cannot be configured.

Tip: Studio cannot access the Cast Iron Secure Connector that is used by Cast Iron Live to access an external system running in a private network behind a firewall. Therefore, to configure access to an external system that is running in a private network, Studio must be provided with a direct connection in to this private network so that it can access the external system at development time.

You can find full documentation about the large number of activities that are available in a Cast Iron Integration Solution, the functionality of these activities, and the properties that must be set for each activity at:

<http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp>

The remainder of this section introduces a few groups of activities.

3.6.2 Starter activities

A *starter activity* starts a job instance of an orchestration based on some condition. Every orchestration must start with either a starter activity or a Pick activity to be valid. A Pick activity groups together a set of possible starter activities, any of which can start the orchestration.

Starter activities start a job instance of an orchestration because an action occurred, for example:

- ▶ An HTTP Receive Request activity that starts a job instance when it receives an HTTP request to a particular URL on a particular port.
- ▶ A Salesforce.com Poll Updated Objects activity that regularly polls Salesforce.com and starts a job instance when it detects that particular objects are updated.
- ▶ A Schedule Job activity that starts a job instance regularly at a specified interval.

Most starter activities use external systems to detect when a new job instance must be started. Thus, these starter activities must have an endpoint associated with them.

All starter activities, apart from the Schedule Job activity, are identified by the icon highlighted in Figure 3-47.

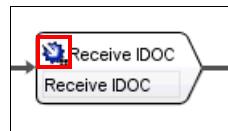


Figure 3-47 An activity showing the starter activity icon

Tip: An HTTP Receive Request activity can be used to manually start an orchestration. This orchestration can then integrate systems that use other endpoints, for example SAP and Salesforce.com. An example of using an HTTP Receive Request activity in this manner is described in Chapter 10, “Scenario: Bidirectional account synchronization” on page 357.

3.6.3 Outbound activities

Outbound activities are the other group of activities along with starter activities that access external systems and use endpoints. Outbound activities send functional requests to external systems. These can be requests to receive information from the external systems or requests to change the external systems. Some examples of outbound activities are:

- ▶ A Web Services Invoke Service activity that runs a web service operation on an external web service provider.
- ▶ A Siebel Update Objects activity that updates an existing data record.

- ▶ A Data Quality Lookup activity that pulls data from a database and uses that data to update values of defined fields in a data structure passed to the activity.

The outbound activities that are available for each external system and the properties for these activities depend on the external system being called.

3.6.4 Transform activities

Transform activities transform data from one format in to another format. This transformation can be done using Cast Iron maps and XML stylesheets, as discussed in 3.9, “Maps” on page 128 or by parsing the data as described in 3.10, “Parsing data” on page 149.

3.6.5 Logic and utility activities

Orchestrations can contain logic activities to control the flow of the orchestration. Studio provides loops using the While Loop, For Each, Split activities, and conditional logic using the If..Then activity.

These four activities provide a variable scope, which means that any variables that are initialized inside the loop or If..Then activity are not visible outside of the activity. Initializing variables and variable scope is discussed further in 3.6.8, “Using HTTP and web service starter activities with Cast Iron Live” on page 116.

For Each and Split, activities loop through a recurring element in a variable. For example, if an order contains a number of products that are being ordered, the activities can loop through each product.

Tip: The For Each and Split activities use an XPath expression to select the element to loop through.

Thus, rather than looping through every product and then using an If..Then activity inside the loop to choose whether to process the product, an XPath expression can be used to select a subset of the products to loop through. For example, an XPath expression of the following form selects only the products whose price is greater than 500:

Product[*:Price>500]

Studio contains an XPath Evaluator tool, which is described in 3.12.5, “XPath Evaluator” on page 163, that you can use to construct XPath expressions. XPath is a XML open standard and is not described in this book.

The Group activity also provides a variable scope. It is used to visually organize an orchestration and has no function when the orchestration executes. If an orchestration is complex, drag a set of activities inside a Group, and give the Group a name to provide more visual structure to the orchestration. Minimize groups by clicking the minus symbol (-) at the top left of the activity to hide the activities inside it, and thus the complexity of the orchestration, as shown in Figure 3-48.

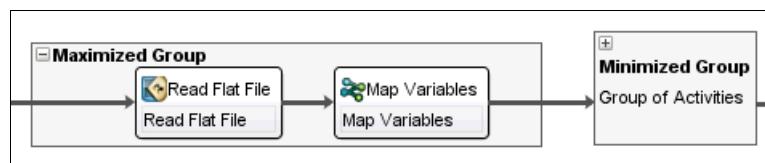


Figure 3-48 A maximized and a minimized Group

Exception handling is provided in an orchestration using the Try activity, which is discussed in 3.11, “Exception handling” on page 155.

There are a number of utility activities. The Log Message activity writes a message to the system log with the log level that is specified in the activity. The Schedule Job activity is discussed in 3.6.2, “Starter activities” on page 111. The Create Job Keys activity is described in 3.7.4, “The JobInfo variable and Job Keys” on page 122.

3.6.6 Local web services calls

To simplify the testing of web services calls between two orchestrations in Studio, you can create a Web Services endpoint with a location of Local. This option is only supported for two orchestrations in the same project that are being tested in Studio and that are calling each other. Testing in Studio is discussed in 3.12, “Unit testing” on page 159.

To create a Local web services call between two orchestrations:

1. Right-click **Endpoints** on the Project toolbox tab, and select **Create Endpoint → WebService**.
2. In the Web Service endpoint, select a Type of Provide and a Transport of Local. Set a suitable path for the endpoint.
3. Right-click **XML Schemas** in the Project toolbox tab, and then select **Add Document** to add an XML schema that defines the data types for the input and output parameters of the web service.
4. Create an orchestration:
 - a. Add a Web Services Provide Service starter activity.
 - b. In the checklist Summary task, change the activity name. This name is used for the operation name of the web service.
 - c. In the checklist Pick Endpoint task, select the WebService endpoint that you created in step 1.
 - d. In the checklist Configure task, select the schema for the request message. Select the schema for the reply message, if the web service returns a result.
 - e. Create the remainder of the orchestration.
5. Right-click **Endpoints** on the Project toolbox tab, and then select **Create Endpoint → WebService** to create a second Web Service endpoint.
6. In the second Web Service endpoint, select a Type of Invoke and a Transport of Local. From the Orchestration drop-down menu, select the orchestration to be called. From the WebService Provide Activity drop-down menu, select the name of the Provide Service activity in the orchestration to be called.
7. Create the calling orchestration with a Web Services activity Invoke Service that uses the second Web Service endpoint. The Configure task of the checklist is set automatically from the local WSDL in the second Web Service endpoint.

The second orchestration can now call the first orchestration locally when testing in Studio. Note that Local calls are not supported on the Cast Iron runtime. Thus, the Local call is replaced by a Remote HTTP web services call before the orchestrations are published.

Tip: When the endpoint for the Provide Service activity is updated to have a Transport of Remote, a WSDL file is generated for the web service that is defined by that Provide Service activity. This WSDL is then given to the application that is invoking the web service.

The generated WSDL is created as an asset for the project and is published automatically to a Cast Iron runtime along with the project. You can download the WSDL using the WMC. This is described in 4.2.11, “Assets” on page 190.

You can view and store the generated WSDL inside of Studio by right-clicking the Provide Service activity and then selecting either **View Generated WSDL** or **Add Generated WSDL To Studio Project**.

3.6.7 Database activities, assets, and the staging database

Database activities poll, read from, and write to database products. Starter database activities can be triggered when a database changes or when rows are inserted, updated, or deleted. Other database activities can execute queries against a database.

The Database activity Call Procedure calls a stored procedure on the database. A stored procedure is a piece of code that runs locally on a database, which means that the data that is needed by the stored procedure does not have to be sent over a network. Not sending the data over the network is more efficient and more secure than loading the data into an orchestration.

The Data Quality activity Lookup also executes a query on a database endpoint. This activity combines a For..Each loop with a database query. The activity can perform a fuzzy lookup to return data that approximately matches the input parameters. This fuzzy lookup requires the Lookup activity to use the local staging database described later in this section. The activity can also preload a cache of the data it is configured to look up. Thus, use the Lookup activity instead of a standard database activity inside a For Each activity. You can find full details about the Lookup activity in the information center at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/lookup_activity.html

The other Data Quality activities in Studio are the Sort activity, the Filter and Profile activity, which is used in Chapter 10, “Scenario: Bidirectional account synchronization” on page 357, and the Merge activity, which is used along with the Lookup activity in Chapter 12., “Scenario: Data enrichment and aggregation” on page 465.

Database assets

To detect when a database table is updated you must use a trigger on the database. You can also use a buffer table that holds the updated data until it is processed.

If each update to a database table needs to be made exactly once, you must use a control table that temporarily remembers the changes that you already made to the table that you are updating. This is controlled by the Deliver Messages field in database activities. This option is explained in the following page in the information center:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/db_specifying_delivery_rules_for_database_activities.html

Both of these situations involve changing the design of the database that is being used. Each table, trigger, or other database artifact that must be created or changed is referred to as a database asset. Cast Iron helps with this by generating database scripts for each database

asset that must be created or changed. These database scripts are generated when a project is published and can be accessed using the WMC.

The following database activities require changes to the design of the database that the activity is accessing:

- ▶ Starter database activities
- ▶ Activities with the exactly once delivery option set

Database assets that might be needed by Cast Iron activities are control tables, buffer tables, sequence tables (for Oracle only), indexes (for the local Staging Database only), or triggers.

The database asset scripts are accessed through the Configuration Details page of the WMC. These scripts can be run from within the WMC or downloaded and run manually. Accessing assets is described in 4.2.11, “Assets” on page 190.

The scripts are also provided in Studio when an orchestration that requires a database asset is tested using the Verify toolbox tab. When the orchestration is started, the window in Figure 3-49 opens. The script can be run directly against the external database from this window. Testing with the Verify toolbox tab is discussed in 3.12.7, “Verifying orchestrations” on page 164.

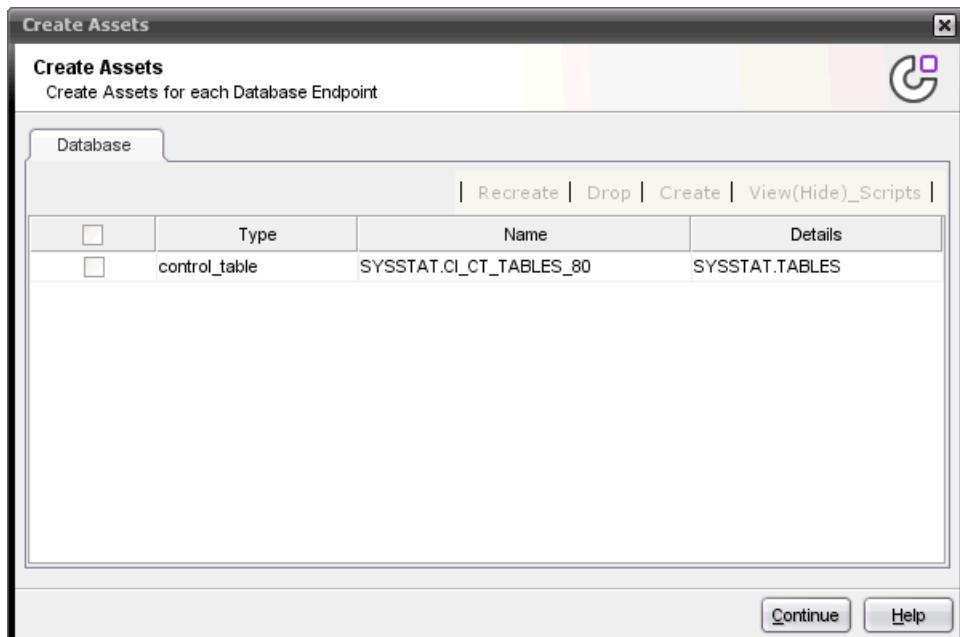


Figure 3-49 The Create Database Assets Studio window

You can find more information about database assets at the following pages in the information center:

- ▶ http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/db_poll_table_activity.html
- ▶ http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.appliance.doc/Managing_Projects/generatingDBAssets.htm
- ▶ http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/conn_ref_Connector_Prerequisites_for_Database_Operations.html
- ▶ http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.appliance.doc/Managing_Projects/recreatingDBAssets.htm

The local staging database

Cast Iron Integration Appliances provide a local database on the appliance that can be accessed only using the Local database endpoint type. This database holds data that is loaded by a database activity in an orchestration. The data can then be processed more efficiently because it is now local to the orchestration that is using the data. The Local database can be accessed only by orchestrations running on the Cast Iron Integration Appliance with the database and cannot be accessed directly from external applications.

Tip: The Local database endpoint uses the data IP address of the Integration Appliance for the Network Location Server field. The user name and password must be for a user with Publisher privileges on the Integration Appliance.

For example, if an orchestration is sorting data from a database, the data is first read from the external database and then written in to the local staging database with a database activity using the Local database endpoint. Then a Data Quality Sort activity is used to sort the data without having to make repeated calls to the external database. The data is then available to be written to the destination external system.

The Data Quality Merge activity and the Data Quality Filter and Profile activity both also benefit from using the local staging database. The Data Quality Lookup activity requires that the local staging database is used if a fuzzy lookup is performed.

The orchestration that is loading the data does not have to be the same orchestration that is performing the sort, making the processing even more efficient. Thus one orchestration can use regular job instances to load data into the staging database and a second orchestration can run as an infrequent batch job to process this data.

The staging database must be started using the WMC before it can be accessed. Controlling the staging database is described in 4.5, “The staging database” on page 217.

Cast Iron Live: The local database is not provided on Cast Iron Live.

3.6.8 Using HTTP and web service starter activities with Cast Iron Live

An orchestration containing either an HTTP Receive Request starter activity or a Web Services Provide Service starter activity is executed by receiving an HTTP request. If this orchestration is running on Cast Iron Live, the HTTP Request determines which tenant and environment contains the orchestration that is executed by the request.

To send an HTTP request to an orchestration running on Cast Iron Live, use the URL:

`https://<Hostname>:443/env/<Environment>/<RequestURI>`

The details of this URL are as follows:

https	The request must always use the HTTPS protocol for Cast Iron Live.
Hostname	Send the request to the host name <i>provide.castiron.com</i> . If you are using the evaluation cloud, the host name is <i>eval-provide.castiron.com</i> .
443	The port that Cast Iron Live uses for HTTPS. This is always 443.
env	This is a mandatory value and must always be present. If a user name and password are included in the URL, this will be envq instead, as discussed later in this section.

Environment	The name of the environment for the Tenant where the orchestration is running, for example Development or Production.
RequestURI	For an HTTP Receive Request, this is the URL (Path After Hostname) property in the Configure task of the activity. For a Web Services Provide Service this is the Path property in the Web Services endpoint.

An example of a completed URL is:

`https://provide.castiron.com:443/env/Development/syncNetSuiteGoogleCalendartest`

The HTTP request must also provide a user name and password in the request. This user name is a Cast Iron user name created using the WMC. The user must have access to the environment where the orchestration is running.

The user name and password can be provided using the HTTP Basic Authentication header. Alternatively, they can be included as part of the URL, for example:

`https://eval-provide.castiron.com:443/envq/Production/loadSAP?ciUser=<user>&ciPassword=<password>`

Notice that this url contains /envq/ instead of /env/, which is required if a user name and password are included in the URL.

An example of sending HTTP requests to an orchestration running on Cast Iron Live is described in Chapter 11, “Scenario: CRM to cloud calendar services” on page 435.

3.7 Variables

Data is held within an integration solution in *variables*. Each orchestration has its own set of variables; therefore, no data can be passed between orchestrations without making an external call. Each execution of an orchestration is given its own set of variables unless that variable is defined as shared, which is discussed later in this section. All non-shared variables for an orchestration are deleted at the end of the execution of the orchestration.

All output data from activities in an orchestration is written to variables. There are no automatic variables created to hold output from an activity. Thus, if data from one activity is to be passed to the next activity in an orchestration, the data must first be written to a variable as the output of the first activity before being read from the variable as the input to the next activity, as shown in Figure 3-50.

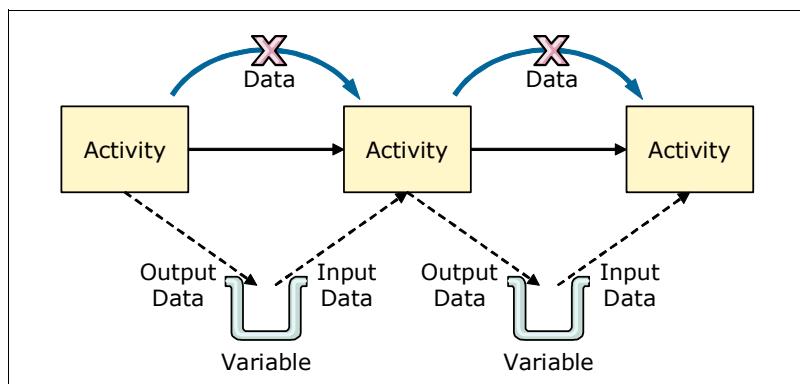


Figure 3-50 Data must be passed between activities using variables

Data is passed in to an activity using the Map Inputs task of the activity. Data is written out of an activity using the Map Outputs task of the activity. *Map Inputs* and *Map Outputs* tasks are described in 3.9, “Maps” on page 128.

All variables have a data *type*, meaning that they can only hold data of the specified type. Variables can have primitive data types or complex data types. Primitive data types are where the data is of only one type, such as a string, an integer, or a boolean. Complex data types consist of a set of primitive types or other complex types, as shown in Figure 3-51. You can define elements inside a complex data type to be repeating fields. You define complex data types in Studio using XML schemas.

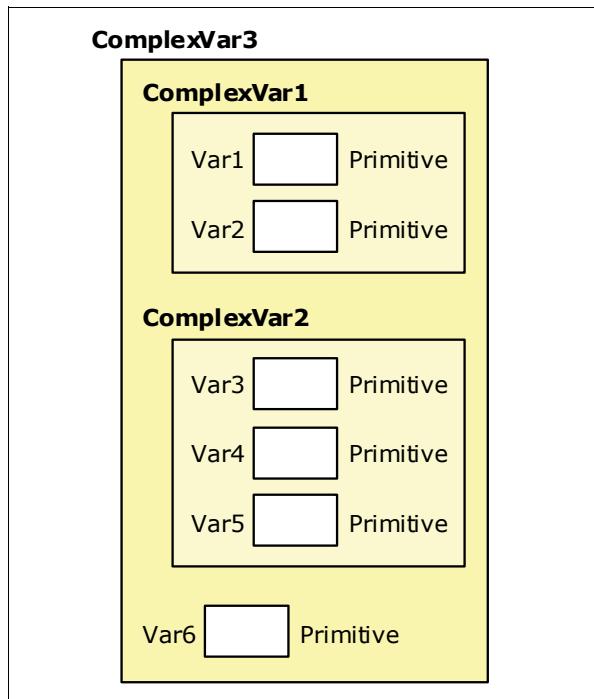


Figure 3-51 Variables with complex and primitive data types

Studio contains a Variables toolbox tab. The orchestration that contains the variables must be open in Studio before the variables for the orchestration show in the Variables toolbox tab. The Variables toolbox tab has the following sections:

- | | |
|-------------------------------------|---|
| Variables | A list of the variables currently available. |
| Schema of <var> | The XML schema structure of the data type of the selected variable. |
| Activities using <var> | A list of all of the activities that use the selected variable. |
| Properties of <var> | The editable properties of the selected variable. |

Figure 3-52 shows an example of the sections of the Variables toolbox tab.

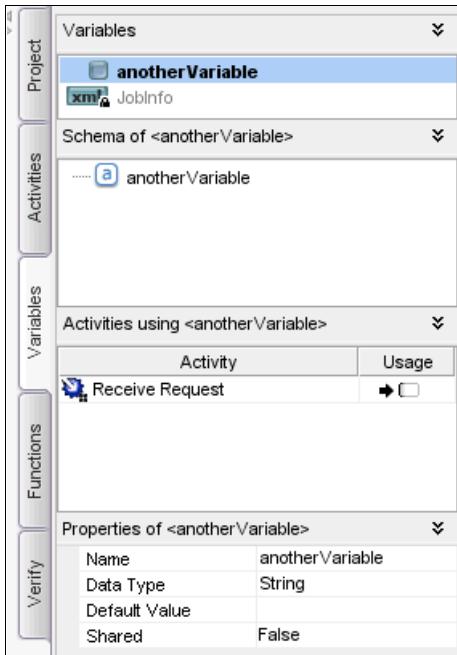


Figure 3-52 The Variables toolbox tab

3.7.1 Creating a variable

To create a variable:

1. Right-click in the Variables section of the Variables toolbox tab, and click **Create New Variable**. A window opens that shows the currently available data types, as shown in Figure 3-53 on page 120.

If a new XML schema is needed, you can add it by clicking **Add**.

2. Select the data type for the variable you are creating.

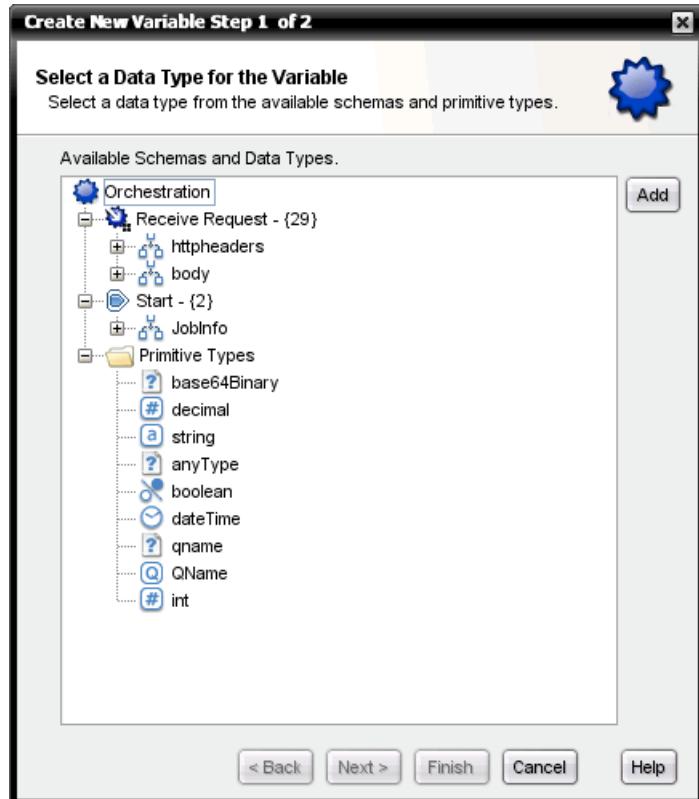


Figure 3-53 The create new variable window

3. Click **Next** to enter the name of the variable. Use a consistent naming convention for all variables in a project.
4. Click **Finish**. The new variable displays in the Variables toolbox tab.

You can rename the variable from the Properties section of the Variables toolbox tab. You can also change the variable data type from this section and set a default value for the variable. Some variables are created by activities and given default names. For example, the Try activity creates variables to hold information about the exception that is being handled. Rename these variables to follow the naming convention that you established for the project.

Tip: It is possible to rename a variable in the mapping editor, but this change does not change the original variable name. Instead, it creates a new variable of the same data type with the new name. If you accidentally create a new variable:

1. Click the down arrow next to the variable name in the mapping editor, select **Choose Another Variable**, and then select the original variable.
2. Select **Remove Unused Variable(s)** in the Variables toolbox tab to remove the new variable, as described in 3.7.3, “Deleting a variable” on page 122.

You must initialize variables in an integration solution before you can use them. Thus, the first use of a variable must be to set a value to that variable and not to read from that variable. Variables are visible only within the scope in which they are first initialized, for example, a variable initialized inside a while loop is not visible outside of that loop, as shown in Figure 3-54 on page 121.

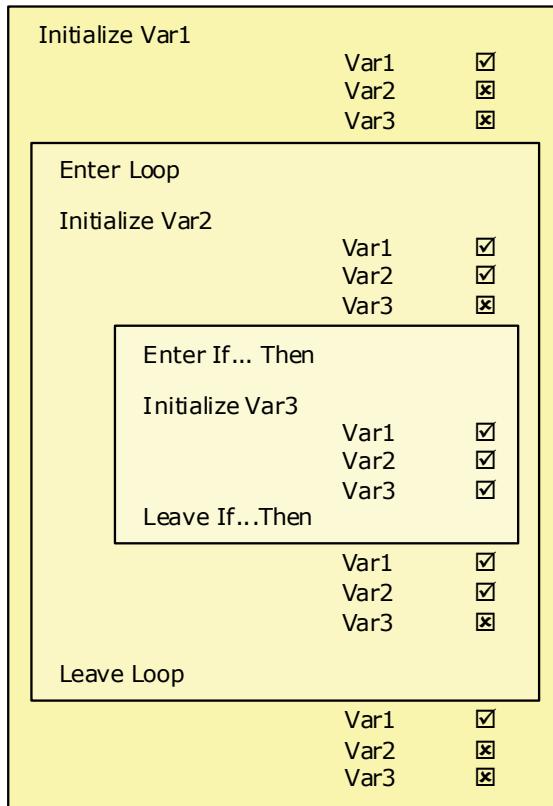


Figure 3-54 Variable scopes

Tip: When a variable is given a default value in the properties section of the Variables toolbox tab, that variable is initialized at the start of an orchestration. Therefore, even if the variable is first used explicitly inside a scope, such as a while loop, it is still visible outside of the scope because it was initialized before the orchestration entered the scope. To use a variable inside a loop and then after the loop, give that variable a default value.

3.7.2 Shared variables

Variables can also be set to be shared by setting shared to true in the properties section of the Variables toolbox tab. A shared variable is shared across all job instances running through one orchestration in a Cast Iron runtime. A new copy of the variable is not created every time a new job instance runs. Instead, the shared variable is remembered and reused by the next job instance of the orchestration until the project configuration for that orchestration is stopped. Thus, a value for the variable can be set in one job instance running through the orchestration and then the next job instance of that orchestration will see the value for this variable.

Tip: An orchestration containing shared variables always runs sequentially, as described in 3.4.3, “Orchestration properties” on page 99.

Shared variables persist information between jobs. An example of this is to store the last date and time that a job ran. Thus, the next job instance knows when data was last retrieved from an external system into the orchestration. The job instance then retrieves only data that was newly updated. Cast Iron polling activities do this automatically. However, where a poll activity

does not exist for an endpoint, for example when the connector does not contain the functionality to poll the endpoint, a shared variable can be used. An example of using shared variables in this manner is described in Chapter 11, “Scenario: CRM to cloud calendar services” on page 435.

3.7.3 Deleting a variable

You can delete variables from the Variables toolbox tab by right-clicking the variable and then selecting **Remove Variable**. You cannot remove a variable if it is being used in the project. Remove all unused variables from a project to reduce the complexity of the project and; therefore, making the project easier to maintain. To remove unused variables, right-click any variable in the Variables toolbox tab, and then select **Remove Unused Variable(s)**.

3.7.4 The JobInfo variable and Job Keys

The *JobInfo variable* is created automatically for every orchestration and cannot be deleted. Every job that runs through the orchestration populates the JobInfo variable with the following information for the job:

jobId	A unique job ID number that is assigned to the job by the Cast Iron runtime.
jobStartTime	The date and time when the job started.
 projectName	The name of the project that contained the orchestration.
 configurationName	The name of the project configuration associated with the orchestration. Notice this can be different from the project name if the project configuration was cloned in the WMC.
 orchestrationName	The name of the orchestration.
 routerHostName	The host name of the Cast Iron runtime where the job is running.

The information in the JobInfo variable can be read by activities in an orchestration. The information is also visible in the WMC on the Job Log pane as discussed in 4.3.3, “Job log” on page 206.

An orchestration can additionally set *Job Keys*, which are extra key/value pairs that are shown in the Job Log pane of the WMC. Job Keys are useful to allow an administrator to use the WMC to check on the results from a particular job.

Job Keys can contain any key/value pairs of data that are available to the orchestration that is running the job. Examples of the Job Key data that can be shown on the WMC are:

- ▶ The number of rows of input data that were successfully processed by each job and the number of rows that failed.
- ▶ The user name of the original sender of an MQ message held inside the MQ message descriptor that was used to access an external system.
- ▶ The name of a file written out by an FTP Put File activity.

Before a Job Key can be used, you must register it in the orchestration properties. To set the orchestration properties:

1. Select **Orchestration → Properties**.
2. Click **Add** in the orchestration properties.
3. Set the name and type of the Job Key variable, as shown in Figure 3-55 on page 123.

Job Identification			
Create new job keys using the Add button. The job keys listed below can be set dynamically during run time by the Create Job Keys activity. From the Web Management Console, you can view the run-time values of the log keys for an orchestration job.			
Key:	Name	Type	Primary
	OrderID	decimal	<input type="checkbox"/>
	CustomerName	string	<input checked="" type="checkbox"/>
	OrderDate	dateTime	<input type="checkbox"/>

Add **Remove**

Figure 3-55 Adding Job Keys

If the Primary option is selected for a Job Key, the value of this Job Key is used as the jobId property in the JobInfo variable. Only one Job Key can be set as primary. To change the primary Job Key, you must first clear the existing primary Job Key option, and then select the new primary Job Key.

To set the value of a Job Key, use the Create Job Keys activity. The input variables for the Create Job Keys activity are automatically populated from the settings in the orchestration properties. Use the Map Inputs checklist task of the Create Job Keys activity to populate the Job Key variables.

Tip: The Create Job Keys activity runs only once in any orchestration. If two Create Job Key activities are executed in any orchestration, the second activity has no effect.

3.8 Functions and lookup tables

Studio provides a set of built-in functions that you can use to alter the data that is processed by an integration solution. These functions are used by the mapping editor to manipulate the input and output parameters of activities within an orchestration. The process for using functions is described in 3.9.2, “Using functions in maps” on page 141

You can access these built-in functions from the Functions toolbox tab. The functions are grouped as shown in Figure 3-56 on page 124. You can find a full description of each function at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/toc_mappingfunctionreference.html

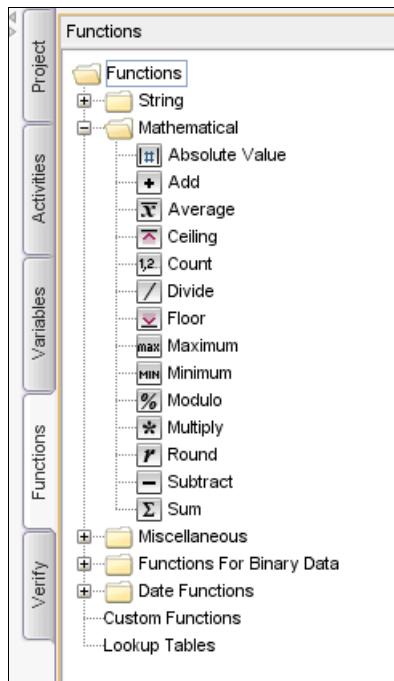


Figure 3-56 The Functions toolbox tab

3.8.1 Custom functions

Studio provides a wizard to create custom functions for use within the mapping editor. These functions are written in JavaScript. Each custom function is stored inside a project. Thus, to use a custom function in more than one project, you must create it in each project individually. After you create a custom function, you can use it in the same manner as a built-in function.

To create a new custom function:

1. Go to the Functions toolbox tab, right-click the Custom Functions folder, and click **Add New Custom Function**. The custom function wizard opens.

The first window of the wizard has fields to set the name, input parameters, and output data type of the function, as shown in Figure 3-57 on page 125. Enter the Function Name and Return Type of the function.



Figure 3-57 The Custom Function wizard, defining the function

2. Click **Add** to add each input parameter. A new parameter line displays with no parameter name and a type of string. Double-click the empty parameter name field, and enter the parameter name.
3. Select the parameter type, as shown in Figure 3-58.

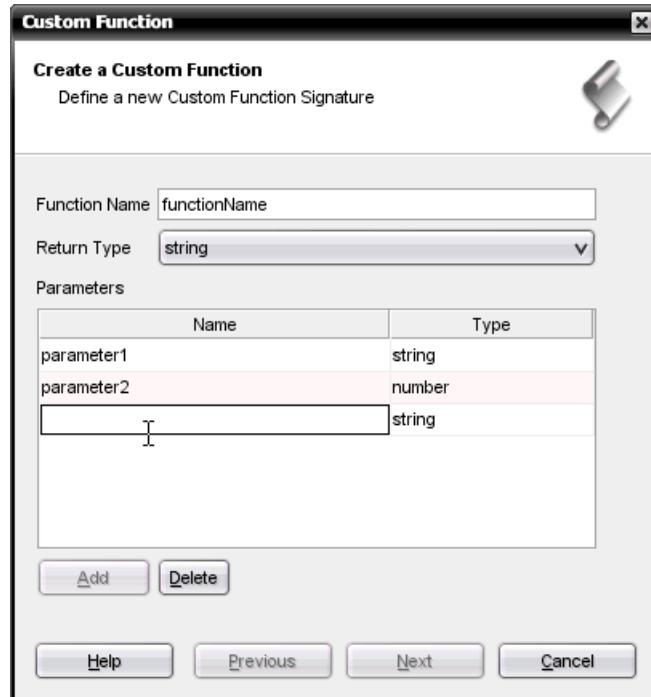


Figure 3-58 Adding input parameters to a custom function

4. Click **Next**. The wizard shows a window with a text entry field. Enter the contents of the JavaScript function in to the text field. The function declaration and the opening and closing braces are provided automatically. Figure 3-59 shows the contents of a custom function entered in to this window.

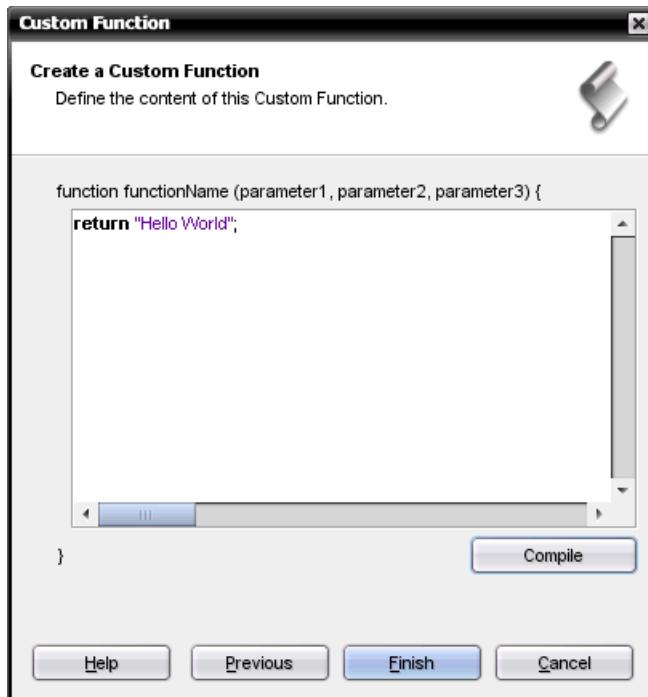


Figure 3-59 Entering the contents of a custom function in to the wizard

5. Click **Compile** to confirm that the JavaScript function can be compiled successfully.
6. Click **Finish** to save the custom function in the project.

You can edit a custom function by right-clicking the name of the custom function in the Functions toolbox tab and selecting **Edit** to relaunch the custom function wizard.

Example: You can find an example of using a custom function in Chapter 12, “Scenario: Data enrichment and aggregation” on page 465.

3.8.2 Lookup tables

A lookup table holds a set of key/value pairs. When the lookup table is called it returns the value associated with the key that is passed to it. Lookup tables are stored in projects and used by the mapping editor in the same way as functions. A lookup table is used for simple key/value pairs that are not changed regularly. You must republish the project to change the values in a lookup table. Use a database to hold key/value pairs of data that change regularly. You can use the staging database to make this lookup efficient. The staging database is discussed in 3.6.7, “Database activities, assets, and the staging database” on page 114.

To create a new lookup table:

1. Go to the Functions toolbox tab, right-click the Lookup Tables folder, and click **Add New Lookup Table**. The lookup table window opens.
2. Enter a name and description for the lookup table, as shown in Figure 3-60.

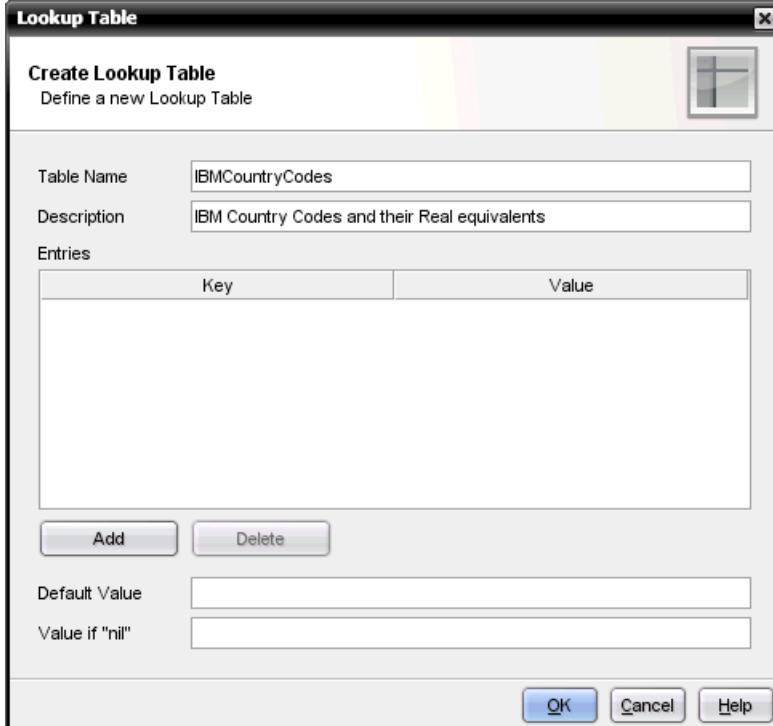


Figure 3-60 Creating a Lookup Table

3. Click **Add** to add a new key/value pair to the lookup table.
4. Double-click the empty key field, and enter the new key.
5. Double-click the empty value field, and enter the new value.
6. Click **Add** to enter more key/value pairs, and click **Delete** to remove key/value pairs as needed.
7. Enter a default value in the Default Value field. This value is used if the key passed to the lookup table cannot be found.
8. Enter a value to be returned if the key passed to the lookup table is “nil” in the Value if “nil” field.
9. When the lookup table is complete, click **OK** to save the table to the project.

Figure 3-61 shows an example.



Figure 3-61 A completed lookup table

You can edit a lookup table by right-clicking the name of the lookup table in the Functions toolbox tab and then selecting **Edit**.

Example: You can find an example of using a lookup table in Chapter 12, “Scenario: Data enrichment and aggregation” on page 465.

3.9 Maps

Activities in an integration solution can require that input parameters are set for the activity. An activity that requires input parameters has a Map Inputs task in the checklist for the activity to define data that is sent to the activity. This task uses a tool called the *mapping editor* to define how the input parameters are set.

The same mapping editor is used on the Map Outputs task for activities that return results to define the data that can be used by other activities in the orchestration. In this case, the mapping editor defines where the results of the activity are stored.

Not all activities require input parameters, and not all activities return results. Some activities have both input and output parameters. Figure 3-62 on page 129 shows a representation of an activity that requires input parameters and returns results.

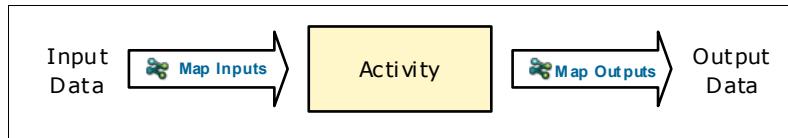


Figure 3-62 An activity that requires input parameters and returns results

You can map data in variables directly to and from activity parameters using the mapping editor. You can give parameters and variables default values or use functions to set the values. You can use default values with mappings and functions together as well. In this case, the default value is only used if the mapped value or function output is empty. Functions are described in 3.8, “Functions and lookup tables” on page 123.

Figure 3-63 shows a representation of an input map for an activity that uses direct mapping, default values, and functions.

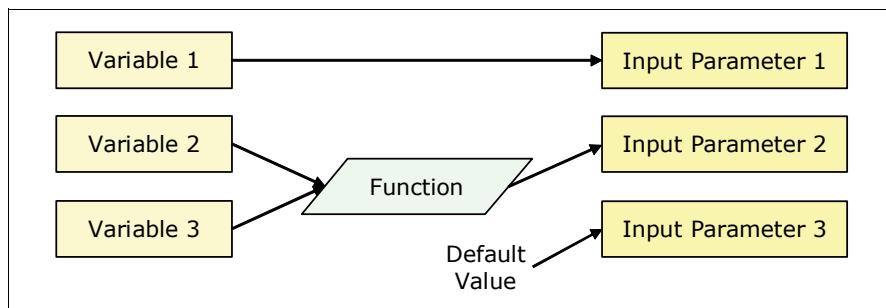


Figure 3-63 An input map that uses direct mapping, default values and functions

You can transform data into the form that is needed for each activity by using direct mapping, default values, and functions in the mapping editor. As the data passes from one activity to another in an orchestration, it is transformed in each activity until it is in the correct form that is needed by the external systems that are used by the orchestration.

3.9.1 The mapping editor

You access the Studio mapping editor by clicking either the **Map Inputs** task or **Map Outputs** task of an activity’s checklist. The mapping editor displays in the configuration pane. The mapping editor has two modes, Design or Test. Figure 3-64 on page 130 shows the Design mode of the mapping editor for the Map Inputs task of a Send Email activity. To switch to the Test mode, click **Test** at the top left of the pane. Testing maps are covered in 3.12.3, “Test maps” on page 161.

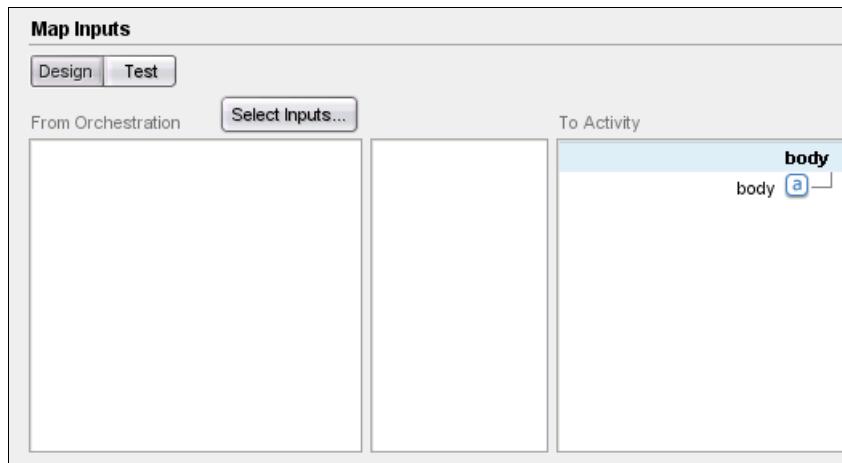


Figure 3-64 The mapping editor for the Map Inputs task of a Send Email activity

Notice in Figure 3-64 that the left pane of the mapping editor for the Map Inputs task is labelled From Orchestration and is empty. The right pane for the Map Inputs task is labelled To Activity and is prepopulated with an input parameter called body.

Figure 3-65 shows the mapping editor for the Map Outputs task of the FTP Poll Directory activity. This mapping editor has a left pane labelled From Activity that is prepopulated with output parameters. The right pane in the mapping editor for the Map Outputs task is labelled To Orchestration and is empty.

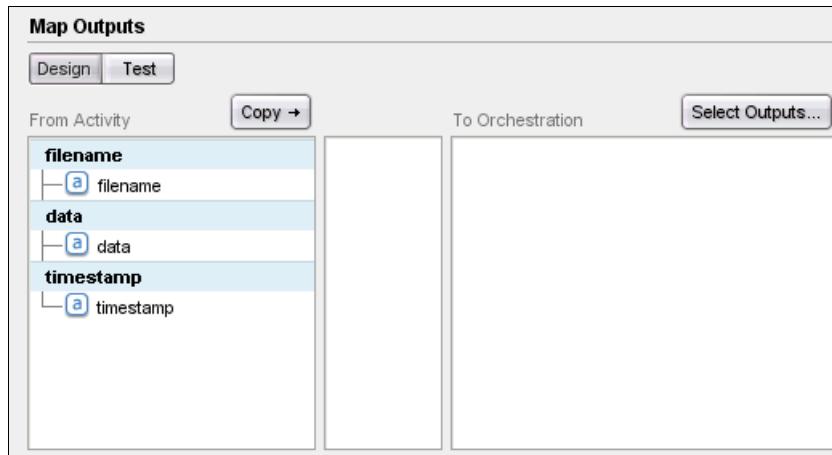


Figure 3-65 The mapping editor for the Map Outputs task of an FTP Poll Directory activity

The Map Inputs and Map Outputs tasks are used in similar ways. The Map Inputs task displays input parameters in the To Activity pane on the right and variables in the From Orchestration pane on the left. The Map Outputs task displays output parameters in the From Activity pane on the left and variables in the To Orchestration pane on the right. Other than this difference, they are the same. Mappings are always performed from the left pane in both the Map Inputs and Map Outputs tasks to the right pane.

Parameters

Each activity automatically populates the list of input and output parameters in the Map Inputs and Map Outputs tasks of that activity. Activities that use external systems interrogate the

external systems at development time to discover the input and output parameters that are required for the activity.

Activities can have optional parameters that are not shown by default. Right-click in the From Activity or To Activity panes, and select **Show Optional Parameters** to show these optional parameters. If the Show Optional Parameters option is not available when you right-click in the pane, no optional parameters are available for the activity you are configuring.

Figure 3-66 shows the To Activity pane of an HTTP Send Response activity after the optional parameters are enabled. The optional parameters are marked as *optional*.

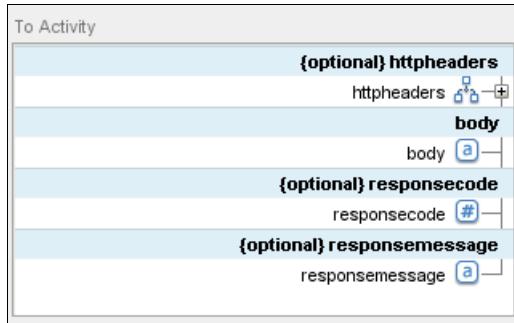


Figure 3-66 The optional and mandatory input parameters of an HTTP Send Response activity

Optional parameters can include routing, proxying, and security information for an HTTP request, web services security headers for a web service invoke, connection and session information for Salesforce.com, and so forth.

Adding variables

You must select the variables that will be used in the From Orchestration and To Orchestration panes. To select an existing variable:

1. Click **Select Inputs** for the Map Inputs task and **Select Outputs** for the Map Outputs task. The Select Inputs or Select Outputs window opens. The Select Inputs window is shown in Figure 3-67. Both windows are used in the same way.

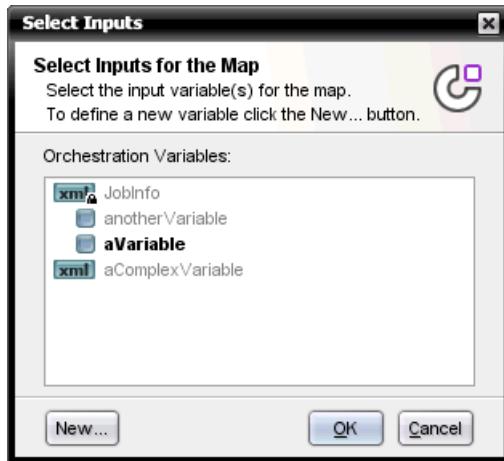


Figure 3-67 The Select Inputs window of the mapping editor

2. Click the variable that you want to add, and then click **OK**.

Variables that are already added are highlighted in bold. The variable *aVariable* in Figure 3-67 was added. The icon to the left of each variable indicates whether the type for

the variable is a primitive type or a complex type. The blue square in Figure 3-67 on page 131 indicates that the variable has a primitive type. The larger blue rectangle with *xml* on it indicates that the variable has a complex type.

Tip: The JobInfo variable is set automatically by the orchestration and cannot be written to. The padlock symbol on its type icon indicates this. The JobInfo variable is described in 3.7.4, “The JobInfo variable and Job Keys” on page 122.

Figure 3-68 shows the From Orchestration of an activity with the variables *aVariable* and *aComplexVariable* added to it. Notice that more than one variable can be added to the From Orchestration pane, which means that values from fields in more than one variable can be combined to populate fields in an input parameter.

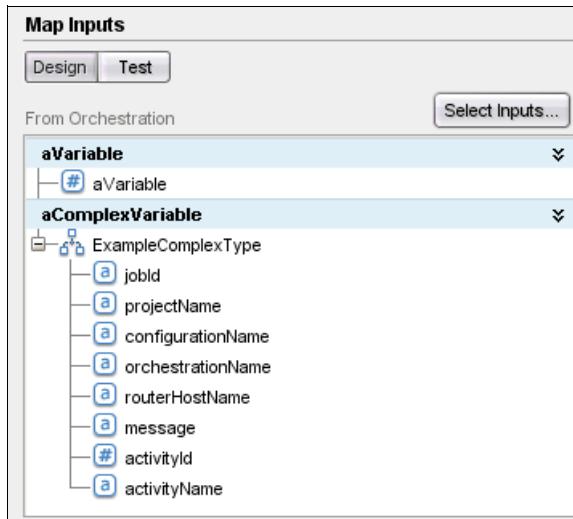


Figure 3-68 The From Orchestration pane of an activity with *aVariable* and *aComplexVariable* added

The name of the variable displays in a blue area, as shown in Figure 3-68. The data contained inside that variable shows as a set of *nodes* in the white area underneath the variable name.

The data for a primitive variable shows as a single field node underneath the variable name in the white area with the same name as the variable. The data type of the primitive variable is indicated by an icon to the left of the field node. In Figure 3-68, the variable *aVariable* is a primitive variable. The field node for the variable is also called *aVariable*. The data type icon for this variable displays a number sign (#), which indicates that this variable is a number.

In Figure 3-68, the variable *aComplexVariable* is a complex variable. The variable contains nodes representing the different fields in the complex variable. Each of the fields can have different data types. For example, the field *projectName* is of type string and the field *activityId* is of type number. The data type of a complex variable displays as a record node immediately under the variable name. In Figure 3-68, the data type of the variable *aComplexVariable* is *ExampleComplexType*.

Figure 3-69 shows a From Orchestration pane displaying a complex variable with a more complicated structure. The variable name is ordersParsedData, and the data type is Orders.

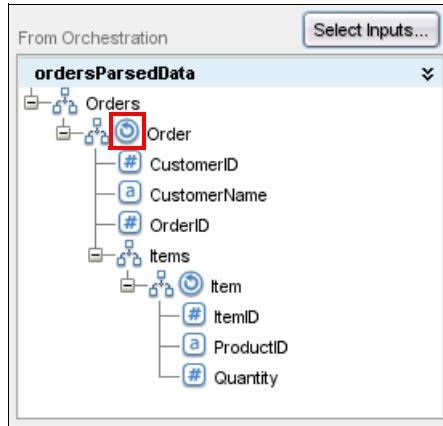


Figure 3-69 A variable with a complex data type in the From Orchestration pane

The Orders data type contains a set of orders, indicated by the repeating node icon highlighted in Figure 3-69. The Orders data type also contains a record node called Items, which has a complex type.

A variable can contain a large number of nested nodes. However, it is important to note that the variable is still displayed in the blue area. Everything in the white area is contained within this variable. This concept is important because mappings are performed on the nodes for the variable that are displayed in the white area. The variable in the mapping editor is used only if you need to remove the variable from the mapping editor. Right-click the blue area that contains the variable name, and select **Remove Variable** to remove the variable, as shown in Figure 3-70.

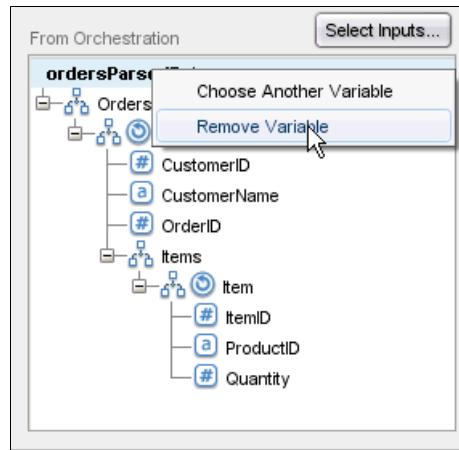


Figure 3-70 Removing a variable from the mapping editor

Right-clicking the variable name and selecting **Choose Another Variable**, as shown in Figure 3-70, allows you to choose another variable of the same data type to use instead of the variable that you right-clicked.

Tip: Pressing CTRL+ALT+K while in the mapping editor enables and disables some advanced XML schema options for the menus that appear when you right-click variables and nodes.

The following options are enabled when you right-click a variable:

- ▶ View Resolved DOM Tree
- ▶ View XML Schema Source

The following options are enabled when you right-click a node:

- ▶ View XML Schema Type Definition
- ▶ View XML Schema Fragment
- ▶ View MetaData for Selected Node

Simple mapping

To define a simple mapping, drag a field node from the left pane on to a field node of the same data type on the right pane, as shown in Figure 3-71. The field node for the primitive variable anotherVariable is dragged to the field node for the primitive variable body.

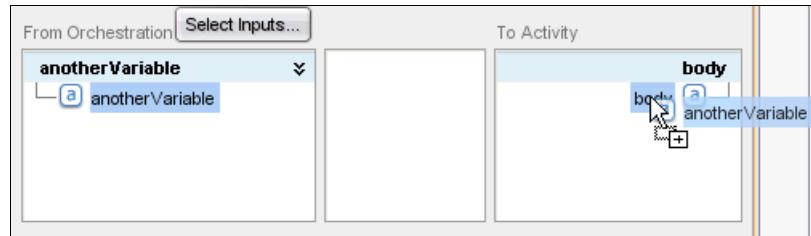


Figure 3-71 Defining a simple mapping

When a mapping is defined, a line displays that connects the two nodes, as shown in Figure 3-72.

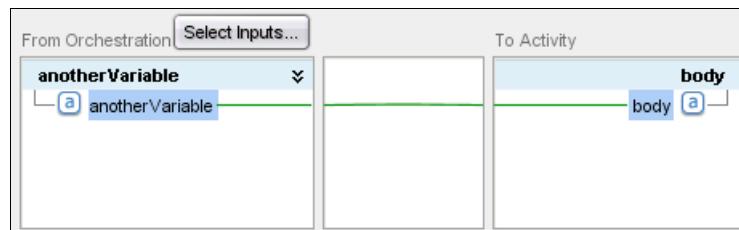


Figure 3-72 A simple mapping has been defined

You can map a node on the left pane to multiple nodes on the right pane, which maps the value of the node in the left pane to all of the nodes in the right pane. However, a node on the right pane can be mapped only to one node on the left pane.

Guaranteed mapping

The data types of the two nodes that are mapped must be the same to guarantee the mapping. The mapping editor allows some mappings to be defined between different data types, but this can cause an error when the orchestration is executed. Error handling is discussed in 3.11, “Exception handling” on page 155. When a mapping is defined that is not guaranteed, a dialog window opens warning that the mapping is not guaranteed and asking you if you want to define the mapping. Figure 3-73 on page 135 shows a mapping that is defined between a string and a number.

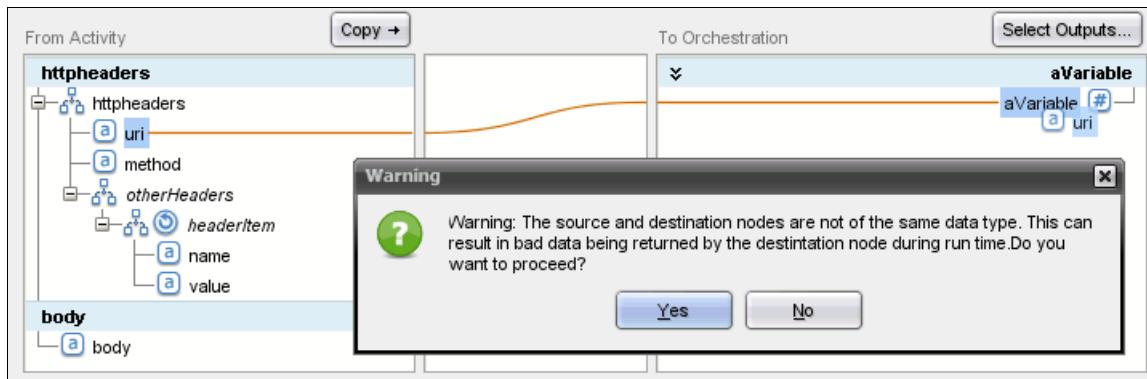


Figure 3-73 A mapping that is not guaranteed

The color of the line representing the mapping in Figure 3-73 is orange. In guaranteed mappings the line is green. The mapping lines are also different colors if a function is used. Using functions for mapping is discussed in 3.9.2, “Using functions in maps” on page 141.

The mapping editor line colors are as follows:

Green	A straight guaranteed map
Blue	A guaranteed map using functions
Orange	A straight map that is not guaranteed
Magenta	A map using functions that is not guaranteed

Copy button

The Map Outputs task contains an extra button labelled Copy, as shown in Figure 3-74. This button creates a new variable of the same name and the same data type as one of the output parameters of the activity. It also defines a simple mapping for all nodes of the output parameter to all nodes of the new variable.

Output parameters are the pieces of data returned from the Activity. They are not automatically saved even though they are given names. They are like variables that are local to the Activity alone. In Figure 3-74, we copy the contents of these output parameters to variables that are available outside of the Activity.

The Copy button:

- ▶ Creates a variable that is available outside the activity.
- ▶ Gives the new variable the same name as the local parameter.
- ▶ Copies the value of the local parameter into the new variable.



Figure 3-74 The Map Outputs Copy button

Click **Copy** to open the Copy Parameters window shown in Figure 3-75. Select the output parameter that you want to copy, and click **Create**.

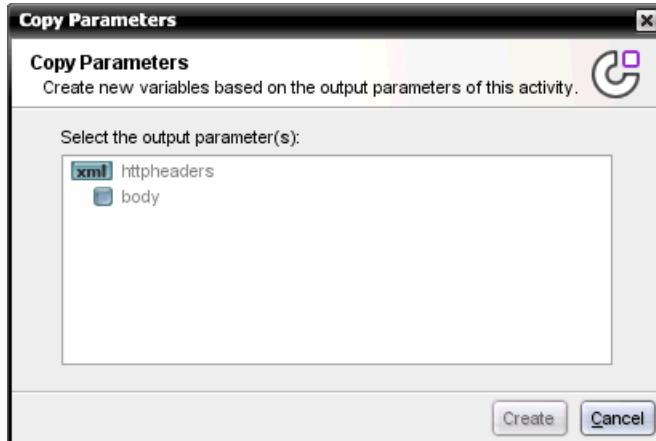


Figure 3-75 The Copy Parameters window

A variable is created and a simple mapping to this variable is defined, as shown in Figure 3-76. In this example, the variable *httpheaders* was selected.

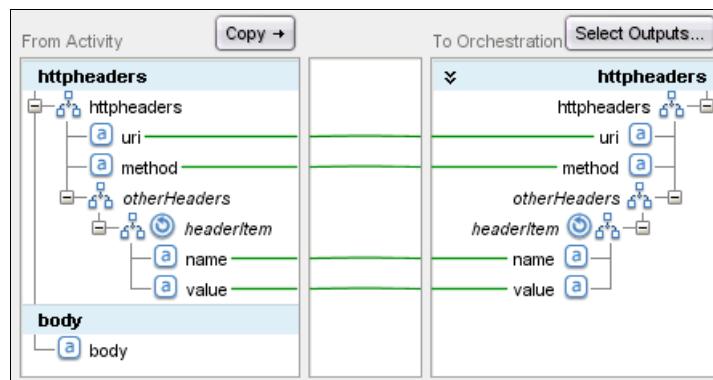


Figure 3-76 The Copy button created the variable *httpheaders* and defined simple mappings

Notice that all of the nodes of the output parameter *httpheaders* are mapped, including the repeating fields. The new variable called *httpheaders* displays in the To Orchestration pane on the right. Rename this new variable using the Variables toolbox tab.

Renaming variables: Do not rename a variable by left-clicking the variable name in the blue area and over-typing a new variable name. This action creates a second variable of the same type with the new name. This action does not rename the original variable, which will still exist. See the Tip box on 120.

Bulk mapping and automapping

You can define mappings for multiple nodes at the same time using either bulk mapping or automapping, which saves time during development. There is no difference in how the mappings work at runtime.

If a source node has a complex data type and therefore contains child nodes, you can drag the source node to a destination node with the same structure and the mapping editor will define a map for every child node. This is called bulk mapping. In Figure 3-76, the following

nodes for the httpheaders output parameter in the From Activity pane have complex data types and therefore have child nodes:

- ▶ httpheaders
- ▶ otherHeaders
- ▶ headerItem

A bulk mapping is defined if the httpheader's node, in the From Activity pane in Figure 3-76 on page 136, is dragged to the httpheaders in the To Orchestration pane that has the same structure. As shown in Figure 3-76 on page 136, all of the children of the httpheader's source node are mapped to the corresponding children of the httpheader's destination node.

If the headerItem node in the From Activity pane is dragged to the headerItem node in the To Orchestration pane, only the children of headerItem are mapped, as shown in Figure 3-77. These mappings can be created by dragging each source child node to the corresponding destination child node individually, but using bulk mapping can be much quicker for nodes with large numbers of children.

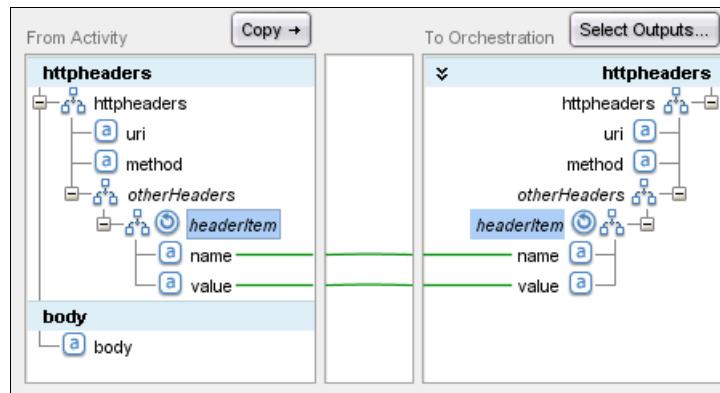


Figure 3-77 The children of headerItem are mapped using bulk mapping

The mapping editor can also perform automapping. Automapping scans the field nodes in the left pane and creates a map to one field node in the right pane with the same name. Case and punctuation are ignored. If there are two or more field nodes of the same name in either pane, automapping cannot be performed. To use automapping, click **Map → Automap**.

Automapping defines the mappings in Figure 3-76 on page 136 and not those in Figure 3-77, where the uri and method fields were left unmapped.

Recurring node mapping

Recurring nodes are nodes that contain lists of data, for example a list of customers or addresses. A recurring node can have a complex data type. The node headerItem in Figure 3-77 is a recurring node with a complex data type. It contains a list of name/value pairs.

Recurring nodes have special mapping options. You map one occurrence of a repeating node to a non-recurring destination node as follows:

1. Right-click the repeating node, and then click **Select One Occurrence**. The Select Occurrence window shown in Figure 3-78 on page 138 opens.
2. Enter the number of the occurrence to be mapped, and then click **OK**.



Figure 3-78 The Select Occurrence window for a fixed occurrence

3. Drag the children of the repeating node to a non-repeating destination field node, as shown in Figure 3-79, to define simple mappings for those field nodes.

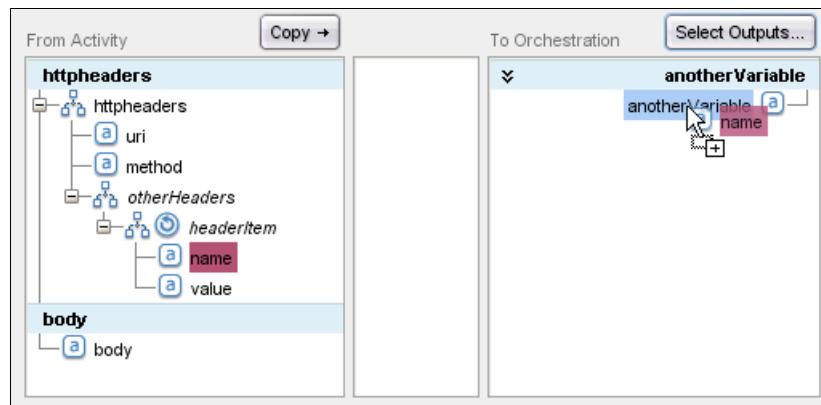


Figure 3-79 Defining a simple mapping for one occurrence of a repeating field

You can use a variable to define the number of the occurrence that is mapped as follows:

1. Right-click the repeating node, and then click **Select Occurrence When**. The Select Occurrence window shown in Figure 3-80 opens.
2. Enter a variable to define the occurrence that is selected.



Figure 3-80 The Select Occurrence window for a variable occurrence

You can also map individual occurrences of a repeating destination node:

1. Right-click the destination node in the right pane, and select **Expand Occurrences**.
2. Enter the number of repeating occurrences to expand, and then click **OK**.

In Figure 3-81 on page 139, four occurrences of the headerItem repeating node in the To Orchestration pane are expanded, allowing each of these occurrences to be mapped

individually. The fifth occurrence of headerItem, highlighted in Figure 3-81, represents the remaining occurrences of the repeating node, which you can still map as well.

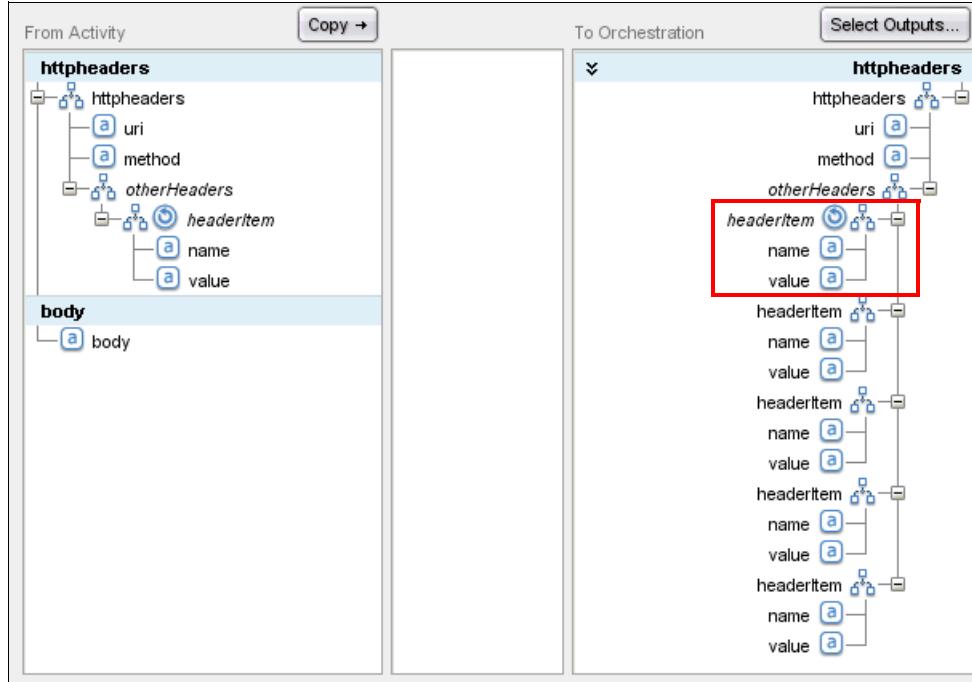


Figure 3-81 Four occurrences of the headerItem repeating field are expanded

Remove repeating occurrences by right-clicking the occurrence to be removed, and then clicking **Remove Selected Occurrence**.

You can also filter the data being passed to a recurring destination node. For example, if a recurring source node contains a list of orders, this list can be filtered by the size of each order so that large orders only are mapped to the recurring destination node. The list of data in the destination node contains only the orders that meet the filter condition. The filter condition is defined using an XPath predicate, which is the XPath form of an if statement. To filter the mapping for a recurring destination node:

1. Map a recurring source node to the recurring destination node.
2. Right-click the recurring destination node, and then click **Filter Recurring Nodes**. The Filter Recurring Nodes window shown in Figure 3-82 opens.
3. Enter an XPath predicate, and then click **OK**.

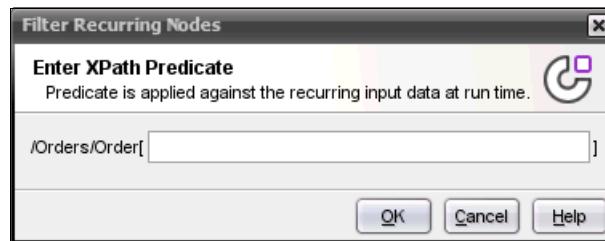


Figure 3-82 the Filter Recurring Nodes window

4. The XPath predicate is checked. If it is valid, a filter icon is displayed next to the recurring destination node.

An example of filtering a destination recurring node is described in Chapter 10, “Scenario: Bidirectional account synchronization” on page 357.

The information center contains a table that shows how to create mappings between different combinations of recurring and non-recurring nodes:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/map_Mapping_Recurring_Nodes.html

Additional property nodes

Additional property nodes for some variables and parameters are defined automatically by the mapping editor. An example of an additional property node is the occurrence node, which holds the number of occurrences of a repeating node.

By default, additional property nodes are hidden. To display additional property nodes, right-click in either the left or right panes, and then select **Show Additional Property Nodes**. Any available additional property nodes for all variables and parameters in both panes will display, as shown in Figure 3-83. These nodes are mapped like other nodes.

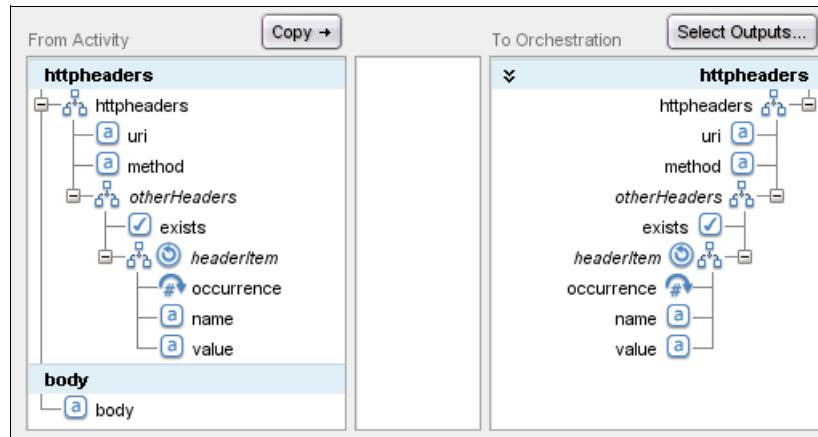


Figure 3-83 Additional Property Nodes

Default values

A destination field node can be given a default value. If no value is set for the destination node using a mapping, the node is set to the default value. If a value is set by a mapping, this value is used.

You can set a default value for a destination field node:

1. Right-click the field node, and then select **Define Default Value**. The Define Default Value window opens.
2. Enter a default value, and click **OK**. The default value icon displays against the field node name to indicate that a default value is set for that field node, as highlighted in Figure 3-84.



Figure 3-84 The default value icon

Tip: You can set the default value of a destination node using a configuration property. The value of this configuration property can then be changed by an administrator using the WMC. Configuration properties are described in 3.3.2, “Configuration properties” on page 88.

Deleting a mapping

To delete an existing mapping, right-click the mapping line, and select **Delete Mapping**, as shown in Figure 3-85. Deleting an existing map removes any default values that are set for the destination field nodes.

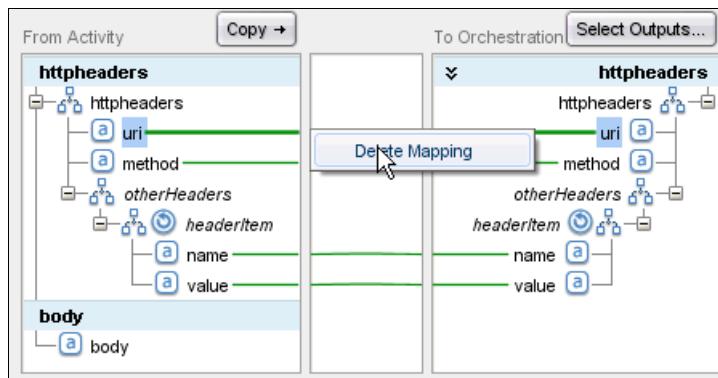


Figure 3-85 Deleting a mapping

You can delete default values for a destination field node without deleting mappings to the field node by right-clicking the field node and then selecting **Define Default Value**. Remove the existing default value so that the Default Value property is blank, and click **OK**.

3.9.2 Using functions in maps

Studio provides a set of built-in functions that are used by the mapping editor. You access the built-in functions from the Functions toolbox tab. To use a built-in function:

1. Drag the function that you want to use from the Functions toolbox tab to the middle pane of the mapping editor, as shown in Figure 3-86 on page 142.

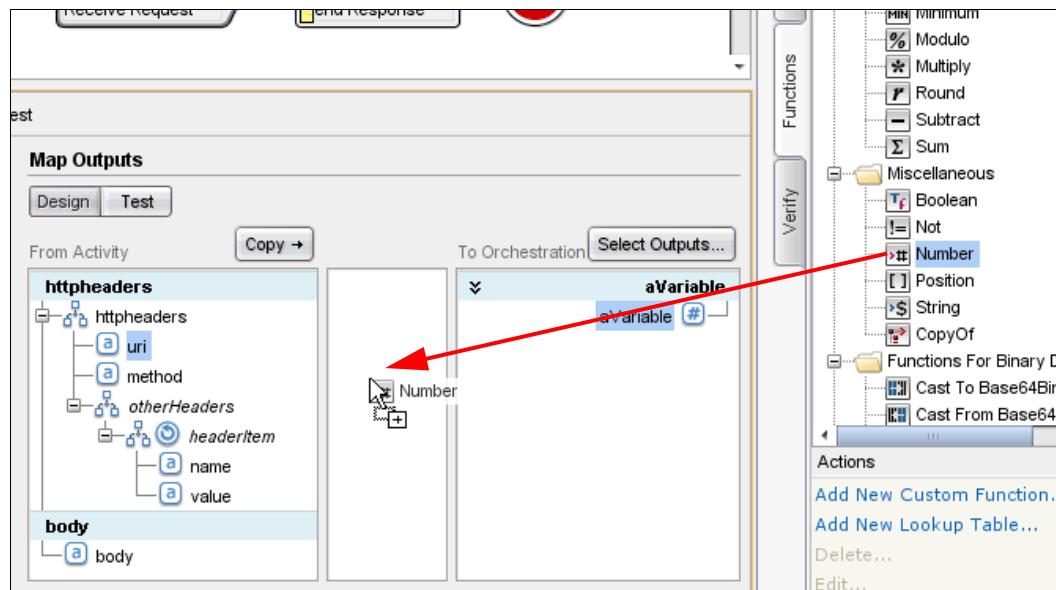


Figure 3-86 Using a function in the mapping editor

The Number function in Figure 3-86 takes a string input parameter and converts it in to a number. As an example, this allows you to convert the uri field node to a number to store the value of that field node in the aVariable field node.

2. The Number function requires an input parameter. To set the input parameter using a field node, drag the field node to the icon representing the function in the middle pane, as shown in Figure 3-87.

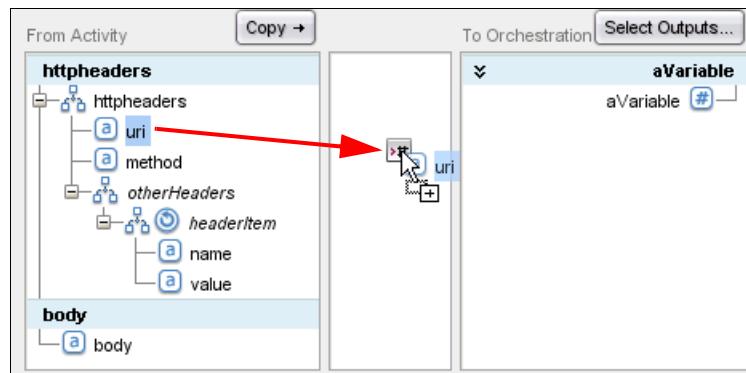


Figure 3-87 Setting the input parameter of a function to a field node value

A line is drawn from the field node to the function, as shown in Figure 3-88 on page 143.

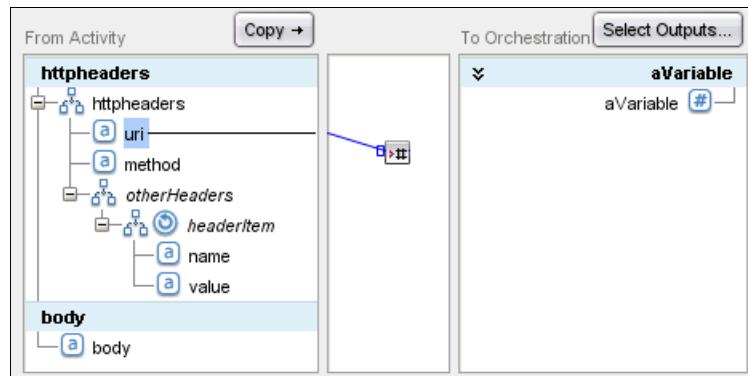


Figure 3-88 A completed input parameter mapping from a field value node

3. Define the mapping of the result of the function by dragging the function to a destination field. You can chain functions together by setting the result of one function to be an input parameter of the next function, as shown in Figure 3-89, where an Add function is dragged to the middle pane, and then the Number function is dragged to the Add function. A line is drawn between the functions to show that they are chained.

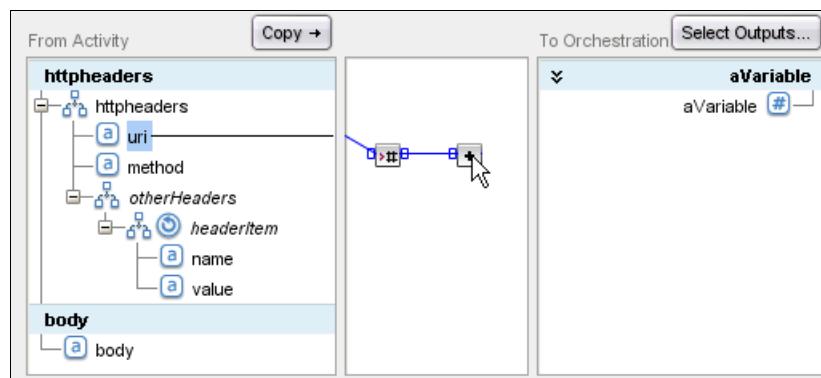


Figure 3-89 Chaining the Number and Add functions

4. After you create all of the parameters and mappings for a function or chain of functions, right-click anywhere in the middle pane, and then click **Apply Function Graph**, as shown in Figure 3-90.

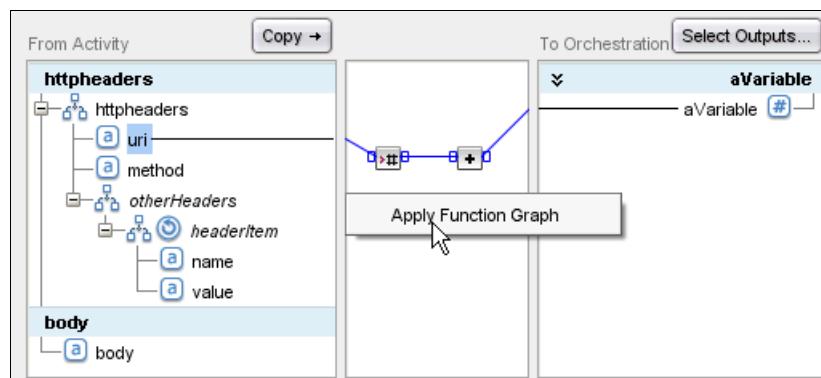


Figure 3-90 Apply Function Graph

Tip: You can create only one chain of functions at any one time and map only one destination field node from any one chain of functions. To create another chain of functions to map to another destination field node, first Apply Function Graph for the current chain.

5. A mapping line displays between the source nodes and the destination node. The line is colored blue for a guaranteed mapping and is colored magenta for a mapping that is not guaranteed. A function icon also displays next to the destination field node, as shown in Figure 3-91. The function icon looks like a scroll and is highlighted in the figure.

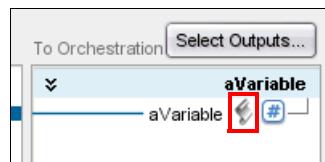


Figure 3-91 The “scroll” function icon

Mapping: The mapping line that displays for the function mapping in this example is blue, meaning that the mapping is guaranteed. However, the mapping that is defined takes a uri from an HTTP request and uses the Number function to attempt to convert it to a number. This action will fail because the input parameter cannot be converted to a number.

Guaranteed mapping means that the data types of all input and output connections in a mapping chain are the same. It does not mean that the mapping will not fail; therefore, include error handling for maps, even if all the mappings are guaranteed.

To edit the function mapping to the destination field node, right-click the field node, and then select **Edit Function Graph**.

Functions that do not need input (*Get Current Date and Time function*)

The Get Current Date and Time function is an example of a function that does not need input parameters. In Figure 3-92, the output result of the function is connected to the anotherVariable field node.

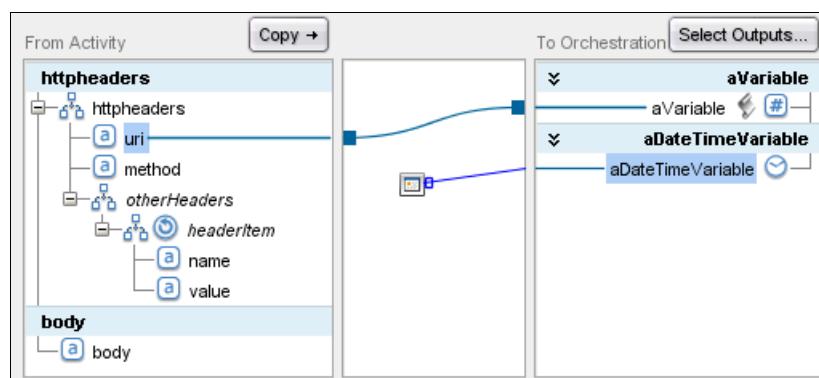


Figure 3-92 A function that does not need input parameters

When Apply Function Graph is selected for the Get Current Date and Time function, no mapping line is displayed for the mapping. Only the scroll function icon displays next to the name of the field node, as shown in Figure 3-93 on page 145.

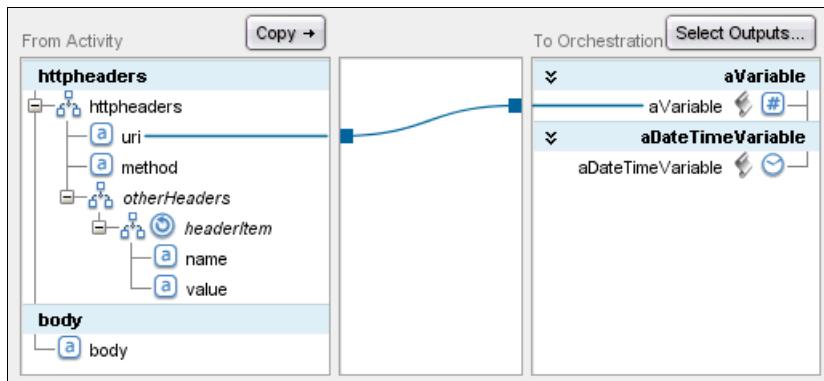


Figure 3-93 No mapping line is displayed for a function that does not have input parameters

Tip: You can drag a function on to an existing simple mapping without needing to delete the mapping first. A mapping is defined where the function takes the source node as an input parameter and a destination node as the output parameter.

Hard coding input parameters

You can hard code input parameters for functions instead of using values from field nodes. To hard-code an input parameter, double-click the icon that represents the function in the middle pane, or right-click, and click **Show Parameters**. The Function Parameters window opens.

Concatenate function

Concatenate is another example of a common function, shown in Figure 3-94. This function concatenates string input parameters together.

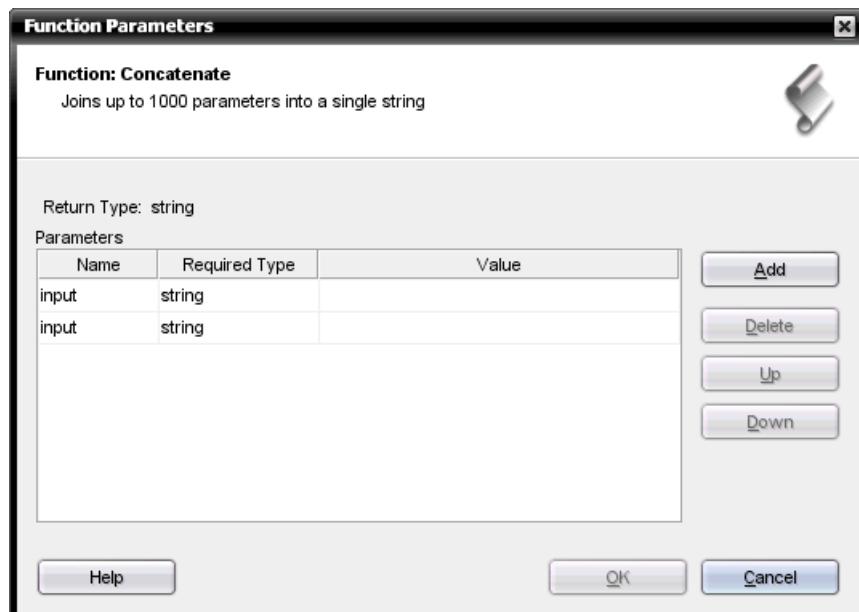


Figure 3-94 The Function Parameters window for the Concatenate function

Double-click the fields in the value column, and add hard-coded values for the input parameters. Some functions, including the Concatenate function, allow additional input parameters to be added. To add an input parameter, click **Add**, and enter the input parameter value. After you enter all input parameter values, click **OK**.

3.9.3 The Map Variables activity and standalone maps

Studio includes an activity that maps input variables to output variables, called the Map Variables activity. This activity is in the Transform activity category. The activity does not have a checklist, an Input Maps task, or an Output Maps task. The activity contains only one task to complete, which uses the mapping editor, as shown in Figure 3-95.

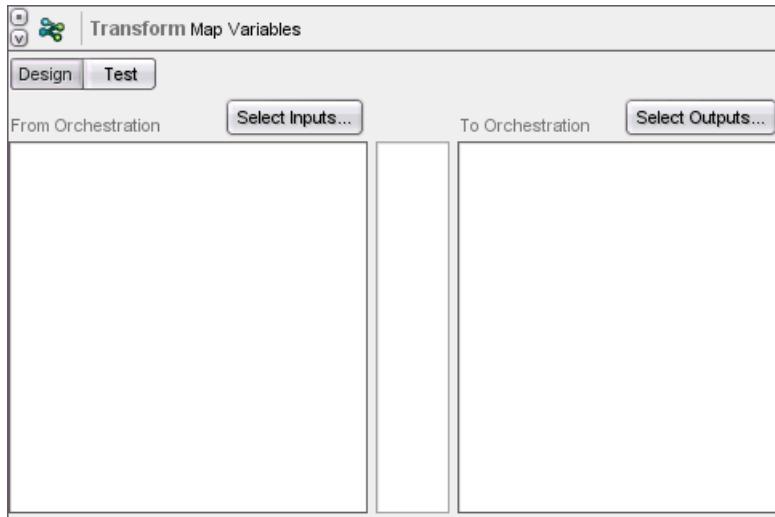


Figure 3-95 The Map Variables activity does not have a checklist

The activity is used in an orchestration in the normal way. When the activity executes, the mappings that are defined in the activity are performed using the input and output variables that are added to the activity.

The mapping editor for a Map Variables activity is the same as the mapping editor for all other activities. Simple mapping, default values and functions can be used in a Map Variables activity. Multiple input and output variables can be added to a Map Variables activity. Figure 3-96 shows an example of some mappings in the mapping editor for a Map Variables activity.

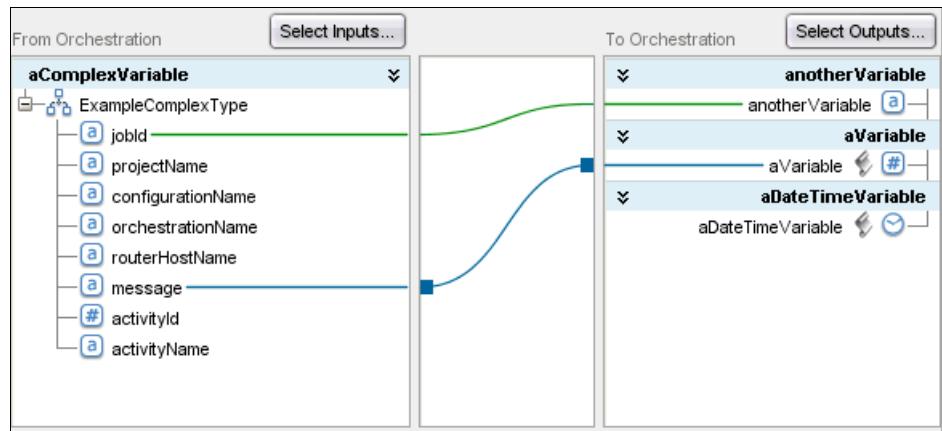


Figure 3-96 A Map Variables activity with some mappings

You might need to perform a mapping from one complex data type structure to another complex data type structure in several places in one orchestration, or in several orchestrations in one project. For example, the structure of an address record held in one

system must be mapped to the structure of an address record held in another system after several different data extracts at different stages throughout an orchestration.

You can redefine the mapping between the address structures every time the mapping needs to be performed. Alternatively, you can define a standalone map, which is used to create multiple Map Variables activities that perform the same mapping. This process saves time during development and makes sure that every Map Variables activity created from the standalone map uses the same mapping.

Mappings for a standalone map do not use variables for the input and output nodes. Instead, they use complex data types. You must define these complex data types in Studio before you create the standalone map. You can define complex data types using XML schema files and WSDL files. Add these files to Studio on the Project toolbox tab by right-clicking **XML Schemas** or **WSDLs** and then selecting **Add Document**. You can also define complex data types using Flat File Schemas, which are described in 3.10, “Parsing data” on page 149.

To create and use a standalone map:

1. Right-click **Transformations** in the Project toolbox tab, and then select **New Standalone Map**. A standalone map is created with a default name.
2. Right-click the newly created standalone map, and select **Rename**. Give the map a suitable name.
3. Double-click the new standalone map to open the mapping editor.
4. Add input and output complex data types to the mapping editor by clicking either **Select Inputs** or **Select Outputs**. The **Browse For Schema Type Element** window shown in Figure 3-97 opens. Select a record node, and then click **OK**. An example of a record node is highlighted in the figure.

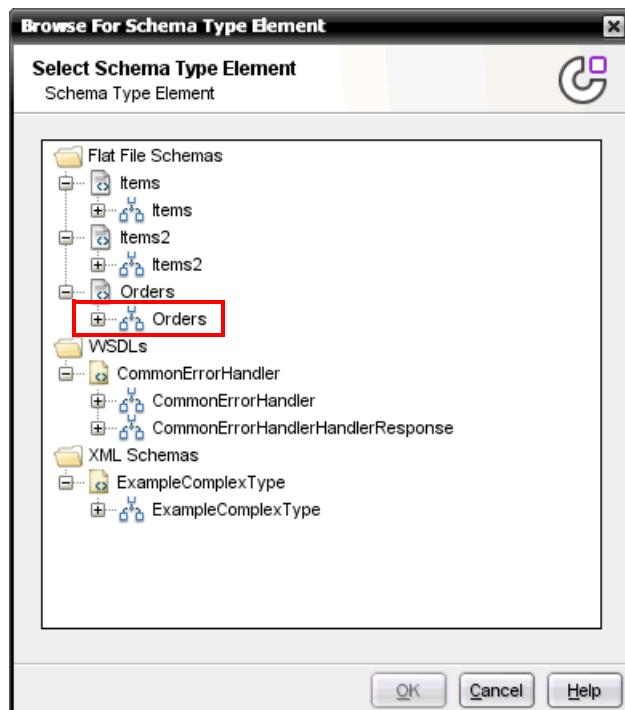


Figure 3-97 The *Browse For Schema Type Element* window

- Define mappings using the mapping editor. You can use simple mapping, default values, and functions in a standalone map. Figure 3-98 shows an example of a standalone map with some mappings.

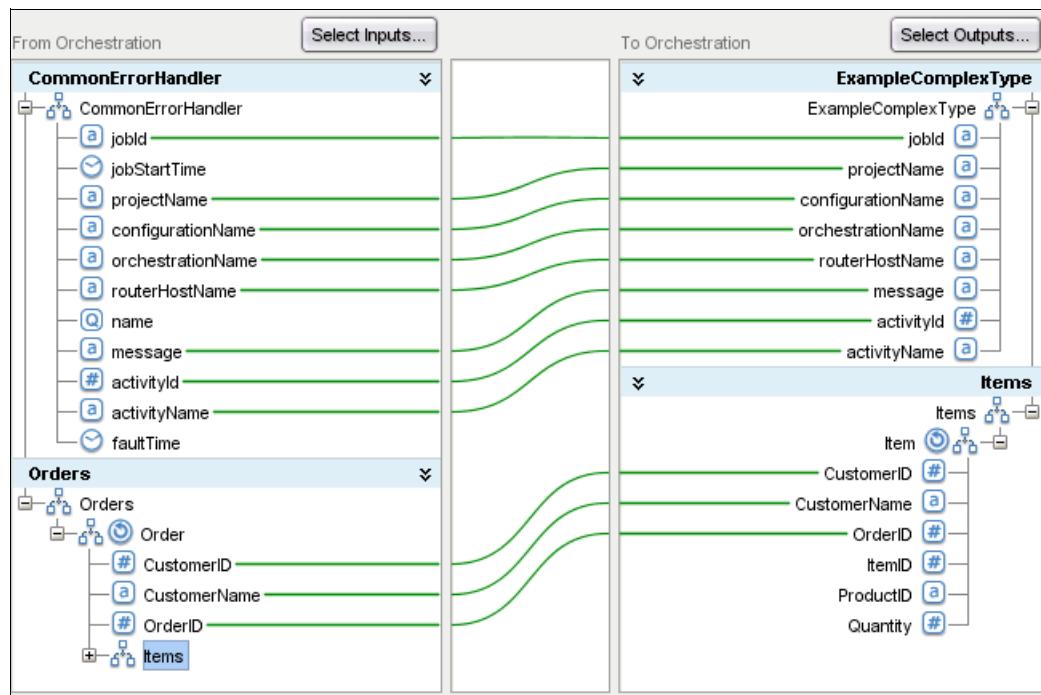


Figure 3-98 Some mappings in a standalone map

- Open the orchestration where you want to use the standalone map. Drag the standalone map from the Project toolbox tab to the place in the orchestration where it will be executed. A Map Variables activity is created using the mapping that is defined in the standalone map.
- The Link Schema Types To Variables window shown in Figure 3-99 opens. Use the drop-down boxes in the Orchestration Variables column to select existing variables of the correct data types that are used by the Map Variables activity that is created, and then click OK.

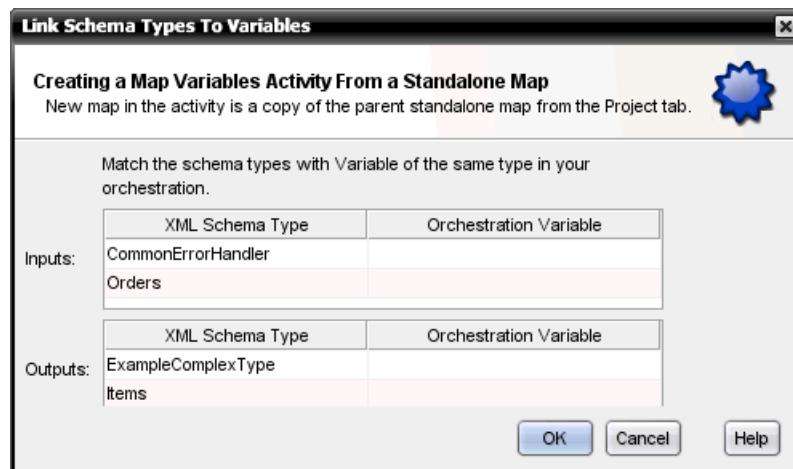


Figure 3-99 The Link Schema Types To Variables window

You can alter the mappings in the Map Variables activity that you create if needed for the particular use of the standalone activity. The original standalone activity is not changed.

3.9.4 XML stylesheets

XML stylesheets are an XML-based way of transforming data. They operate on XML documents. Because Cast Iron Integration Solutions use XML schema types to hold variable data types, XML stylesheets can be used to transform data in an orchestration.

Studio does not contain an XML stylesheet editor. If you want to use an XML stylesheet, you must create it outside of Studio, and then add it to a project after the stylesheet is written. To add an XML stylesheet to Studio, right-click **Stylesheets** in the Project toolbox tab, and then select **Add Document**.

To use an XML stylesheet in an orchestration:

1. Drag the Apply XSLT activity from the Transform category of the Activities toolbox tab to the orchestration.
2. Select the **Pick Stylesheet** task in the checklist, and then click **Browse**. Select the XML stylesheet from the window that opens, and click **OK**.
3. Select the **Set Input & Output** task in the checklist. Click **Input → Browse** to select the input variables that are used, and click **Output → Browse** to select the output variables.

You can test XML stylesheets in Studio, as described in 3.12.4, “Test Custom XSLT” on page 163.

3.10 Parsing data

Data from an external system is often passed to a Cast Iron Integration Solution as a set of variables with a predefined structure, for example, the Database Poll Table activity queries the database at development time. The activity finds the columns in the table that are being polled. The Map Outputs for the activity has an output parameter that contains field nodes representing the columns in the table that were returned, as shown in Figure 3-100. When the orchestration runs, the data that is extracted from the table populates each field node for the output parameter. Each field node has the correct data type for the data.

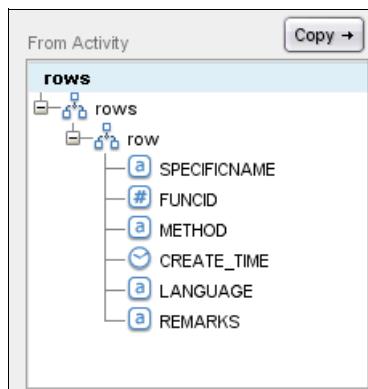


Figure 3-100 The Map Outputs task of a Database Poll Table activity

The data that is extracted from the table is parsed, meaning that the data was interpreted intelligently to make sense of the data. This parsing allows the remainder of the orchestration

to use the individual field nodes for the data rather than having to use all of the data as a whole, as shown in Figure 3-101.

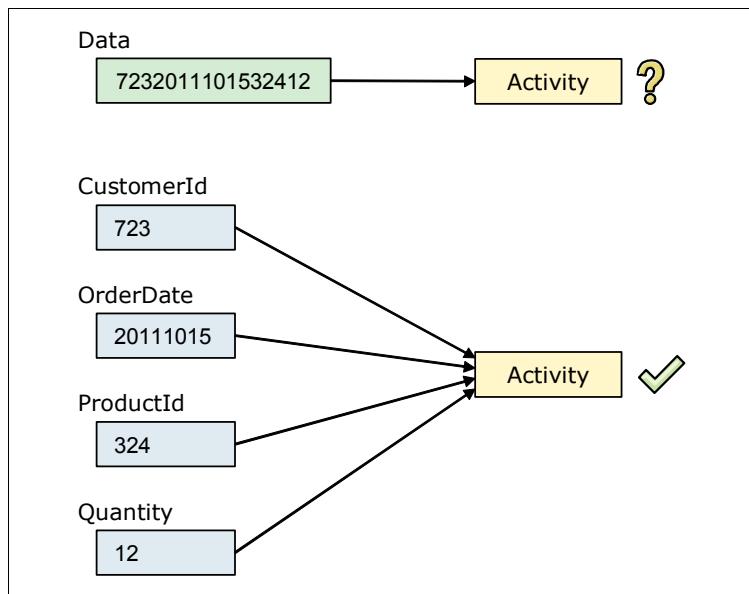


Figure 3-101 The advantage of parsing data

Some data that is received from external systems is not parsed automatically. For example, the MQ Poll Message activity returns an output parameter called payload of data type string that contains the whole body of the MQ message. To make sense of this data, it must be parsed.

Cast Iron Integration Solutions can use the following techniques to parse data manually:

- ▶ Maps and functions
- ▶ XML parsing
- ▶ MIME parsing
- ▶ Flat File Schema parsing

3.10.1 Maps and functions

You can use functions in the mapping editor to parse data, for example, you can use the Substring function to extract part of a string. Thus, one string can be parsed into a set of substrings that contain the parsed data. Other functions that you can use to parse data include Trim, Base64 Decode, and Read Date String. You can also use custom functions to parse data.

This approach is not encouraged because it is slow to develop and inflexible. However, for some data, this method is the only technique that successfully parses the data.

3.10.2 XML parsing

If the data that is parsed is a stream of XML data, you can use the Transform activity Read XML to parse the data. This activity takes a single string as its input parameter and returns a variable containing the parsed XML structure. To use this activity, you must select an existing variable that is of the same data type as the XML data that is going to be parsed.

There is also a Write XML activity that takes a complex variable and outputs a string in the XML format containing the data held in the complex variable.

3.10.3 MIME parsing

Multipurpose Internet Mail Extension (MIME) is a standard that allows several types of data to be sent in the same input stream along with information that allows the input stream to be parsed. It is often used to hold attachments in emails.

If the input data being sent to an integration solution uses the MIME standard, it can be parsed using the Transform activity Read MIME. This activity can currently be used only with the Get Email activity, which is described in the information center:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/email_get_activity.html

There is also a Write MIME activity that is used with the Send Email activity.

3.10.4 Flat File Schema parsing

An integration solution can use a *Flat File Schema* to parse a string intelligently. You can use Flat File Schemas to parse a string that is received from any source. The string does not have to be read from a file, for example, the MQ Poll Message activity mentioned earlier in this chapter returns a string that can be parsed using a Flat File Schema.

The data in a string parsed by a Flat File Schema can have the following structures:

Delimited	The fields in the string are separated from each other by a delimiter character
Positional	The fields in the string are of fixed length and appear in the string at defined positions
Combination	Some fields in the string are delimited and some are of fixed length

Creating a Flat File Schema using the Flat File Wizard

Studio provides a wizard that takes sample data and generates a Flat File Schema from the data. The wizard can generate only a delimited Flat File Schema. The sample data must contain one record per line, with the fields in each record separated by a single character, such as a comma.

To run the Flat File Wizard:

1. Select **Project → Flat File Wizard**.
2. Click **Browse** to select some sample data, and then click **Next**.
3. Select the separating character. The wizard displays the parsed sample data. You can also set the wizard to ignore the first and last rows of the data.

Creating a Flat File Schema manually

The Flat File Wizard cannot generate Flat File Schemas from most sample data; therefore, you can build a Flat File Schema manually. To create a Flat File Schema without using the wizard:

1. Right-click **Flat File Schemas** in the Project toolbox tab, and then select **New Flat File Schema**. A window opens asking for the name of the schema.

- Enter a name, and click **OK**. The Flat File Schema editor opens in the workspace, as shown in Figure 3-102.

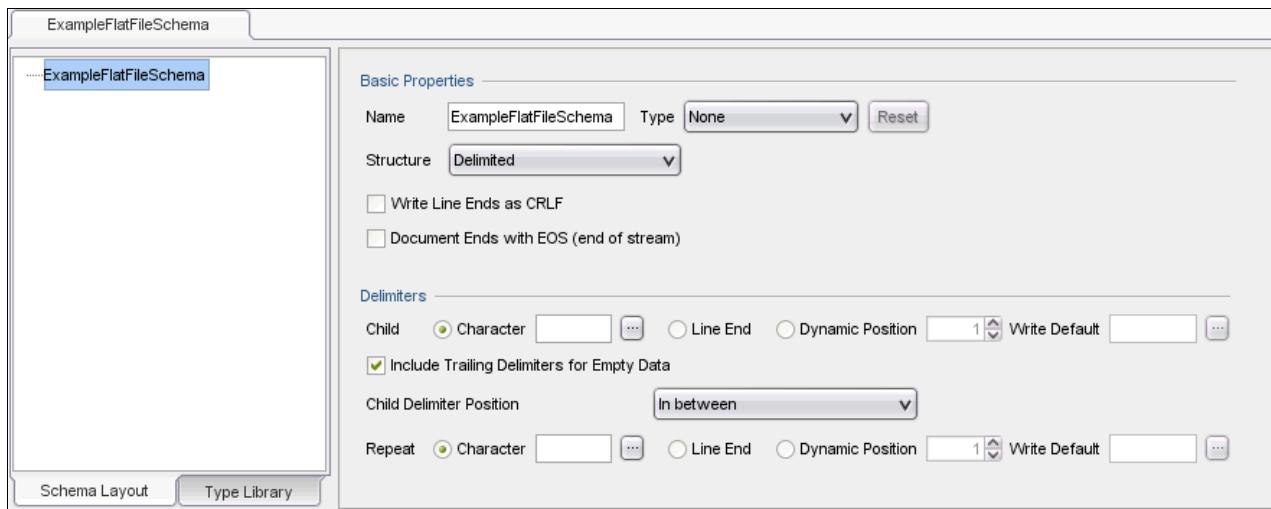


Figure 3-102 The Flat File Schema editor for schema ExampleFlatFileSchema

The Flat File Schema editor also contains a testing tool that is displayed in Studio below the panes shown in Figure 3-102. This tool is described in 3.12.2, “Test Flat File Schema” on page 160.

The name of the Flat File Schema in Figure 3-102 is ExampleFlatFileSchema. The white pane on the left of the Flat File Schema editor is the Schema Layout pane. A new schema contains only one node in its Schema Layout which is the *Root Node*. Every Flat File Schema has exactly one Root Node. In Figure 3-102, the Root Node is called ExampleFlatFileSchema. The Root Node is given the same name as the Flat File schema by default, but you can change this name using the node properties.

The properties for each node that is selected in the Schema Layout pane displays on the right of the Flat File Schema editor. In Figure 3-102, the Root Node is selected. So, the properties that display are for the Root Node.

The Root Node contains parsing settings for the entire stream of data or document that is being parsed, for example, the Root Node properties can require that the document ends with an end of stream character. The Root Node properties also describe how the child nodes of the Root Node are parsed, for example, are the children of this node delimited or are they of fixed length (positional)? These parsing properties of the Root Node are the same as those on the Record node.

The Root Node can have the following child nodes:

Field	A single field containing a value of a primitive data type
Record	A node containing other child nodes which can be Fields or Records
Group	A more flexible type of record

Tip: Groups are required only when parsing a document that contains repeating records, where the document does not clearly define the start and end of each record.

You can find more information about groups at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/mde_Adding_Groups_or_Group_Types.html

To add a child of a node:

1. Right-click the node.
2. Select **New Child** and then **Field**, **Record**, or **Group** as shown in Figure 3-103.
3. Enter the name of the new child, and click **OK**.

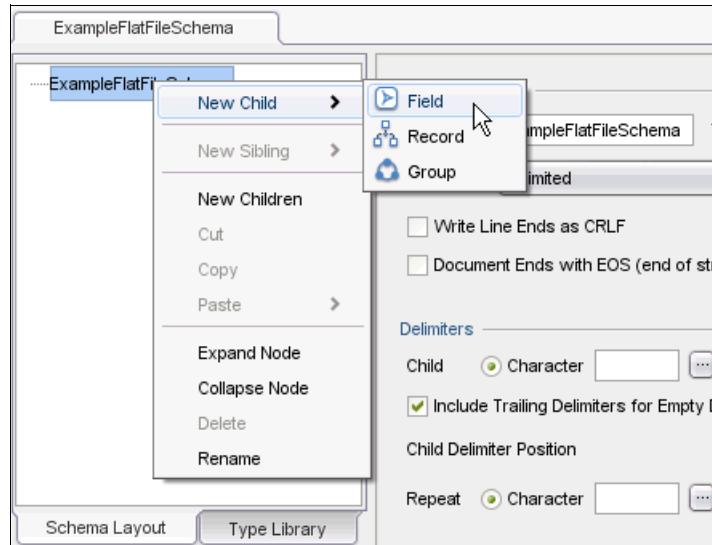


Figure 3-103 Adding a child to a node in the Flat File Schema editor

Flat File Schema node properties

Figure 3-104 shows the properties of a Record node. The record defines the structure of its children with the Structure property, which you can set to either Delimited or Positional. If the Structure is set to Delimited, the Delimiters section defines the separator characters that will be used for the children of the record (Child) and for repeated occurrences of the record (Repeat). The Record node properties define the position of the delimiter for the children of the node with the Child Delimiter Position drop-down property. This position can be either before each child, in between the children, or after each child.

The screenshot shows the 'Basic Properties' tab of a Record node's properties dialog. It includes fields for Name (ExampleRecord), Type (None), Structure (Delimited), Record Identifier, Occurrences (Minimum 1, Maximum 1, Range), and Delimiters (Child Character, Repeat Character). The 'Delimited' structure is selected, so the 'Delimiters' section is active, showing options for Child and Repeat delimiters, including 'Include Trailing Delimiters for Empty Data' and 'Child Delimiter Position' set to 'In between'.

Figure 3-104 The properties of a Record node

The properties also define how many occurrences of the node are valid. You can define explicit minimum and maximum occurrences of the node. You can also make the occurrences

of the node unbounded, meaning that there is no maximum number of occurrences of the node. You can set a node as unbounded only if its parent node has a structure of Delimited and if you have set a valid delimiter in the parent node.

Figure 3-105 shows the properties of a Field node. You can set the field node to repeat a minimum and maximum number of times, or it can be unbounded. Again, you can set a node as unbounded only if its parent node has a structure of Delimited and if you have set a valid delimiter in the parent node.

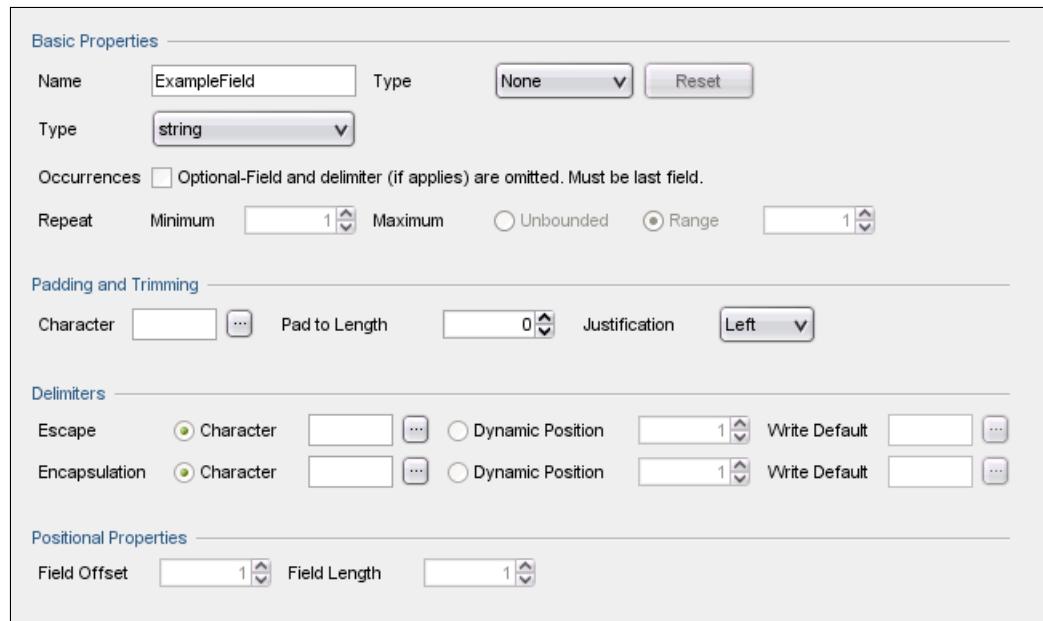


Figure 3-105 The properties of a Field node

The field node has a data type property, which means that data in the field node is treated as that primitive type. If data for the field node is not of the correct type, parsing fails. The Positional Properties control how the data for the field node is parsed if the parent Record node was set to Positional structure.

The Delimiters section controls how the field node can handle characters that are part of the data but were also used as separators. For example, if a record contains a product ID, a product name and a quantity separated by a slash character (/), the parser has difficulty parsing a product name of OS/2.

There are two ways to solve this problem. You can define an escape character, such as a caret symbol (^), so that any character immediately after the escape character is not treated as a delimiter, for example:

123/OS^2/14

Alternatively, you can surround or encapsulate the data with a pair of characters, such as quotation marks (" "), and any delimiter character found between those characters is ignored, for example:

123/"OS/2"/14

The Delimiter Escape and Encapsulation sections contain these properties.

You can find a full description of the properties for Root nodes, Group nodes, Record nodes, and Field nodes in the information center:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/toc_flatfileschemas.html

A complex data type is created when you create a Flat File Schema. You can use this complex data type as the data type for a variable. The name of the complex data type that you create is the same as the name of the Root node of the Flat File Schema. The names and positions of the nodes in the Flat File Schema are the same as the names and positions of the nodes in the complex data type.

You can change the names of all nodes in the Flat File Schema and the positions of the nodes using the Flat File Schema editor. Rename a node by right-clicking the node in the Schema Layout pane and then selecting **Rename**. Change the position by dragging the node to its new position in the Schema Layout pane. The complex data type that you created from the Flat File Schema is updated dynamically.

Tip: If the complex data type for any variable is changed or updated, any mappings to and from that variable might be invalid. Check all mappings using a variable after changing its data type.

You can edit a Flat File Schema after you create it by double-clicking the Flat File Schema in the Project toolbox tab. You can also edit a Flat File Schema that is created using the Flat File wizard in the same manner.

Tip: SAP uses a data structure called *IDOC*. Studio includes a tool for generating a Flat File Schema from an SAP IDOC, as described in the information center:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/Generating_a_Flat_File_Schema_from_an_IDOC.html

Using a Flat File Schema

Flat File Schemas are used by the Transform activities Read Flat File and Write Flat File. Remember that the data for a Flat File Schema does not have to come from a file. Both activities have a Configure task on their checklist where you select the Flat File Schema to use.

The Read Flat File activity takes an input parameter called Data of data type string and returns a parsed complex data type of the type defined by the Flat File Schema. The Write File activity takes an input parameter of the structure of the complex data type defined by the Flat File Schema and returns a single string, which is the data constructed in the form defined by the Flat File Schema with all of the Delimiters and Padding Characters (for positional data) inserted.

You can find an example of creating and using Flat File Schemas in Chapter 12, “Scenario: Data enrichment and aggregation” on page 465.

3.11 Exception handling

All orchestrations must contain logic to deal with unexpected situations to make sure that an orchestration cannot fail or lose data without either recovering automatically or creating a notification. An unexpected situation in an orchestration can be indicated when an activity

throws an exception, which is also called a *fault*. However, it is important to be aware that not every unexpected situation causes an exception to be thrown, for example, Salesforce.com only throws an exception if there is a problem with the login process or with the external ID being used. All other unexpected situations are indicated by the content of the response message from Salesforce.com.

When designing an orchestration, be aware of the type of exception that an activity will throw and any status codes that are returned in the response. Create logic in an orchestration to validate the response message to discover if an unexpected situation occurs.

Tip: An activity that uses an endpoint has retry settings to allow the activity to automatically retry connecting to the external system without the need for explicit logic. These retry settings are configured in the checklist for the activity and must be altered to meet the specific needs of the integration solution.

Another example where you check the status of results is a Database Update Rows activity. The activity fails if the activity cannot connect to the database but can also fail if the number of rows updated is zero. In this case, an exception is not thrown. Check the Get Row Counts option in the activity's checklist to return the number of rows updated, and then add an If..Then activity to the orchestration to check whether zero rows were updated.

The remainder of this section looks at Cast Iron Integration Solution exception handling activities.

3.11.1 Try activity

Exception handling in an orchestration uses the Try activity. After you drag the Try activity to an orchestration, you can drag other activities inside the Try activity, as shown in Figure 3-106.

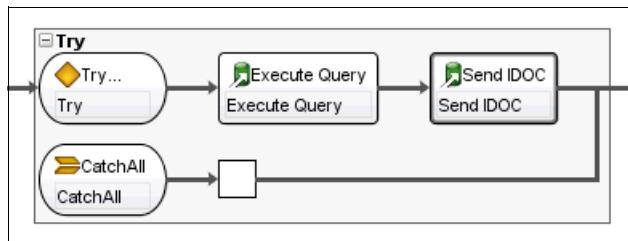


Figure 3-106 A Try activity with other activities dragged inside it

The activities inside the Try activity are attempted. If any of the activities fails with an exception, an exception message is written to the system log, and the execution flow moves to the CatchAll path. All other activities in the Try path are skipped.

Activities that you drag to the CatchAll path are executed when there is an exception. You can use these activities to handle the exception. Some examples are shown in Figure 3-107 on page 157.

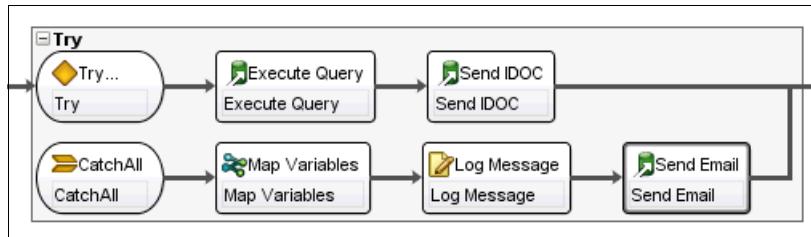


Figure 3-107 Some examples of activities in a CatchAll path

Variables that contain exception information are initialized in the CatchAll path. Because these variables are initialized inside the CatchAll path, they are not visible outside of that scope. The following variables are initialized:

FaultName	The name of the exception
FaultData	For Web Service activity exceptions only, the error message for the exception
FaultInfo	A set of fields containing exception information, including the exception name

You can give these variables specific names by clicking the CatchAll activity. The faultInfo variable contains the faultName and extra information. Thus, the faultInfo variable is almost always used. The faultInfo variable contains the following fields:

- ▶ name
- ▶ message
- ▶ activityId
- ▶ activityName
- ▶ faultTime

The activityId is a system generated number that uniquely identifies the activity where the exception occurred.

You can include an If..Then activity in the CatchAll path to check the faultName and provide specific functionality based on the exception that occurred, as shown in Figure 3-108.

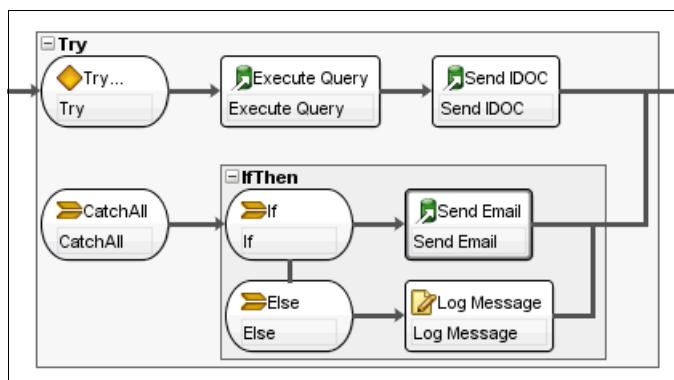


Figure 3-108 An If..Then activity included in the CatchAll path

You can create a specific Catch path by right-clicking the Try activity and then selecting **Add Catch Branch**. Figure 3-109 on page 158 shows an example of a Try activity with a specific Catch path.

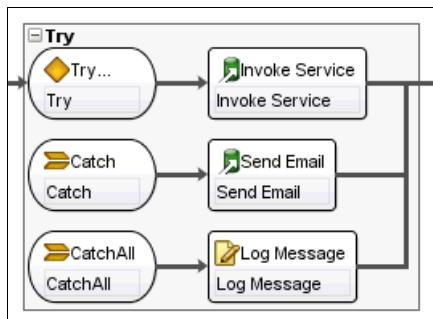


Figure 3-109 A Try activity with a specific Catch path

Specific Catch paths catch only web service faults. Use them only when you include a Web Services Invoke Service activity in the Try path, as shown in Figure 3-109. The WSDL for the web service that is being invoked must declare a fault. You then select this fault by clicking the Catch activity and selecting the fault from the FaultName drop-down property.

3.11.2 Global exception handler

You define a global exception handler for an orchestration by right-clicking the green play icon and then selecting **Add CatchAll Branch**, as shown in Figure 3-110.

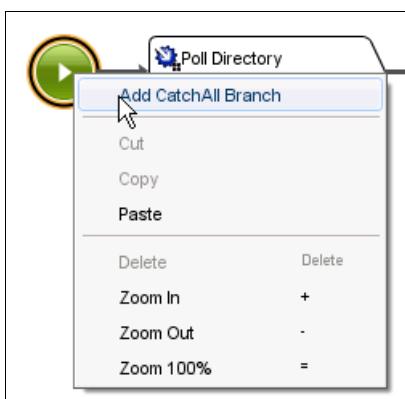


Figure 3-110 Adding a Global Exception Handler to an orchestration

This global exception handler catches any exceptions in the orchestration that are not caught by specific Try activities. Figure 3-111 shows an orchestration with a global exception handler. You can also attach a global exception handler to a Group activity by right-clicking the Group activity and then selecting **Add CatchAll Branch**. All uncaught exceptions in the Group activity are caught by the CatchAll branch.

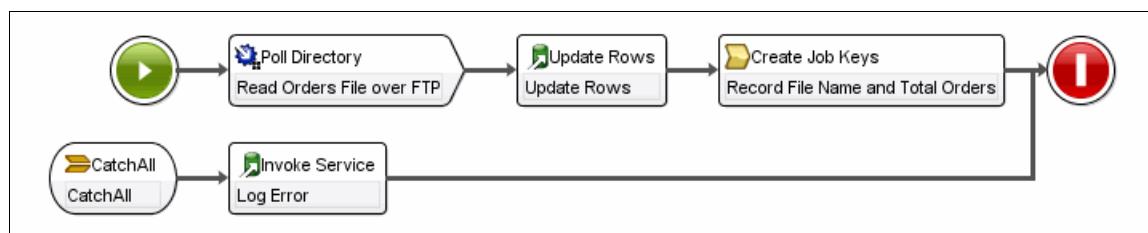


Figure 3-111 An orchestration with a Global Exception Handler

3.11.3 Common Error Handler

The CatchAll path in Figure 3-111 on page 158 invokes a web service. This web service can contain common error handling that is used by multiple orchestrations on the same Cast Iron runtime. The Common Error Handler is described in Chapter 9, “Common error handlers” on page 349.

3.12 Unit testing

Studio can test orchestrations and orchestration components locally without publishing a project to a Cast Iron runtime. Studio simulates the Cast Iron operating system so that orchestrations can be tested as though they were running on a Cast Iron runtime and any issues can be found and fixed. Studio needs a connection to all external systems used by an orchestration to fully test the orchestration and its components.

Tip: Studio cannot access the Cast Iron Secure Connector that is used by Cast Iron Live to access an external system running in a private network behind a firewall. Therefore, to test an orchestration that uses an external system running in a private network, you must provide Studio with a direct connection in to this private network so that it can access the external system at testing time.

When you test a project in Studio, you can publish it to a Cast Iron runtime with the expectation that there will be no issues with the project. The system log level for an orchestration executing on a Cast Iron runtime is changed using the WMC as discussed in 4.3.2, “The system log repository” on page 200.

The following sections describe the testing tools that are provided by Studio.

3.12.1 Endpoint Test Connection

Endpoints provide a tool to make a test connection from Studio to the external system that is defined by the endpoint. To test the endpoint, complete the connection properties, and then click **Test Connection** at the bottom of the endpoint pane. The Test Connection button is disabled if any mandatory connection properties are not provided.

The following endpoints do not allow a test connection:

- ▶ MQ
- ▶ WebServices
- ▶ HTTP for an endpoint Location of Integration Appliance Receives Request

The test connection checks only that the connection properties in the endpoint are defined correctly to allow Studio to connect to the external service. That does not mean that an activity using the endpoint can use the external service. For example, a database endpoint might be created with a user that does not have access to write to the database. In this case, the test connection successfully connects to the database, but an Insert Rows activity using the endpoint fails.

3.12.2 Test Flat File Schema

The Flat File Schema editor includes a testing tool at the bottom of the editor pane, as shown in Figure 3-112. The tool takes sample data and attempts to parse the data using the flat file schema that is defined.

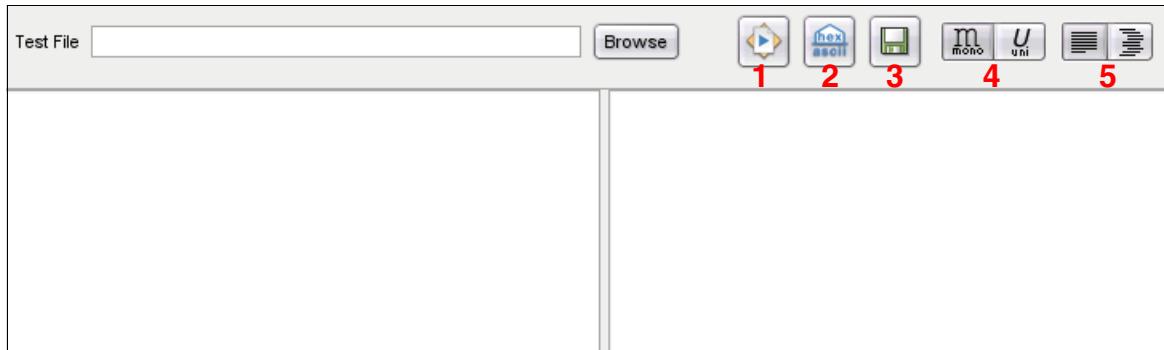


Figure 3-112 The Flat File Schema testing tool

The left pane in Figure 3-112 holds the test data, and the right pane displays the results after that data is parsed.

Enter data in the left pane by either cutting and pasting the data or using the Browse button to select data from a file. Click the Test icon, marked (1) in Figure 3-112, to run the test. A window might open displaying an error or a warning if the test failed.

Click the Display in Hexadecimal icon, marked (2), to display the input test data in hexadecimal and the Font icons, marked (4), to change the font used to display the test data and the results. The results are saved using the Save icon (3) and visually formatted in the test tool using the Format icons (5).

After the test data is parsed, click a node in the Schema Layout pane, and the corresponding sections of the test data are highlighted, as shown in Figure 3-113 on page 161.

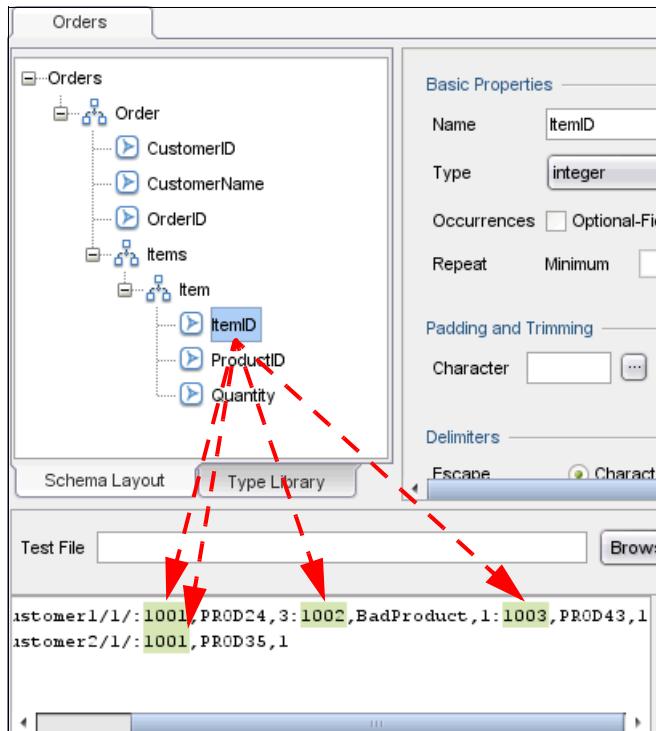


Figure 3-113 The test data elements corresponding to the selected node are highlighted

Tip: The Flat File Schema testing tool can also be used in reverse, by selecting a file containing XML of the format that the Flat File Schema produces. When the test is run, data is displayed in the right pane with the correct format needed as input to the Flat File Schema. This data can be cut and pasted into a file and used for further testing.

3.12.3 Test maps

Click **Test** in the mapping editor to switch to the Test mode, as shown in Figure 3-114. The input data for the map displays in the top section that is labelled Select Input Test Files. The bottom left pane shows the test data, and the bottom right pane shows the results after the mapping run.

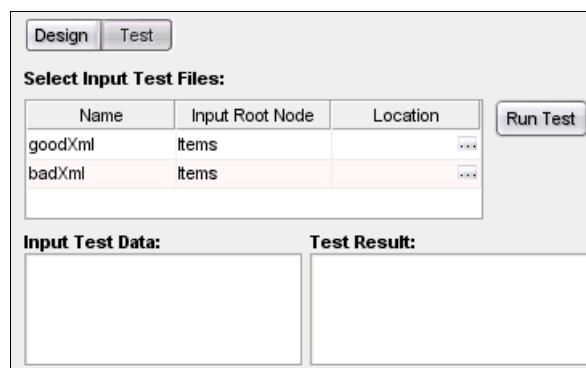


Figure 3-114 The Map test tool

You cannot paste sample data into the Input Test Data pane. You can either import data from sample files or it can be provided by Studio. Sample data that is exported from an orchestration running on a Cast Iron runtime and saved in a file is recommended. To load a sample file, click the ellipsis in the Location column for the input data that you are loading, and then select the file from the window that opens. Click **Run Test** to run the test.

If you tested the current orchestration using the Verify toolbox tab, you can import the real execution data for the activity that is associated with the map that you are testing using the Assign Execution's Results icon in the Verify toolbox tab. Verifying orchestrations is described in 3.12.7, “Verifying orchestrations” on page 164.

To generate example test data in Studio, select **Map → Generate Test Data**. Click **Run Test** to run the test. Figure 3-115 shows an example of a map test that was performed using generated test data.

Select Input Test Files:			
Name	Input Root Node	Location	Run Test
goodXml	Items	...	
badXml	Items	...	

Input Test Data:		Test Result:
<pre><?xml version="1.0" encoding="UTF-8"?> - <Items xmlns=""> - <Item xmlns=""> <CustomerID xmlns="">2</CustomerID> <CustomerName xmlns="">CustomerName</CustomerName> <OrderID xmlns="">3</OrderID> <ItemID xmlns="">4</ItemID> <ProductID xmlns="">ProductID</ProductID> <Quantity xmlns="">5</Quantity> <DESC xmlns="">DESC</DESC> <PRICE xmlns="">PRICE</PRICE> <TRANSPORT xmlns="">TRANSPORT</TRANSPORT> </Item> </Items></pre>		<pre><?xml version="1.0" encoding="UTF-8"?> - <Items> - <Item> <CustomerID>2</CustomerID> <CustomerName>CustomerName</CustomerName> <OrderID>3</OrderID> <ItemID>4</ItemID> <ProductID>ProductID</ProductID> <Quantity>5</Quantity> <DESC>DESC</DESC> <PRICE>PRICE</PRICE> <TRANSPORT>TRANSPORT</TRANSPORT> </Item> </Items></pre>
<pre><?xml version="1.0" encoding="UTF-8"?> - <Items xmlns=""> - <Item xmlns=""> <CustomerID xmlns="">2</CustomerID> <CustomerName xmlns="">CustomerName</CustomerName> <OrderID xmlns="">3</OrderID> <ItemID xmlns="">4</ItemID> <ProductID xmlns="">ProductID</ProductID> <Quantity xmlns="">5</Quantity> </Item> </Items></pre>		<pre><?xml version="1.0" encoding="UTF-8"?> - <Items> - <Item> <CustomerID>2</CustomerID> <CustomerName>CustomerName</CustomerName> <OrderID>3</OrderID> <ItemID>4</ItemID> <ProductID>ProductID</ProductID> <Quantity>5</Quantity> <DESC>ERROR: PRODUCT ORDERED WAS INVALID</DESC> <PRICE>0</PRICE> <TRANSPORT>ERROR</TRANSPORT> </Item> </Items></pre>

Figure 3-115 An example of a map test that has been performed using generated test data

Tip: Cast Iron maps use generated XML stylesheets to perform the transformations. To toggle displaying the XML stylesheet for a particular map, open the map in the Test mode of the mapping editor and press CTRL-ALT-SHIFT-8.

3.12.4 Test Custom XSLT

To open the Test Custom XSLT window shown in Figure 3-116, click **Test** in the Pick Stylesheet task of the checklist for an Apply XSLT activity.

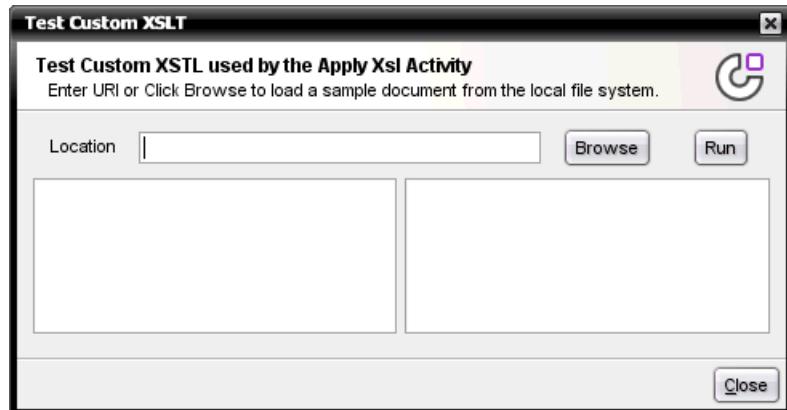


Figure 3-116 The Test Custom XSLT window

Click **Browse** to load a sample test document, and then click **Run** to perform the test.

3.12.5 XPath Evaluator

XPath expressions are used in some activities to search an XML document and return a set of data.

To open the XPath Evaluator window shown in Figure 3-117:

1. Select **Tools** → **XPath Evaluator**. Click **Browse** to load a file containing a sample XML document.
2. Enter an XPath expression in to the Test Query XPath Expression window, and click **Run**. Any XML data returned for the XPath expression against the sample data displays.

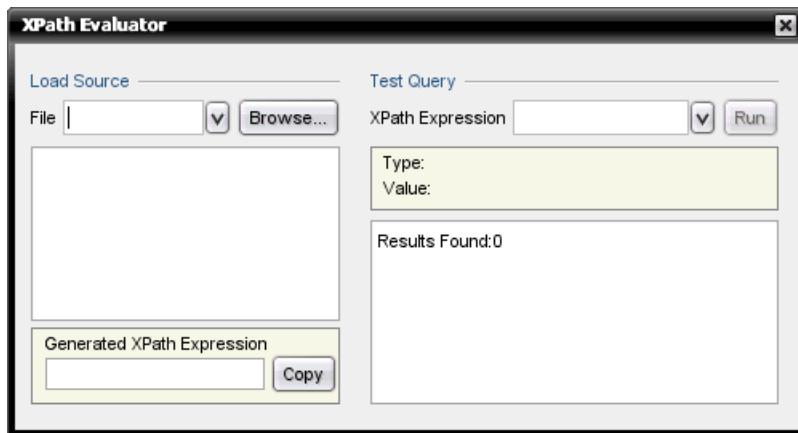


Figure 3-117 The XPath Evaluator window

You can also use the XPath Evaluator to generate an XPath expression. To generate an expression, click an XML element in the sample data window. The expression that returns the element displays in the Generated XPath Expression field.

3.12.6 Validating orchestrations

To validate a single orchestration, open the orchestration in the workspace, and then select **Orchestration** → **Validate**. You can also validate an entire project using the validate button in the toolbar, as shown in Figure 3-118.

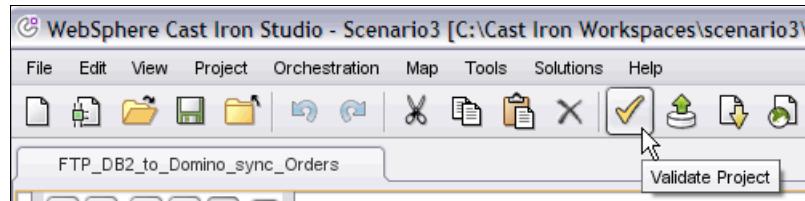


Figure 3-118 Validate an entire project

3.12.7 Verifying orchestrations

You can run a job instance for an orchestration inside Studio using the Verify toolbox tab. An orchestration must be valid to be tested in this way; however, other orchestrations in the project do not have to be valid. The orchestration that you are testing must be open and selected in the workspace.

To start a job instance for an orchestration, select the Verify toolbox tab, and then click the green Start Orchestration icon shown in Figure 3-119. One job instance for the orchestration is started.



Figure 3-119 The Verify toolbox tab icons

Tip: If the orchestration starts with a Pick activity, right-click one of the starter activities in the Pick activity, and select **Verify Orchestration From Here** to start the job instance from the starter activity and not from the Pick activity.

The orchestration job runs. While it is running, the Verify toolbox tab displays a trace of the activity that is currently executing and the activities that were already executed, as shown in Figure 3-120. To stop the job before it completes, click the red Stop Orchestration icon. The job stops automatically when it runs to completion.

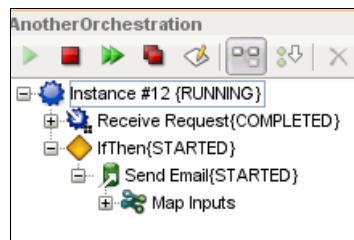


Figure 3-120 The trace data for a job running in the Verify toolbox tab

Only one job instance can execute for each orchestration in the Verify toolbox tab at a time. However, after one job instance completes, another job instance for the same orchestration can be started. Trace data for each job instance for the orchestration displays in the Verify toolbox tab, as shown in Figure 3-121. Job instance data is removed by selecting the job instance data and clicking the Delete Verify toolbox tab icon.

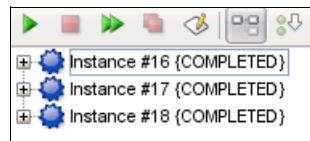


Figure 3-121 The trace data for multiple job instances is displayed

You can test multiple orchestrations in the Verify toolbox tab at the same time. To select the orchestration being tested, open and select the orchestration in the workspace. An orchestration must be open and selected in the workspace to be tested and to display the trace data for previous job instances of the orchestration.

To test multiple orchestrations, select each orchestration in turn, and click the Start Orchestration Verify toolbox tab icon for that orchestration. To start all orchestrations in a project, select any orchestration, and then click the Start all orchestrations Verify toolbox tab icon. To stop all orchestrations, click the Stop all orchestrations Verify toolbox tab icon.

The job instance trace data

When the job completes, the job instance trace shows the activities that were executed by the job. By default only the trace from the last five jobs is displayed. You can change this number in the Maximum Job Instances to be displayed field in the orchestration preference settings (select **Edit > Preferences > Orchestration**).

Figure 3-122 shows an example of this trace information. Expand and collapse the trace information using Verify toolbox tab icons.

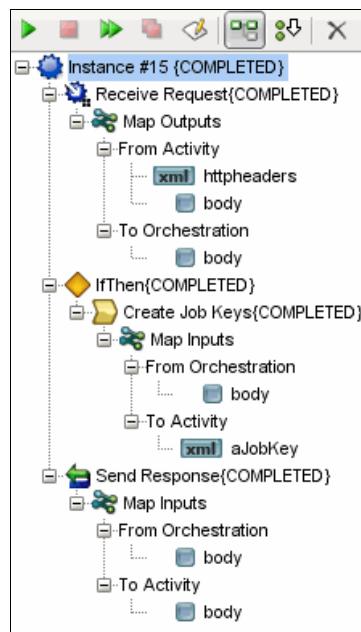


Figure 3-122 The trace data for a completed job in the Verify toolbox tab

Clicking each line of the trace data highlights the activity in the orchestration in the workspace that produced that line of trace data. Using this technique, you can discover the execution path that the job took through the orchestration. You can also click the Show Execution Path Verify toolbox tab icon, which highlights in green the entire execution path taken by the job through the orchestration, as shown in Figure 3-123.

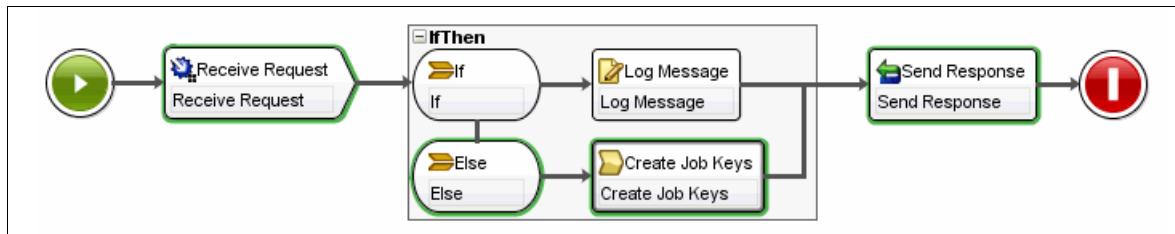


Figure 3-123 An orchestration highlighting the execution path of a job run using the Verify toolbox tab

If an orchestration fails, the trace displays a status of ERRORED and ends with the activity that failed, as shown in Figure 3-124. The trace data contains the error message for the exception.

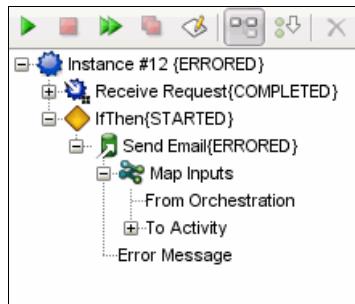


Figure 3-124 The trace for a job that failed

The trace Execution Path also highlights the activity that failed in red, as shown in Figure 3-125.

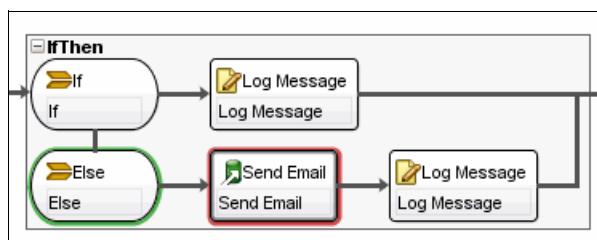


Figure 3-125 The Execution Path for a job that failed

Studio records the values of all variables and parameters used by each activity in the orchestration as the job executes. The bottom of the Verify toolbox tab contains a pane displaying message data. Clicking a line of trace data for a variable or parameter displays the value of that variable or parameter at that point of the job in the Message Data pane, as shown in Figure 3-126 on page 167.

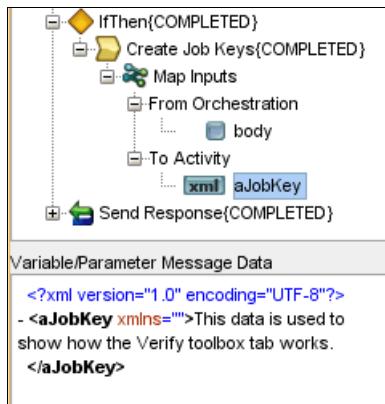


Figure 3-126 The message data for the parameter level is displayed

You can copy this message data into the Test mode of the mapping editor to test each map in the orchestration using real data from a job execution. To copy the data, click the Assign Execution's Results Verify toolbox tab icon. Note that the job execution data is not saved when Studio is closed.

The log viewer

Clicking the Show Logging Console icon in the Verify toolbox tab icon opens the Log Viewer window shown in Figure 3-127. The Log Viewer displays system log messages that are created by orchestrations that were run in Studio.

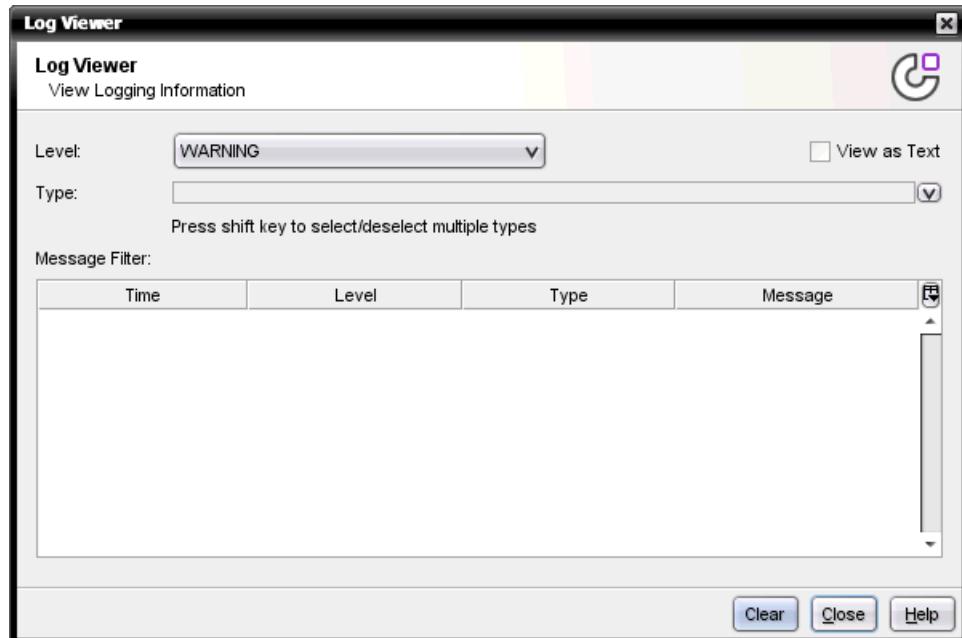


Figure 3-127 The Log Viewer window

You must set the minimum log level and log message type of the log messages before an orchestration is tested. These fields set the level and type for log messages that are recorded in the Log Viewer. If a message that is created by an orchestration does not match the log level and type set in the Log Viewer, it is not recorded.

Storage location for Studio log files: Log files for Studio are stored in the C:\Documents and Settings\User\.castiron directory. In Windows 7 systems, the path is C:\Users\User\.castiron.

Verifying an individual activity

To test an individual activity in an orchestration using the Verify toolbox tab, right-click the activity in the orchestration, and then select **Verify Activity**. The activity is tested in isolation. No other activities in the orchestration are executed.

If the activity requires input parameters, set these parameters by clicking the Map Inputs task in the checklist for the activity. You can also set a default value for the variable temporarily in the Variables toolbox tab. Click **Test** to enter the Test mode for the mapping editor and import test data in to the Input Test Data pane. Importing test data in to the Input Test Data pane is described in 3.12.3, “Test maps” on page 161.

3.12.8 HTTP Post Utility tool

Orchestrations that have an HTTP Receive Request starter activity must receive an HTTP request to run. This request can come from an ordinary web browser. However, Studio includes an HTTP Post Utility tool to help create the HTTP request. To use the tool, select **Tools → HTTP Post Utility**. The window shown in Figure 3-128 opens.

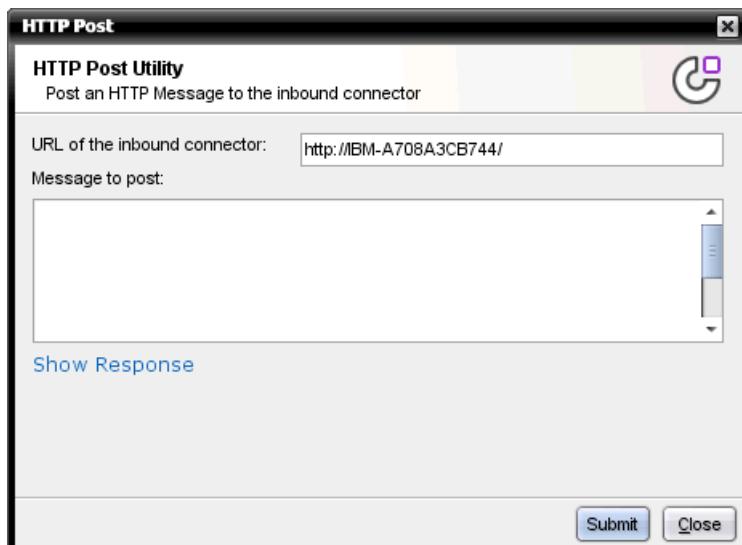


Figure 3-128 The HTTP Post Utility window

The URL of the inbound connector defaults to the host name of the local machine. Append to this URL the port defined in the HTTP endpoint and the URI for the HTTP Receive Request you are testing. Enter message data in the Message to post field if the HTTP request requires data to be posted. Click **Submit**. If an HTTP response is expected, click the **Show Response** link, which will display the HTTP response.

A simple HTML version of the HTTP Post Utility that can be used when Studio is not running is supplied with Studio when it is installed locally. This utility is typically used to test orchestrations with HTTP Receive Request starter activities after they are started in a Cast Iron run time. To open the web page, use the Windows Start menu to select **Programs → IBM → WebSphere Cast Iron Studio → HTTP Post Utility**.

3.12.9 Invoke Service tool

Orchestrations that have a Web Services Provide Service starter activity must receive a web services request to run. This request can come from an application making a web services call, for example another orchestration. However, Studio includes a web services Invoke Service tool to help create the web services request. To use the tool, select the menu **Orchestration → Invoke Service** or right-click a Provide Service activity, and select **Invoke Service**. The orchestration starts in the Verify toolbox tab and the Invoke Service window shown in Figure 3-129 opens.

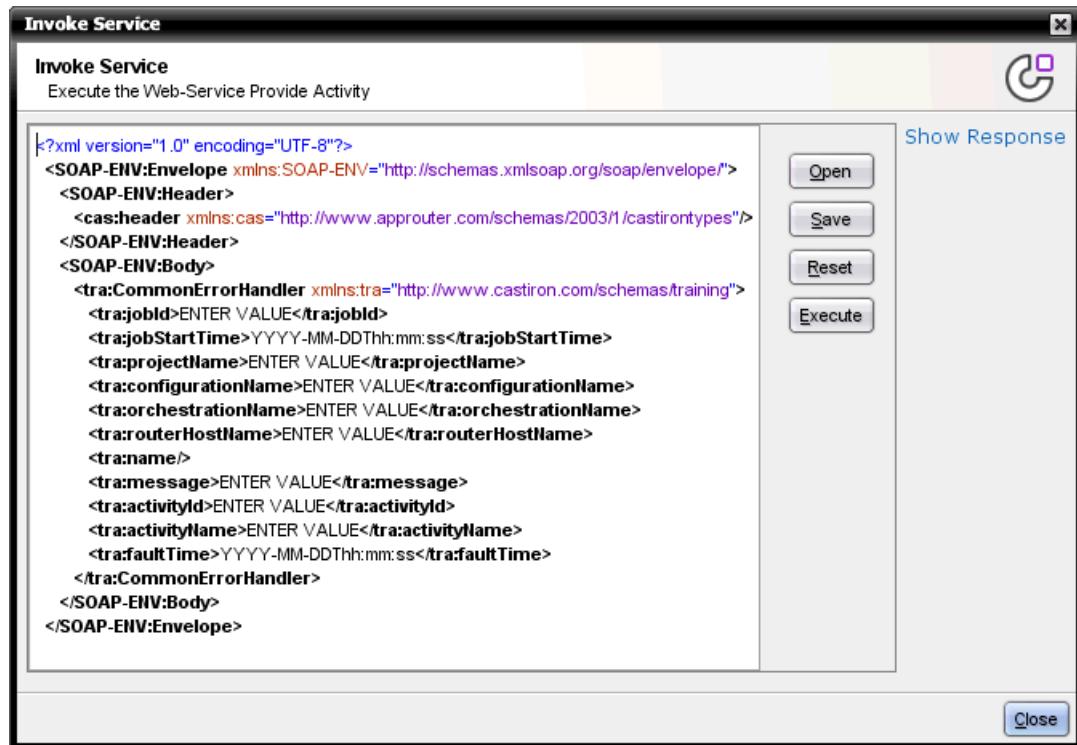


Figure 3-129 The Invoke Service window

The window is prepopulated with a request message that is generated from the WSDL for the web service that you are testing. You can edit the generated request message. To select a different test request message, click **Open**. To save the current request message, click **Save**. To invoke the web service, click **Execute**. To view the web service response, click the **Show Response** link.

The orchestration that provided the web service ran in the Verify toolbox tab. Thus, trace data for the job instance of the orchestration shows in the Verify toolbox tab. This trace data is described in 3.12.7, “Verifying orchestrations” on page 164.

To retest the web service, you must restart the orchestration in the Verify toolbox tab. To do this, you must close the Invoke Service window. You can then either start the orchestration manually in the Verify toolbox tab or just reopen the Invoke Service window as before.

If you close the Invoke Service window before executing the test, the Invoke Service option might not be available because the orchestration was not stopped in the Verify toolbox tab. To reopen the Invoke Service window, first click the Stop Orchestration icon on the Verify toolbox tab, and then open the window as before.

3.13 Exporting and publishing

After you develop and test a project, you can publish it to a Cast Iron runtime. You can use one of the following methods to publish a project:

- ▶ Export the project from Studio, and upload the project using the WMC for the Cast Iron runtime where it is to run
- ▶ Publish the project directly from Studio to a Cast Iron runtime

A user with Administrator or Publisher privileges is needed to publish directly from Studio to a Cast Iron runtime.

For security reasons, access to production environments might be restricted. In this case, you can still develop new orchestrations in Studio, but you might not be able to directly publish projects to production environments. You can then export the project and upload using the WMC.

All the orchestrations in a project are exported or published simultaneously. Thus, all the orchestrations in a project must be valid before the project can be exported or published. Source code for all orchestrations is also published at the same time. This source code can be later downloaded from the Cast Iron runtime using the WMC as described in 4.2.13, “Uploading and downloading a project” on page 197.

3.13.1 Exporting a project

To export a project from Studio:

1. Select **File** → **Export Project**.
2. In the dialog box that opens, click **Yes**.
3. Set a name and directory, and click **Save**.

The file saves with an extension of .par. Use the WMC to upload the file to a Cast Iron runtime, as described in 4.2.13, “Uploading and downloading a project” on page 197.

Studio does not provide an option to import projects. Studio can open only projects that have the correct project directory structure and project file, as described in 3.3.3, “Project directory structure” on page 91. However, you can download an archive file that contains the project source code from the WMC by selecting the Configuration Details pane for the project configuration that is associated with the project and then clicking the **Download** link, as shown in Figure 3-130. You can then extract the contents of this archive file and create the correct directory structure for the project.

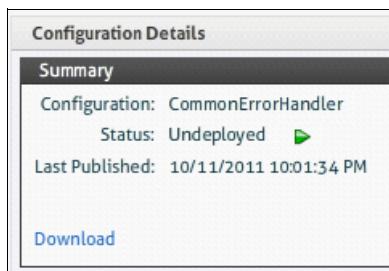


Figure 3-130 Downloading the archive file for a project from the WMC

Tip: The .par file is also a .zip file and contains the source code. To extract the source code, rename the .par file to an extension of .zip, and then open the .zip file. The source directory in the .zip file contains the source code in the same format that it is downloaded from the WMC.

3.13.2 Publishing a project directly from Studio

To publish a project directly from Studio to a Cast Iron Integration Appliance:

1. Select **File → Publish Project**.
2. When prompted to save the project, click **Yes**.
3. In the Publish Project dialog box, shown in Figure 3-131, enter the host name of the management IP address for the Cast Iron Integration Appliance where you are publishing.
4. Enter a user name and password for that Integration Appliance. This user must have Administrator or Publisher privileges.

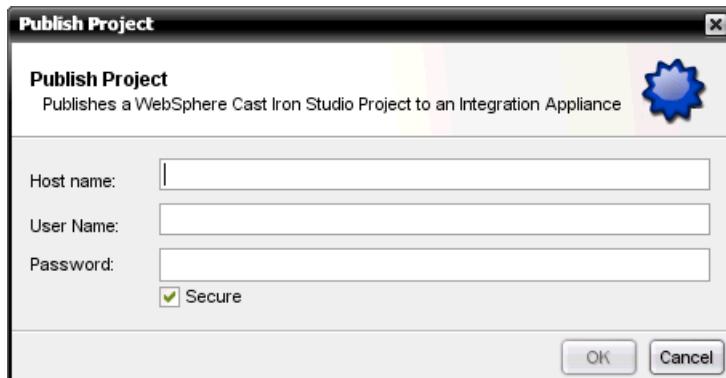


Figure 3-131 Publishing a project

5. The connection to the Cast Iron Integration Appliance uses SSL by default. If this is turned off by the administrator of this Integration Appliance, clear the Secure option.
6. Click **OK** to publish.

Tip: The values on the Publish Project window are remembered until Studio is closed. To publish any project from Studio that already published one project and was not closed since it was published, just click **OK**.

Publishing to Cast Iron Live is performed using the WMC. The project is first saved to Cast Iron Live. In Studio, click **File → Save Project [To Cloud]**, as shown in Figure 3-132 on page 172. Notice that the File menu for the Cast Iron Live Studio does not have an option to Publish Project.

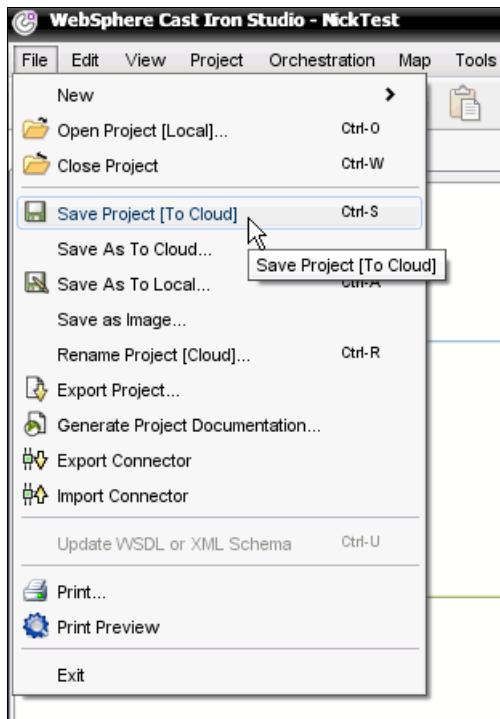


Figure 3-132 Saving a project to Cast Iron Live

After you save the project to Cast Iron Live, select the Modify tab in the WMC. Click the **Publish Project** icon shown in Figure 3-133 for the project that you want to publish, and select the environment where the project is being published to.

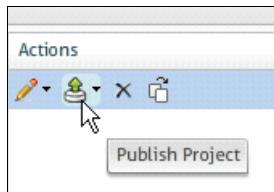


Figure 3-133 Publishing a project in Cast Iron Live

A project fails to publish if a project of the same name and version is currently started or deployed on the Cast Iron runtime. To fix this issue, use the WMC to stop and undeploy the project. The new project replaces the existing one, so if the existing version needs to be saved, it must be downloaded from the WMC first. Notice that the project on the Integration Appliance does not have to be deleted. Deleting a project removes its Job Log data. Thus, do not delete a project just to republish the project.



Management and monitoring

This chapter provides an overview of the monitoring and management capabilities that are available in IBM WebSphere Cast Iron. It discusses how to manage the appliance and orchestrations and also covers troubleshooting and log management.

This chapter includes the following topics:

- ▶ Management
- ▶ Project configuration life cycle
- ▶ Monitoring and troubleshooting
- ▶ Network
- ▶ The staging database
- ▶ Resource utilization
- ▶ Commands
- ▶ Security
- ▶ Appliance and connector upgrades

4.1 Management

The management of the WebSphere Cast Iron runtime involves administration tasks, such as orchestration management, appliance upgrades, and problem identification. You can manage the WebSphere Cast Iron runtime using the three different interfaces shown in Figure 4-1:

- ▶ Management API (web services)
- ▶ Command Line Interface (CLI)
- ▶ Web management console (WMC)

Although some tasks are available in all three options, each interface generally has a distinct applicability.

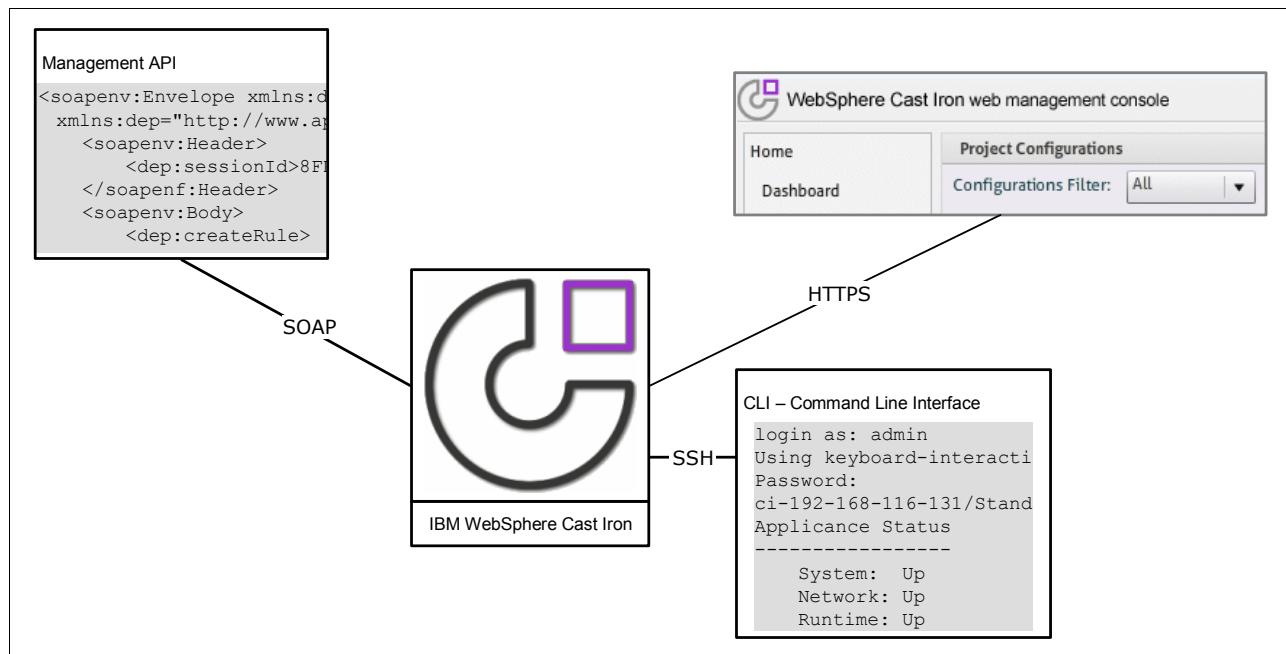


Figure 4-1 IBM WebSphere Cast Iron administration components

4.1.1 Command Line Interface

The Command Line Interface (CLI) is a powerful administration tool used to manage and maintain the IBM WebSphere Cast Iron Integration Appliance. Some administration functions, such as the initial setup for the network, can be executed only through the CLI.

The CLI is accessible using one of the following methods:

- ▶ SSH session
- ▶ Telnet session
- ▶ Console session

Example 4-1 uses a CLI session, where a system command with the show status function is issued.

Example 4-1 Executing a system show status command in a console session

```
ci/Standalone> system show status
Appliance Status
-----
```

System: Up
Network: Up
Runtime: Up

Command complete

Note: The CLI is not available for IBM WebSphere Cast Iron Live.

For more details about the CLI, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.cli.doc/CLI_about_CLI.html

4.1.2 Web Management Console

The Web Management Console (WMC) is a web-based tool that is used to administer the Cast Iron runtime and to manage projects and orchestrations. You can complete tasks, such as Integration Appliance health and log monitoring, network management, and orchestration troubleshooting, using this interface. The WMC is available after the Cast Iron runtime starts successfully. Figure 4-2 shows the WMC.

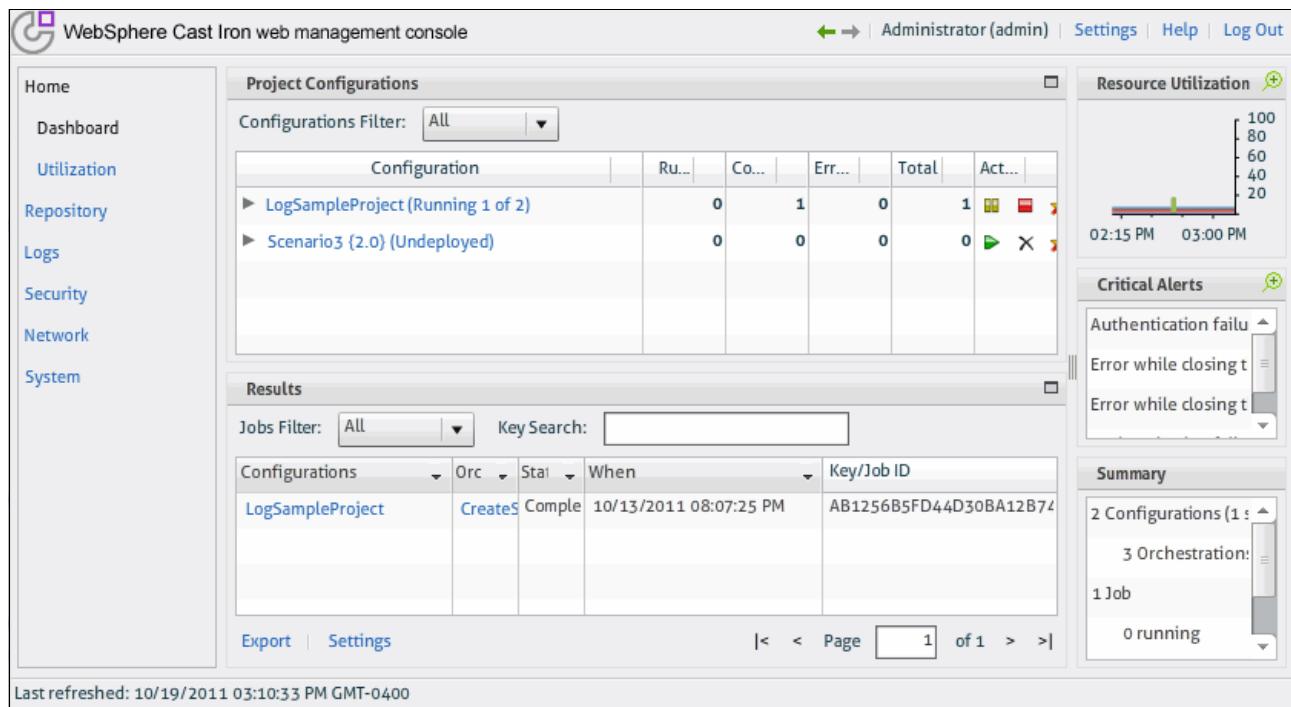


Figure 4-2 Web Management Console

In Cast Iron Live, multiple runtime environments are provided, as highlighted in Figure 4-3 on page 176. These environments are available on tabs within the WMC. Each environment provides the same functions of the WMC as described in this chapter. Differences in using Cast Iron Live are highlighted.

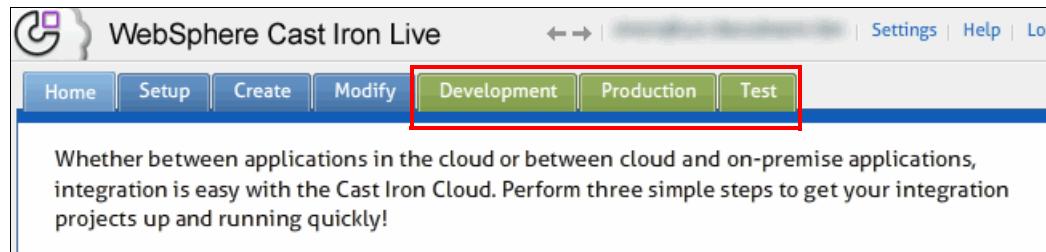


Figure 4-3 Cast Iron Live environment tabs

Important: Although this chapter focuses on the WMC, several functions described in this chapter are also available in the CLI and the Management API. Refer to the online documentation for more information.

Logging in

You log in to the WMC through a web browser using the following URL, where <emgmt> is the IP address of the management interface:

`https://<emgmt>`

You provide a user name and password, as shown in Figure 4-4.



Figure 4-4 WMC login page

Cast Iron Live: The WMC in Cast Iron Live has a different URL that is provided with your Cast Iron Live subscription.

The initial WMC page looks like Figure 4-5. Most of the functions discussed in this chapter are available through the left-side menu, as shown in Figure 4-5.

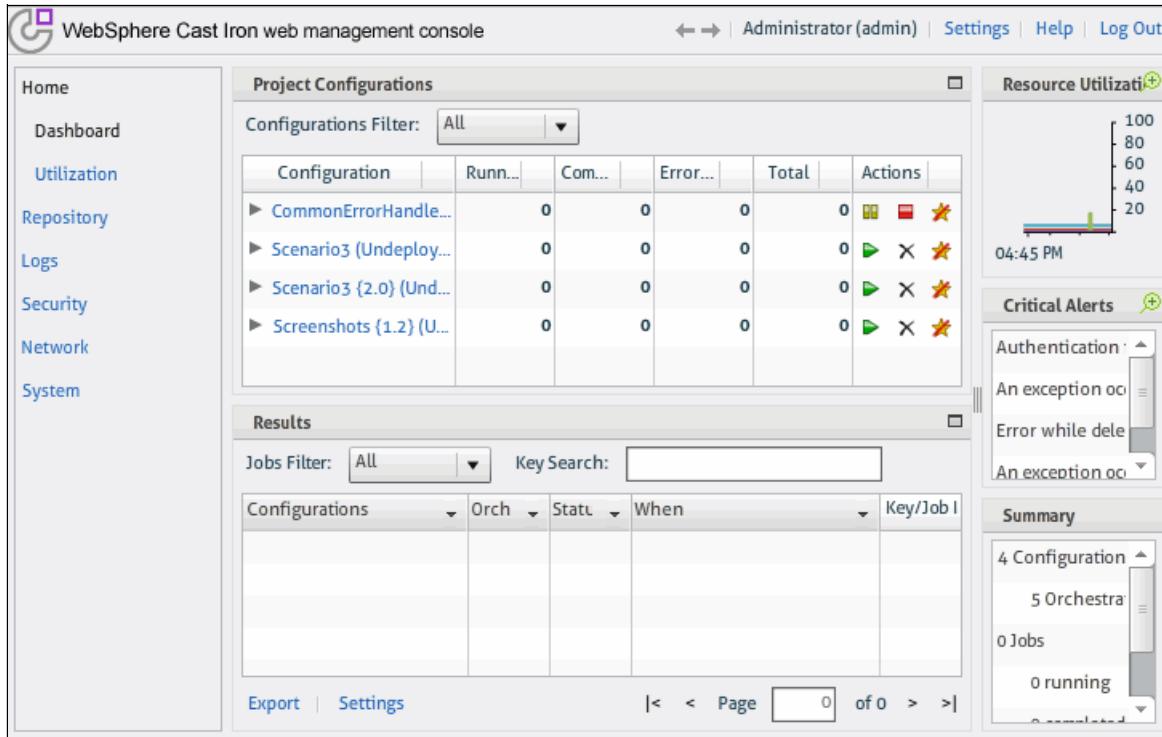


Figure 4-5 Web Management Console menu

Hint: A Help link is always available when using the WMC.

Web management console settings

Users can modify their own user profile settings in the WMC. The Settings link is available in the console on the top-right corner, as indicated in Figure 4-6.

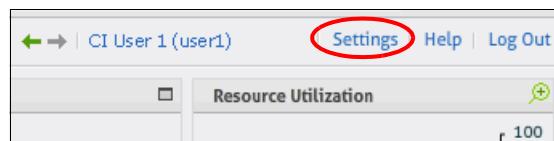


Figure 4-6 User profile Settings link

The settings that you can change include user profile, session, and display settings.

The User Profile section allows you to change personal fields, such as the password, the display name, and email. With WebSphere Cloud Live especially, the email field is important because it is the primary method of communication between the system administrator and the user.

Users can also modify the session time-out value and dashboard refresh interval through the Settings page.

Session time out: When using Cast Iron Live, be aware of your session timing out. If your session times out when you are editing a project in Studio, you cannot save your project to Cast Iron live. You can save to your local disk. See the Tip on page 81 for more information.

You can also change settings that are related to display, such as page size for logs and other display settings.

Figure 4-7 shows all of the fields that are available in the Settings page.

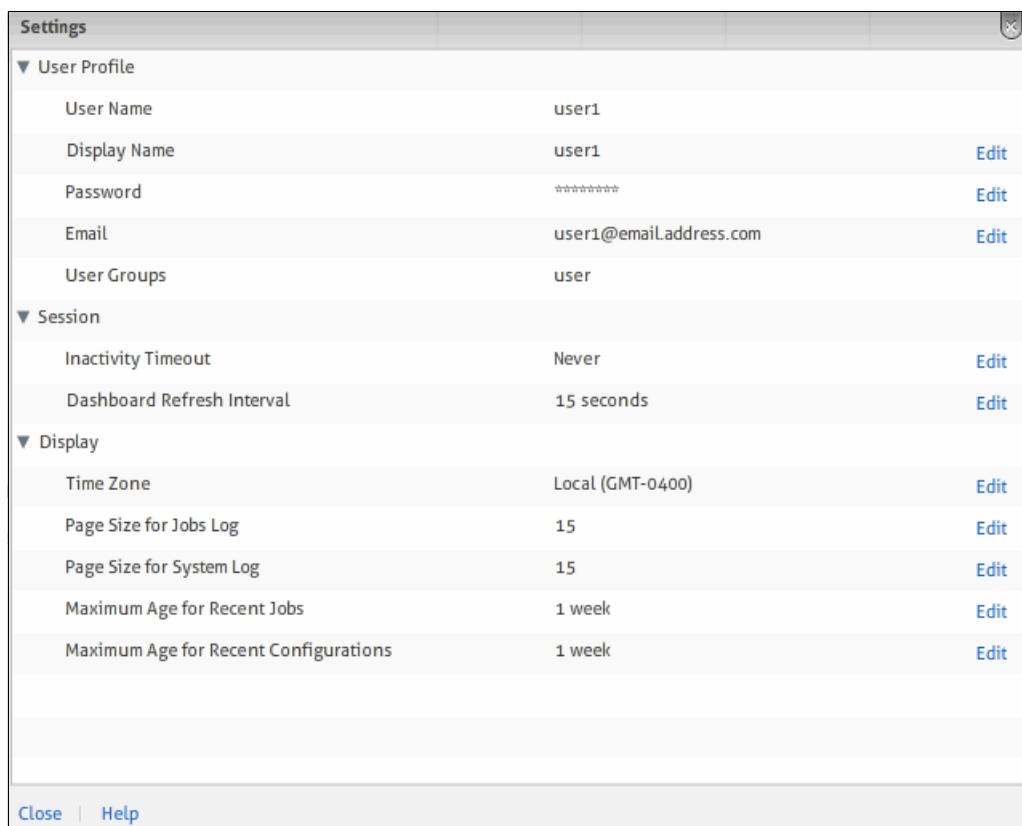


Figure 4-7 Fields from the Settings page

The Settings page for WebSphere Cast Iron Live is slightly different because it includes complementary fields. For example, the Libraries Environment field allows you to select from which environment Studio loads the connector libraries, as indicated in Figure 4-8 on page 179.

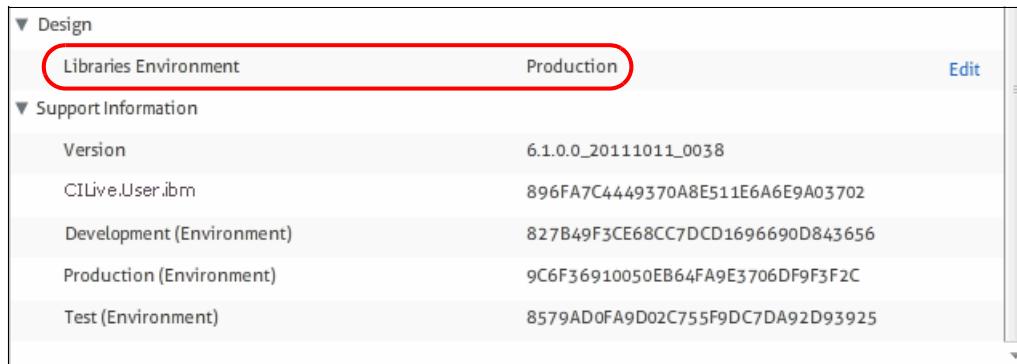


Figure 4-8 Libraries Environment field for the Cast Iron Live Settings page

For more information about the Settings page, visit:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.appliance.doc/About_the_WMC/consoleSettings.htm

4.1.3 Management API

The Management API is a web service layer that allows administration of the Cast Iron runtime using web service calls to perform tasks, such as configuration management, security configuration, and other areas.

You might use the Management API for the following reasons:

- ▶ To build scripts for automation
- ▶ To create your own client application for management

The following WSDL files are available in both the Integration Appliance and in Cast Iron Live:

- ▶ security.wsdl
- ▶ staging.wsdl
- ▶ deployment.wsdl
- ▶ system.wsdl
- ▶ orcmn.wsdl
- ▶ lognotif.wsdl

Example 4-2 shows how to get a list of all users based on a session ID.

Example 4-2 The getUsers soap request

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <ns1:sec="http://www.approuter.com/schemas/2008/1/security">
    <soapenv:Header>
      <sec:sessionId>8EB8F17160F7D1E561EC45EFEBF344CF</sec:sessionId>
    </soapenv:Header>
    <soapenv:Body>
      <sec:getUsers/>
    </soapenv:Body>
  </soapenv:Envelope>

```

When using the Management API with Cast Iron Live versus an integration appliance, you must use the setCurrentScope operation to set the scope to the environment where you want to run SOAP operations before you send any other requests.

More information is available at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.api.doc/ci00001.html

For more information about the Management API, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.api.doc/ci00000.html

4.2 Project configuration life cycle

A key area for WebSphere Cast Iron is orchestration management. After a project is published to the Cast Iron runtime, the project and its orchestrations must be managed. For example, after a project is published, it does not start by default. User intervention is required to start the project.

To illustrate the management of a project, this example uses a project called `ITSO_Management_Chapter` with two orchestrations:

- ▶ `CSVToSalesforce`
- ▶ `CurrentTime`

This project was published to the Cast Iron runtime. The **Repository → Configurations** view, Figure 4-9, shows the published project in the WMC.

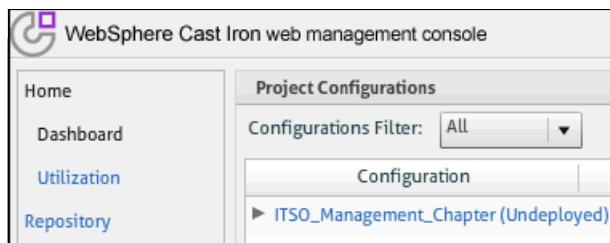


Figure 4-9 Dashboard page showing a configuration

4.2.1 Project configurations

What is known as a *Project* in Studio, is known as a *Project Configuration* in the runtime. The WMC (and management API) uses this terminology because one single project can be published many times inside the WebSphere Cast Iron runtime using other versions and configuration properties. For example an enterprise can run a project configuration with a set of properties to handle an integration for their headquarters. The same project can also be used for a branch office with several properties for the endpoints. A similar example is using one configuration to connect to test target endpoints while another configuration is connects to development endpoints.

When you publish a project from Studio, the WMC displays the project with default project settings, which creates a default project configuration. You can clone any project configuration to create new project configurations. For more information about changing configuration properties, refer to 4.2.6, “Changing configuration properties” on page 185. You can find more information in 4.2.8, “Cloning configurations” on page 188.

Every project configuration has at least one orchestration. To view the orchestration or orchestrations that are part of the project configuration, click the arrow on the side of the

configuration name to expand it. Figure 4-10 shows the two orchestrations that are part of the ITSO_Management_Chapter project configuration.

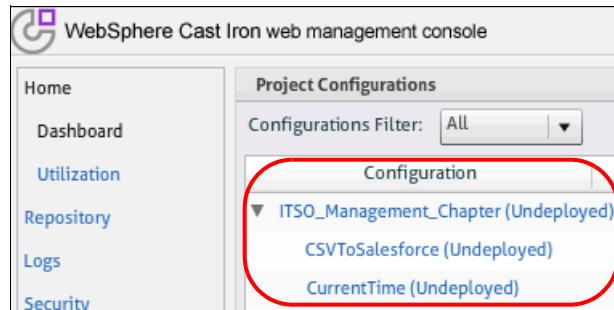


Figure 4-10 Project configuration expanded showing the available orchestrations

4.2.2 Starting a configuration

When a project is published through Studio or imported into the WMC, the project is in undeployed status. Deployment is the process of preparing and instantiating the run time components that are necessary for the integration. Among other tasks, the deployment process is responsible for providing configuration properties values to each orchestration.

In the WMC, you can deploy and start a configuration with the Run Configuration action (represented as the arrow as indicated in Figure 4-11), available from the following screens in the WMC:

- ▶ Through the Project Configuration page (**Home** → **Dashboard**)
- ▶ Using **Repository** → **Configurations**

Click the Run Configuration icon, shown in Figure 4-11, for a project to deploy and start the configuration.

Configuration	Running	Completed	Errored	Total	Actions
ITSO_Management_Chapter (Undeployed)	0	0	0	0	
CSVToSalesforce (Undeployed)	0	0	0	0	
CurrentTime (Undeployed)	0	0	0	0	

Figure 4-11 Run Configuration icon

The configuration status transitions to *Deploying*, as shown in Figure 4-12. While the configuration is deploying, you cannot perform any actions, and the Actions icons are unavailable.

Configuration	Running	Completed	Errored	Total	Actions
ITSO_Management_Chapter (Deploying)	0	0	0	0	
CSVToSalesforce (Deploying)	0	0	0	0	
CurrentTime (Deploying)	0	0	0	0	

Figure 4-12 A configuration being deployed

Important: You can deploy a configuration only if there is at least one enabled orchestration. You can learn more about disabling orchestrations in 4.2.7, “Disabling orchestrations” on page 186.

If no errors occur, the configuration is ready to run the enabled orchestrations. The WMC shows the status as Running, and additional life cycle actions are available for the configuration, as indicated in Figure 4-13.

Project Configurations						
Configuration	Running	Completed	Errored	Total	Actions	
▼ ITSO_Management_Chapter (Running)	0	0	0	0	0	   
CSVToSalesforce (Running)	0	0	0	0	0	
CurrentTime (Running)	0	0	0	0	0	

Figure 4-13 A Configuration at Running state

When a project configuration is running, all the enabled orchestrations that belong to that project also show the same Running status.

Enabling and disabling orchestrations: You can enable or disable individual orchestrations, as described in 4.2.7, “Disabling orchestrations” on page 186.

If for some reason one of the orchestrations cannot start, the entire configuration goes to the stopped state.

Avoid trouble: For critical integrations, avoid a situation where one failed orchestration stops other orchestrations in the same project configuration by having only one orchestration per configuration.

For information about troubleshooting, refer to 4.3, “Monitoring and troubleshooting” on page 198.

Cast Iron Live: When using Cast Iron Live, an orchestration will not start if it requires a secure connector that is not currently running.

The dashboard includes the following execution counters that provide information about each orchestration, as shown in Figure 4-14 on page 183:

- ▶ Jobs running
- ▶ Successfully completed job instances
- ▶ Jobs that have errored

Running	Completed	Errored	Total
0	19	7	26
0	12	4	16
0	7	3	10

Figure 4-14 Orchestration counters

4.2.3 Pausing a configuration

You might have situations where you must suspend a configuration temporarily. For example, you might want to suspend a configuration if you know that an endpoint that is used by one of the orchestrations is out of service or has ongoing maintenance. You can suspend the configuration to avoid new jobs being started and excessive and unnecessary log generation.

To suspend a configuration temporarily, use the Pause icon in the Actions column, as shown in Figure 4-15. This action pauses any jobs that are currently running and prevents new job instances from starting. When the configuration resumes, the suspended job continues from the point where you paused it.

Project Configurations						
Configurations Filter:	All	Running	Completed	Errored	Total	Actions
▼ ITSO_Management_Chapter (Running)	0	0	0	0	0	
CSVToSalesforce (Running)	0	0	0	0	0	
CurrentTime (Running)	0	0	0	0	0	

Figure 4-15 Pause configuration button

Scheduling downtime: You can schedule down time for configurations, which changes the status to a suspended state. For more information, refer to 4.2.12, “Scheduling downtime” on page 194.

4.2.4 Stopping a configuration

Use the Stop icon for maintenance purposes. You might stop a configuration in the following circumstances:

- ▶ Prevent the run time from starting new orchestration jobs (instances)
- ▶ Make changes in the project configuration (orchestration settings)
- ▶ Make changes in any related system to which an orchestration connects

This action stops the run time from processing new orchestration jobs for the project configuration. It puts the configuration out of service.

When stopping a service, a Stop Configuration window, shown in Figure 4-16 on page 184, allows you to decide how to handle jobs that are currently running:

- ▶ *Allow jobs to finish* prevents any new orchestration jobs from starting and allows current jobs to finish processing. This action allows the job instances to run to completion before the project configuration stops.

- ▶ *Cancel running jobs* prevents any new orchestration jobs from starting and stops all currently running jobs. This action stops all jobs immediately.

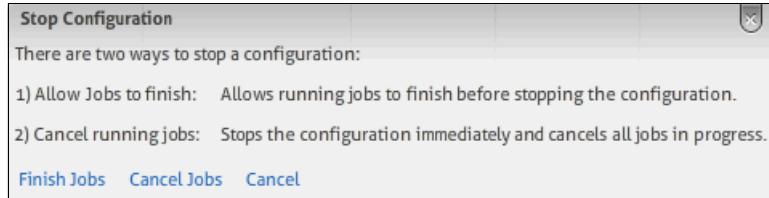


Figure 4-16 Stop Configuration window

You cannot stop orchestrations individually. When a project configuration is stopped, all orchestrations that belong to it are stopped also.

Other methods to stop a configuration: The Stop action is also available in the Management API as the stopAndCancel and stopAndWait operations. The Schedule Downtime function also allows you to stop configurations.

Important: A stopped configuration is started by the Schedule Downtime process (see 4.2.12, “Scheduling downtime” on page 194) even if it was stopped manually.

4.2.5 Undeploying a configuration

When a project must be republished or when a configuration (or orchestration) parameter modification is required, you must undeploy the project configuration. Modifications are only valid for new job instances after the project configuration is redeployed and started.

To undeploy a project configuration, stop the configuration first, and then undeploy it using the Undeploy action, as highlighted in Figure 4-17.

Project Configurations						
Configuration	Running	Completed	Errored	Total	Actions	
ITSO_Management_Chapter (Stopped)	0	19	7	26		
CSVToSalesforce (Stopped)	0	12	4	16		
CurrentTime (Stopped)	0	7	3	10		

Figure 4-17 Undeploying a configuration

Notice in Figure 4-18 that the configuration state changed from *Stopped* to *Undeployed*.

Configuration
ITSO_Management_Chapter (Undeployed)
CSVToSalesforce (Undeployed)
CurrentTime (Undeployed)

Figure 4-18 Configuration undeployed

4.2.6 Changing configuration properties

A typical use for configuration properties is to define a property for endpoint settings, allowing them to be changed at runtime. This section explains how to apply configuration property changes for a project configuration using the WMC. To do this, the project configuration must be in an undeployed state. The example in this section changes properties for an endpoint in the CSVToSalesforce orchestration.

Only users with administrator or publisher permissions can promote project configuration changes.

Using the WMC, click **Repository** → **Configurations**, and then click the configuration to open the Configuration Details window, as shown in Figure 4-19. The Summary view at the top shows the number of orchestrations and properties. Notice that the status is Undeployed.

Configuration Details	
Summary	
Configuration: ITSO_Management_Chapter	# Orchestrations: 2
Status: Undeployed ➤	# Properties: 8
Last Published: 09/29/2011 06:51:52 PM	# Assets: 0
	# Downtimes: 0
Download	

Figure 4-19 Configuration Details page in the WMC

The Properties view, Figure 4-20, shows the properties from all orchestrations that are part of the selected project configuration.

Properties (8)	
Name	Value
ftp_file	accounts.csv
ftp_host	192.168.116.1
ftp_password	*****
ftp_path	/
ftp_username	itso
http_path	go
salesforce_password	*****
salesforce_username	castiron@itso.ibm.com

Figure 4-20 Properties view

The example in this section changes the *ftp_host* and *salesforce_username* properties to point to a production FTP host and to use a different user name for salesforce credentials.

To modify the properties for this configuration, click **Edit**, which is located on the bottom of the properties view. In the Edit Configuration Properties window, change the property value for each field. Figure 4-21 shows the ftp_host and salesforce_username new property values.

Edit Configuration Properties		
Name	Default Value	Value
ftp_file	accounts.csv	accounts.csv
ftp_host	192.168.116.1	9.44.157.164
ftp_password	*****	*****
ftp_path	/	/
ftp_username	itso	testuser@castiron.com
http_path	go	go
http_path2	getTime	getTime
salesforce_password	*****	*
salesforce_username	devsalesforce@castiron.com	testuser@castiron.com

Figure 4-21 Changing properties configuration

When you finish the modifications, click the **Save** link on the bottom of the page to make the configuration persistent and available.

With these changes, an orchestration that references these configuration properties will use the new values after the project is deployed and started.

4.2.7 Disabling orchestrations

By default, when a project configuration is published, all the orchestrations are enabled. You might need to disable an orchestration in the following situations:

- ▶ A project configuration has one or more orchestrations that are partially developed and not fully functional.
- ▶ An endpoint that an orchestration connects to is down. It is wise to disable this orchestration to stop it from starting new job instances.
- ▶ Another orchestration or an external system is temporarily receiving requests that are usually sent to this orchestration.
- ▶ The orchestration is simply going to be out of service.

Enabling or disabling orchestrations requires the configuration to be stopped and undeployed.

Disabling an orchestration is a simple task in the WMC. When you open the Configuration Details view for a project, the orchestrations are listed in the Orchestrations section along with the current status for each, as shown in Figure 4-22.

Orchestrations (2)	
Name	Status
CSVToSalesforce	Enabled
CurrentTime	Enabled
Edit	

Figure 4-22 Orchestrations list and current status

Click **Edit** at the bottom of the list to open the Edit Orchestration Settings window. The enabled orchestrations are checked in the Enabled column, as shown in Figure 4-23. Clear the setting for the orchestration or orchestrations that you want to disable.

Edit Orchestration Settings	
Name	Enabled
	<input type="checkbox"/>
CSVToSalesforce	<input checked="" type="checkbox"/>
CurrentTime	<input checked="" type="checkbox"/>

Figure 4-23 Enable or disable individual orchestration

To enable or disable all orchestration in a single step, check or clear the unnamed header, as indicated in Figure 4-24.

Edit Orchestration Settings	
Name	Enabled
	<input checked="" type="checkbox"/>
CSVToSalesforce	<input checked="" type="checkbox"/>
CurrentTime	<input checked="" type="checkbox"/>

Figure 4-24 Enabling or disabling all orchestrations

If you disable all orchestrations, the project configuration will not start, and a warning message is issued, as shown in Figure 4-25.

Orchestrations (2)	
Name	Status
CSVToSalesforce	Disabled
CurrentTime	Disabled
Operation Failed	
At least one orchestration must be enabled.	
OK	

Figure 4-25 Project configuration fails to start

Save the orchestration settings to return to the Configuration Details view, where you can see that the CurrentTime orchestration is disabled, as shown in Figure 4-26.

Orchestrations (2)	
Name	Status
CSVToSalesforce	Enabled
CurrentTime	Disabled
Edit	

Figure 4-26 CurrentTime orchestration is disabled

When you start the project configuration and monitor the status through the dashboard, it is easy to see that the CurrentTime orchestration is disabled, as shown in Figure 4-27 on page 188.

Figure 4-27 CurrentTime orchestration remains disabled

4.2.8 Cloning configurations

The WMC allows you to clone a project configuration without having to publish it again. This process can be extremely useful when you need to have the same project configuration running multiple times simultaneously but with different configuration properties. This process shows how to duplicate a project by cloning it and change the properties to different values. This task allows you to have both project configurations (the original and the clone) running simultaneously, for example, consider a case where a company has a set of orchestrations that are configured for its headquarter's endpoints settings. Now, the company must have the same orchestrations also running for a branch office, but using another group of endpoint settings.

The Clone action is available for each project configuration in the WMC, as highlighted in Figure 4-28.

Figure 4-28 Clone action

In this example, the ITSO_Management_Chapter configuration is the configuration to be cloned. To clone this project configuration, click the Clone Configuration icon, and then enter a name for the new cloned project in the Clone Project Configuration window, as shown in Figure 4-29.

Figure 4-29 Naming a clone configuration

The name for the cloned project must follow the same name conventions you use for all projects. The new name is the original name with the new name after it in square brackets, as shown in Figure 4-30 on page 189. The clone is in the undeployed state, so you can change any property values that you want before starting it.

Project Configurations	
Configurations Filter: All	
Configuration	
▼	ITSO_Management_Chapter (Running)
	CSVToSalesforce (Running)
	CurrentTime (Running)
▼	ITSO_Management_Chapter [ITSO_Management_Branch] (Undeployed)
	CSVToSalesforce (Undeployed)
	CurrentTime (Undeployed)

Figure 4-30 Cloned configuration project

Tip: Make sure that the cloned project configuration property values that you provide do not conflict with any other configuration that is already deployed, including the configuration from which it was cloned. It is common to have conflicts that are related to the HTTP port and URL configuration settings. If there are conflicts in settings, the project configuration will not start.

4.2.9 Deleting configurations

To delete a configuration:

1. Put the project configuration in an Undeployed state.
2. In the Project Configurations list, click the **Delete** icon, as indicated in Figure 4-31.

▼ ITSO_Management_Chapter [ITSO_Management_Branch] (Undeployed)	0	0	0	0	▶	X	★	⋮
CSVToSalesforce (Undeployed)	0	0	0	0				
CurrentTime (Undeployed)	0	0	0	0				

Figure 4-31 Delete action

3. Click **Yes** in the confirmation window to delete the configuration.

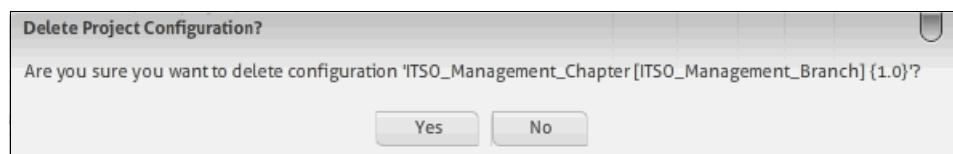


Figure 4-32 Deletion confirmation

Important: When a Project Configuration is deleted, all monitoring log history that is associated with it is also deleted.

4.2.10 Versioning

WebSphere Cast Iron runtime provides version control functionality that allows you to have several Project Configuration versions in the runtime in parallel.

Important: If a configuration project with the same name and version is published, it overwrites the configuration project that is deployed in the run time.

Figure 4-33 shows a project configuration published with a different version, in this case version 2.0. Notice that version 1.0 does not display, which is the expected behavior.

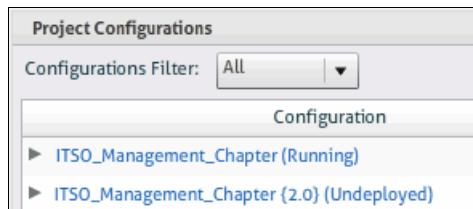


Figure 4-33 Another project configuration version

Hint: A version number is required for some Management API operations, even if the project configuration that you are targeting is version 1.0.

When uploading a project, you can change the version number to another number other than the version originally exported from the Studio project.

For more information, see 4.2.13, “Uploading and downloading a project” on page 197 and 3.3.4, “Versioning projects” on page 92.

4.2.11 Assets

This section explains how to manage database and web services assets from the Web Management Console.

The following assets are managed from the WMC:

- ▶ Database scripts
- ▶ Web Services Description Language (WSDL) files

Database scripts

For those orchestrations that contain database activities that specify the Exactly Once delivery option, database scripts are created automatically when you publish a project. A typical scenario is when a starter activity monitors a database table for updated rows (Get Updated Rows activity). In this example, a trigger and a buffer table are created by WebSphere Cast Iron. The trigger monitors the source table and copies updated data to the buffer table. The orchestration starts with the activity reading data from the buffer table. Scripts to create the trigger and database table are created for you. You also get database scripts if you are using the Get Inserted Rows, Get Updated Rows, or Get Deleted Rows activities, but not the Poll Table database starter activity.

You can manage the assets from the WMC by opening the project configuration’s Configuration Details page. The database endpoints with assets are listed in the Assets section. Figure 4-34 on page 191 lists two entries:

- ▶ ProductsDatabase
- ▶ BankWS



Figure 4-34 Assets pane in the Configuration Details

Clicking the ProductsDatabase database endpoint entry opens a window showing the list of assets you can generate scripts for. The ProductsDatabase in this example contains an update trigger and a buffer table, as shown in Figure 4-35.

Type	Name	Default Value	Exists	Valid
Buffer Table	T SYSCAT.CI_BT_ROUTINEDEP	SYSCAT.ROUTINEDEP	!	!
Update Trigger	T CI_UT_ROUTINEDEP	SYSCAT.ROUTINEDEP		

Figure 4-35 ProductsDatabase assets that can have scripts generated

When you select one of the database script links at the bottom of the asset list, the script is generated if you click **Create**, **Drop**, or **Recreate**, as shown in Figure 4-36. You can execute the script through the WMC, or you can download it. If you download the script, you can change it and run it outside the WMC.

Figure 4-36 Creating a trigger

These operations can run only when the project configuration is undeployed.

Note: The proposed names of the database objects can be modified in WMC before executing the script. The default names begin with CI_. The next two characters indicate the type of object:

- ▶ buffer table (BT)
 - ▶ control table (CT)
 - ▶ Oracle sequence (SEQ)
 - ▶ insert trigger (IT)
 - ▶ update trigger (UT)
 - ▶ delete trigger (DT).

For more information about working with these assets, refer to 3.6.7, “Database activities, assets, and the staging database” on page 114.

WSDL files

An orchestration exposed as a web service through the Provide Service activity has a WSDL file that describes the service. Clients of this service need a copy of this file. The WSDL file is made available automatically in the WMC when the project is published. You can download it to use it to build the web service client.

The asset is available in the project configuration’s Configuration Details page in the Assets section, as shown in Figure 4-37. When you select the asset that is related to a Provider Service activity, in this case BankWS, you have access to the WSDL file that is related to it.



Figure 4-37 Accessing the WSDL file

You can view the WSDL file or download it, as shown in Figure 4-38 on page 194.

Web Services Assets	
Activity	Orchestration
Get Accounts	project-url://Orchestrations/Orchestration
Get Accounts	
<pre><?xml version="1.0" encoding="UTF-8"?> <wsdl:definitions name="Get_Accounts" targetNamespace="http://www.approuter.com/connectors/wsdl" xmlns:tns2="http://www.approuter.com/connectors/wsdl" xmlns:tns="http://www.approuter.com/connectors/wsdl" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"> <wsdl:types> <xsd:schema targetNamespace="http://www.approuter.com/connectors/wsdl" xmlns:tns="http://bank.rad75.itso/"> <xsd:import namespace="http://bank.rad75.itso/" /> </xsd:schema> <xss:schema targetNamespace="http://bank.rad75.itso/" version="1.0" xmlns:tns="http://bank.rad75.itso/"> <xss:element name="AccountDoesNotExistException" type="tns:AccountDoesNotExistException"/> <xss:element name="CustomerDoesNotExistException" type="tns:CustomerDoesNotExistException"/> <xss:element name="getAccountBalance" type="tns:getAccountBalance"/> <xss:element name="getAccountBalanceResponse" type="tns:getAccountBalanceResponse"/> <xss:element name="getAccountId" type="tns:getAccountId"/> <xss:element name="getAccountIdResponse" type="tns:getAccountIdResponse"/> <xss:element name="getCustomerFullName" type="tns:getCustomerFullName"/> <xss:element name="getCustomerFullNameResponse" type="tns:getCustomerFullNameResponse"/> <xss:element name="getNumAccounts" type="tns:getNumAccounts"/> <xss:element name="getNumAccountsResponse" type="tns:getNumAccountsResponse"/></pre>	
Download Close Help	

Figure 4-38 WSDL content and download link

For more information, see:

- ▶ Chapter 9, “Common error handlers” on page 349. The common error handler is called as a web service.
- ▶ http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.appliance.doc/Managing_Projects/toc_managing_projects.htm

4.2.12 Scheduling downtime

Maintenance of the Cast Iron runtime or a related or connected system is sometimes required, for example, a company has an orchestration that integrates with an ERP system. Each day the ERP system is brought down for maintenance. To stop this orchestration from receiving new requests and to avoid starting new jobs that can produce errors, the company can create scheduled downtime rules that stop or suspend the project configuration at a specified date and time.

To create or edit downtime rules, you do not have to undeploy the project configuration. Only users with administrator or publisher access rights can manage scheduled downtime rules.

To schedule downtime for a project:

1. Open the Configuration Details page for the project configuration. The Configuration Details page includes a Scheduled Downtimes section, as shown in Figure 4-39.

The screenshot shows the 'Configuration Details' page for a project named 'ITSO_Management'. The 'Scheduled Downtimes' section is highlighted with a red box. It contains two buttons: 'New Rule' and 'Delete'.

Figure 4-39 Scheduled Downtimes pane

2. Click **New Rule**, highlighted in Figure 4-39.
3. In the New Downtime Rule window, shown in Figure 4-40, select the start and end time for the downtime. You can also specify the actions to be taken when the time comes.

The 'New Downtime Rule' dialog box. It has the following settings:

- Action: Stop (allow jobs to finish)
- Start Downtime: 10/03/2011 08:00 PM
- End Downtime: 10/03/2011 09:00 PM
- Repeat: Every: 0

Figure 4-40 Downtime rule settings

Table 4-1 lists the actions for the downtime rules. One of the three actions listed must be selected for the Action field for a downtime rule.

Table 4-1 Schedule downtime actions

Action	Result
Stop (allows jobs to finish)	Prevents any new orchestration jobs from starting and allows current jobs to finish processing. This action facilitates a graceful stop.
Stop (cancel running jobs)	Prevents any new orchestration jobs from starting and stops all currently running jobs. This action immediately stops all jobs.
Suspend	Prevents any new orchestrations from starting and pauses any jobs that are currently running. At the end of the scheduled downtime, the run time processes the suspended job from the point where it paused.

The fields in the Repeat section in the New Downtime Rule window are optional. In Figure 4-41, the downtime rule is set to run repeatedly every two weeks.

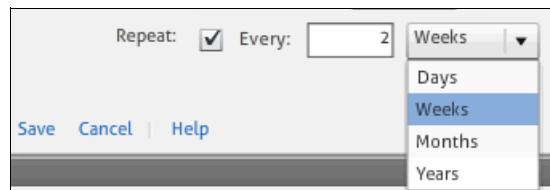


Figure 4-41 Using repeat configuration for downtime rule

4. Click **Save** to create and activate the rule. It is now listed in the Scheduled Downtimes pane. To edit a rule, click the link in the Action field. To delete a rule, select it, and click the **Delete** link, as shown in Figure 4-42.

Scheduled Downtimes (1)			
	Start Date	End Date	Action
<input checked="" type="checkbox"/>	10/03/2011 08:00:00 PM	10/03/2011 09:00:00 PM	Stop (allow jobs to finish)
New Rule Delete			

Figure 4-42 Changing an existing downtime rule

Hint: You can delete a group of multiple rules for the same project configuration at the same time by selecting multiple configurations and then clicking **Delete**.

Hint: By default, downtime rules are stored in the GMT time zone. If the WebSphere Cast Iron system clock is changed from GMT to another time zone, downtime rules start and end fields are adjusted for the new time zone. To keep the same start and end time with a new time zone, you must edit all rules and change the times appropriately.

4.2.13 Uploading and downloading a project

The WMC allows you to upload a project to the WebSphere Cast Iron runtime. A typical use case for uploading and downloading a project is to ensure that the development environment systems and users do not impact production systems. The administrator of a test integration appliance downloads the project, which is then uploaded into the production Integration Appliance, and then configured. You can ensure that Studio cannot be used to publish directly to the production Integration Appliance.

You can upload a project that was exported from Studio as a .par file using **Repository** → **Upload Project**, from the left navigation menu, as shown in Figure 4-43.

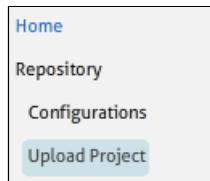


Figure 4-43 Upload a project

Click **Upload Project** to open the Upload Project Configuration window. This window allows you to browse for the project file in the file system. You can set the Project name and version for the project, as shown in Figure 4-44.

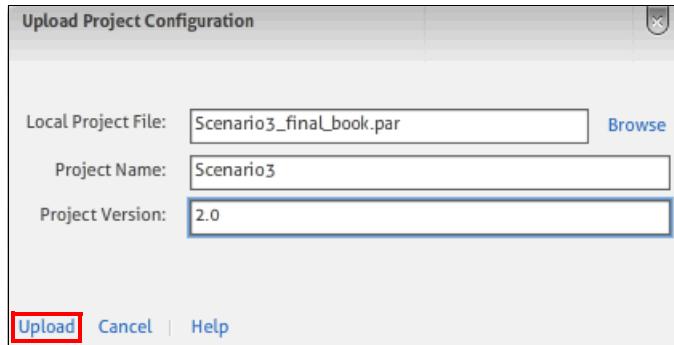


Figure 4-44 Upload project settings

Click **Upload** to upload the project into the WebSphere Cast Iron runtime.

Important: Some browsers can have problems when using Cast Iron with self-signed certificates. If you encounter such a problem, use another supported browser. For a list of supported browsers, see the system requirements for your Integration Appliance at:

<http://www-01.ibm.com/software/integration/cast-iron-cloud-integration/reqs/>

When uploaded, the project configuration is listed, as shown in Figure 4-45.

Project Configurations	
Configurations Filter: All	
Configuration	Last Published
► LogSampleProject (Running 1 of 2)	10/13/2011 06:36:18 PM
► Scenario3 {2.0} (Undeployed)	10/13/2011 08:45:18 PM

Figure 4-45 An uploaded project listed

Projects can also be downloaded. You might do this to then upload the same project to another WebSphere Cast Iron environment or to retrieve the source code for the project (stored with the project in the runtime).

To download a project, select the project configuration, and click **Download** in the Configuration Details page, as highlighted in Figure 4-46.

Home		Configuration Details
Repository		Summary
Configurations		Configuration: LogSampleProject # Orchestrations: 2
LogSampleProject		Status: Running # Properties: 0
Upload Project		Last Published: 10/13/2011 06:36:18 PM # Assets: 0
Import/Export		# Downtimes: 0
Download		Your content is ready Download Now

Figure 4-46 Download a project

The download creates a compressed file that contains project source code from the configuration that you selected.

4.2.14 Importing and exporting repository configuration

You can import and export the entire repository configuration, as described in 6.4, “Backup and restore” on page 285.

Cast Iron Live: This functionality is currently not available with Cast Iron Live.

4.3 Monitoring and troubleshooting

You will at times encounter situations where you must investigate issues that occurred in the WebSphere Cast Iron runtime. Monitoring can provide a means of tracking both the appliance and the running orchestrations.

Problem investigation is not the only reason to have monitoring. A common scenario that is related to monitoring is to make sure the orchestrations are running as expected by monitoring the results (through job keys, as described in 3.7.4, “The JobInfo variable and Job Keys” on page 122) and response time perspectives.

WebSphere Cast Iron contains useful tools to generate, visualize, and export logs. This section discusses logging, resource utilization, and notifications.

4.3.1 Logging overview

Logs are generated internally in WebSphere Cast Iron and can be managed through the WMC. You can use and manage system and job logs from the Logs menu, shown in Figure 4-47.



Figure 4-47 Logs menu

As illustrated in Figure 4-48 on page 200, WebSphere Cast Iron pushes log events to the system and job logs. Log events generated by orchestrations (job instances) go through job filters (illustrated as the smaller filters in the diagram) that can block events from being logged.

The system log determines the level of logs it will accept through an additional filter(the larger filter in the diagram). Although a log event can pass the job filter and be sent to the system log, the event can still be blocked.

In addition to job instance log events, WebSphere Cast Iron also produces log events that are not related to orchestrations, such as security, hardware, and network events. These log events are pushed to the system log.

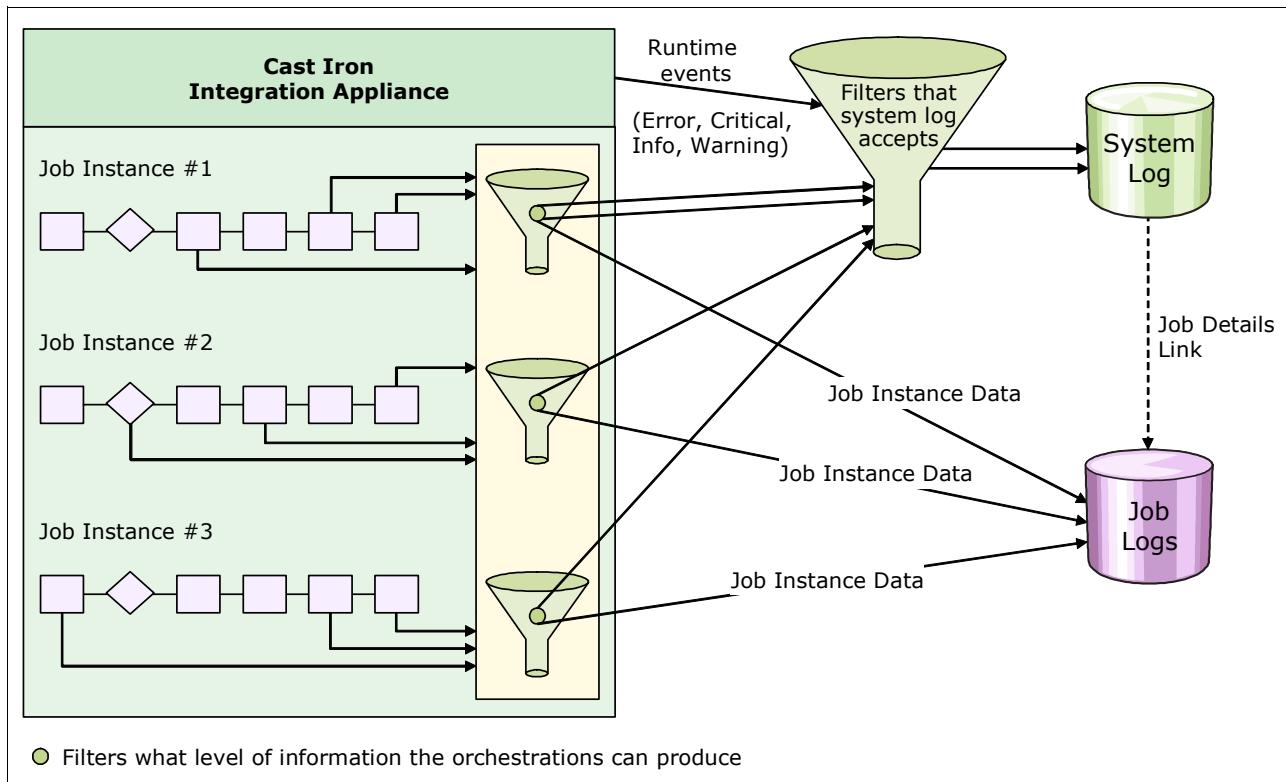


Figure 4-48 Logs architecture

4.3.2 The system log repository

The system log records entries that are related to occurrences in the Integration Appliance (system messages). The number and the type of entries can vary, depending on the log level set.

You can access this log by selecting **Logs** → **System Log**, as shown in Figure 4-49.



Figure 4-49 Accessing the system log repository

In addition to logging system messages, the system log also records events that are related to activities from running (or tried to run and failed) orchestrations, including logs from the Log Message activity.

Entry types

WebSphere Cast Iron records log entries in one of the following categories:

- ▶ Hardware
- ▶ Resources
- ▶ Network
- ▶ Security
- ▶ Orchestration
- ▶ Deployment

Logging levels

There are four logging levels that can be bound to events generated inside WebSphere Cast Iron, as listed in Table 4-2. As mentioned previously, these log levels (also known as *severity* levels) are produced by the run time and also by the job instances.

Table 4-2 Severities levels

Severity	Description
Critical	Used for error condition that causes a shutdown.
Error	Used for non-critical issues that must be handled immediately they occur.
Warning	Used when potential error conditions occur.
Info	Used for informational usage.

The System Log page has a configuration window that allows you to define the minimum logging level that an event must have to be recorded in the system log repository. At the bottom of the System Log page is the **Settings** link, as indicated in Figure 4-50. Use this link to change the settings.

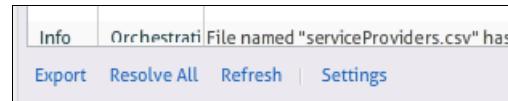


Figure 4-50 Settings link to change the log level

From the system log Settings window, choose the minimum log level that the system log records based on the event source, as shown in Figure 4-51.

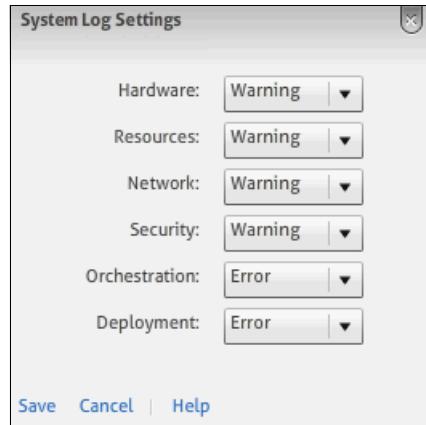


Figure 4-51 Modifying the log level

Cast Iron Live: Cast Iron Live system log settings contain only the Security, Orchestration, and Deployment settings.

Example log entries

Figure 4-52 on page 202 shows a log entry from a failed attempt to log in to the WMC. In this case, the log entry is not related to any orchestration.

System Log					
Minimum Level:		All Levels	System:	Security	Date: All Dates
Level	System	Message	Job	When	
Error	Security	Authentication failure for user: admin, from host: unl		10/05/2011	

Figure 4-52 Security log entry in the system log

Figure 4-53 shows log entries from an orchestration that tried to poll an FTP server using the wrong password. In this case, the system log contains five warnings messages and one error, all related to the event of not having access to the FTP server.

Error	Orchestration	Unable to connect to the server "192.168.116.1" after 5 attemp	10/05/201
Warning	Orchestration	Unable to conne Unable to connect to the server "192.168.116.1" after 5 attempts. Error is: 530 Login or password incorrect!	10/05/201
Warning	Orchestration	Unable to connect to the server "192.168.116.1". Error is: 530 L	10/05/201
Warning	Orchestration	Unable to connect to the server "192.168.116.1". Error is: 530 L	10/05/201
Warning	Orchestration	Unable to connect to the server "192.168.116.1". Error is: 530 L	10/05/201

Figure 4-53 Logs generate for a FTP failed login

For FTP, both the number of retries and the time between each retry are defined in the Poll Directory activity during the development phase. For more information, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/toc_FTPactivities.html

Notice that these logs belong to the Orchestration category because the error is related to a job instance that ran and is not related to the Integration Appliance.

The log entry in Figure 4-54 shows a failed attempt to parse the input data. The data can be corrupted or can contain the wrong format. Some system log entries can also contain links to job instances logs.

System Log					
Minimum Level:		All Levels	System:	Orchestration	Date: All Dates
Level	System	Message	Job	When	
Error	Orchestratio	Parsing Error: Error Parsing node CustomerID (/AA48E018BCB78AB6EF62BF19BDD639D		10/17/2011 11:08:21 PM	
Error	Orchestratio	Parsing Error: Error Parsing node CustomerID (/B7817411DCFF12C74FADE1534AA48E018B		10/17/2011 11:08:21 PM	

Job Details - AA48E018BCB78AB6EF62BF19BDD639DE	
Summary	Job Keys
Details (circled in red)	
Job ID: AA48E018BCB78AB6EF62BF19BDD639DE	Status: Completed in 0.30s
Configuration: Scenario3 {1.0} [Default]	Start Time: 10/17/2011 11:08:21 PM
Orchestration: FTP_DB2_to_Domino_sync_Orders	End Time: 10/17/2011 11:08:21 PM

Figure 4-54 Log entry error from a parsing error

Click Details, highlighted in Figure 4-54, to show more information in the Job Details section, shown in Figure 4-55 on page 203. This level of detail with activities and its variable values is provided when the job log level is set to All.

Job Details - AA24FFD2BC531F9799B54AFE804432DA					
Summary Job Keys Details			Displaying variables and events for activity: FTP - Poll FTP Directory		
	Elapsed	Activity	Variable		
	0.014s	FTP Poll FTP Directory	timestamp (output)		
■	0.000s	Transform Read CSV file	filename (output)		
			data (output)		
			dataRow		
			Total		
			Filtered		

Figure 4-55 Job Details panel

If the log level is set to Error Values, the Job Detail panel is still present but shows only the activity where the error happened, as shown in Figure 4-56. You can read more about job detailed logs in 4.3.3, “Job log” on page 206.

Job Details - 9C7F7B224485E798E66A046259C73115					
Summary Job Keys Details			Displaying variables and events for activity: Transform - Read CSV file		
	Elapsed	Activity	Variable		
■	Unknown	Transform Read CSV file	Data (input)		
			Xml (output)		
			dataRow		
			Total		
			Filtered		
			JobInfo		

Figure 4-56 Job details for an error message with log level set to Error Values

Sending entries to the log with the Log Message activity

The system log also records log events that are generated manually by the developer inside the orchestration through the Log Message activity. The sample orchestration in Figure 4-57 contains a Log Message activity that logs a fixed message for debugging purposes.

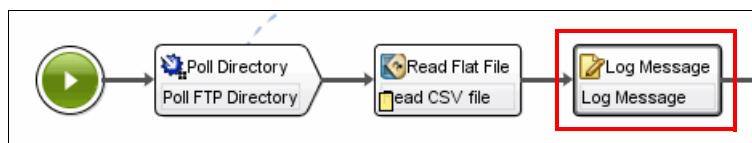


Figure 4-57 Log Message activity

The System log page now contains the message created by the Log Message activity, as shown in Figure 4-58.



Figure 4-58 Log entry created by the Log Message activity

Important: Although an error is caught in a try/catch or catch all block, one or more log entries are still recorded in the system log.

Viewing system log entries in the WMC

The page at **Logs → System Log** provides a filter, shown in Figure 4-59, that allows you to display only the log entries in which you are interested.



Figure 4-59 System log filters

For example, you can select the type of log entries to view by selecting one of these categories in the System drop-down, as shown in Figure 4-60. Additional filters allow you to filter the view by logging level, date, and resolved or unresolved entries.

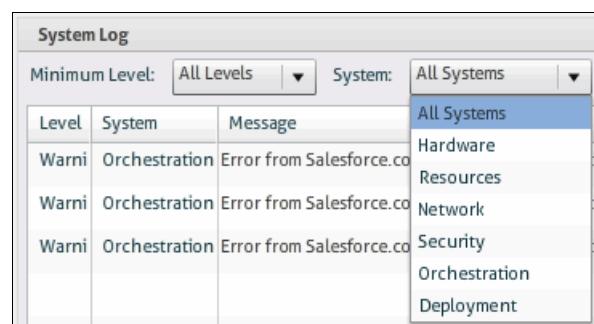


Figure 4-60 System categories used by the system log

You can also sort columns by clicking the column header, as shown in Figure 4-61.



Figure 4-61 The arrow on the right side indicates the sorted column

For each entry in the system log, you can keep track of whether or not an error or warning was resolved. You can mark one or more entries as resolved, which automatically completes the other two fields, Resolved On and Resolved By, as indicated in Figure 4-62.

Flagging a log entry as resolved helps you to identify records that are not resolved so far. There is an Unresolved filter to find them.

Level	System	Message	Job	When	Resolved On	Resolved By	Resolved
Error	Orchestrator	Encountered fault	8237C350	10/05/2011			<input type="checkbox"/>
Error	Orchestrator	Parsing Error: Err	8237C350	10/05/2011	10/05/2011 0	admin	<input checked="" type="checkbox"/>
Warning	Orchestrator	Could not delete f		10/05/2011	10/05/2011 0	admin	<input checked="" type="checkbox"/>

Figure 4-62 Resolved columns on the System Log page

Hint: On the bottom of the System Log page, click **Resolve All** to change all entries to the resolved state.

You can also access the System Log page through the dashboard. There is Critical Alert panel on the right of the dashboard that shows Error and Critical alerts. You can use this panel to link to the System Log page by clicking the plus sign (+) on the upper-right corner, as shown in Figure 4-63.

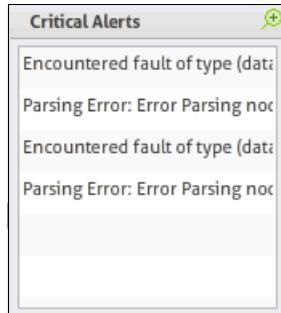


Figure 4-63 Critical Alerts panel

Exporting the system log

System log entries can be exported to an XML file where you can further analyze the log entries using any log viewer tool that supports XML files. The default name for the exported file is `ia_export_syslog.xml`. Only administrators can export logs:

1. Click **Export** on the bottom of the System Log page to create a system log export file, as shown in Figure 4-64.

System Log			
Minimum Level:	All Levels	System:	All Systems
Level	System	Message	Job
Error	Orchestrati	Encountered fault of type (data.parse.error), F	9A0
Error	Orchestrati	Parsing Error: Error Parsing node profession (data.parse.error)	9A0
Warn	Orchestrati	Could not delete file "serviceProviders.csv" af	
Warn	Orchestrati	Could not delete file "serviceProviders.csv" af	

Export [Resolve All](#) [Refresh](#) [Settings](#)

Figure 4-64 Export link at the System Log page

2. Click **Download Now**, as shown in Figure 4-65.

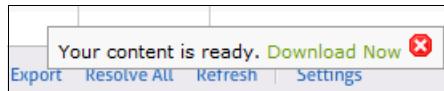


Figure 4-65 Download the XML file

Exactly the same number of entries that are filtered for display in the System Log page are exported to the file. You can narrow the entries that are exported to the file by changing the filter values on the filter header row.

Example 4-3 shows one log entry that was extracted from an exported system log XML file.

Example 4-3 A log entry in the exported file

```
<log xmlns:ns2="http://www.approuter.com/schemas/2008/1/lognotif/types">
  <ns2:system>security</ns2:system>
  <ns2:message>Authentication failure for user: admin, from host:</ns2:message>
  <ns2:id>3</ns2:id>
  <ns2:level>error</ns2:level>
  <ns2:timestamp>2011-10-05T15:22:11.179Z</ns2:timestamp>
</log>
```

Important: When upgrading WebSphere Cast Iron, the logs are deleted. To retain system log information, export system logs before you begin the upgrade.

Cast Iron live: The functionality for exporting system logs is currently not available with Cast Iron Live

4.3.3 Job log

The job log records the results of a job instance (or orchestration instance), including completion status, time stamps, and predefined business data that is captured into job keys. The amount of information depends on the log level.

Log levels

Each orchestration in a project can have the log level set individually or all orchestrations can have the same log level. Table 4-3 lists the available log levels.

Table 4-3 Log levels for orchestrations

Log Level	Description
None	Orchestration status and activity are not logged.
Initial Values	Only the initial values of orchestration variables are logged.
Initial and Error Values	Both the initial values of orchestration variables and orchestration errors are logged.
Error Values	Only orchestration errors are logged.
Inline	Orchestration job details are logged for sub-orchestrations inline with the job details of the calling orchestration.
All	The highest level of logging. All orchestration activity is logged.

To view or change a log level for an orchestration or orchestrations, select **Repository → Configurations**, and then select the target project configuration.

As shown in Figure 4-66 on page 207, the Configuration Details page shows the current logging level set for each orchestration. Click **Edit**, which is highlighted in Figure 4-66 on page 207, to open the Edit Orchestration Settings page, where you can change the logging level setting and other parameters. The **Edit** link is enabled only if the configuration is undeployed.

Orchestrations (2)		
Name	Status	Logging Level
CVSToSugar	Enabled	Error Values
CreateServicesPeopleCatalog	Enabled	All
Edit		

Figure 4-66 Orchestration details pane

Figure 4-67 shows the Edit Orchestration Settings page and the parameters that you can change for each orchestration. The logging level defaults to Error Values for a newly published project. You can change the logging level for each orchestration, or you can apply a change for all orchestrations using the unlabeled header row.

Edit Orchestration Settings				
Name	Enabled	Logging Level	Log Synchronously	Max Simultaneous Jobs
	<input type="checkbox"/>	None	<input type="checkbox"/>	<input checked="" type="checkbox"/> Unlimited <input type="text"/>
CVSToSugar	<input checked="" type="checkbox"/>	Error Values	<input type="checkbox"/>	<input type="checkbox"/> Unlimited <input type="text" value="10"/>
CreateServicesPeopleCatalog	<input checked="" type="checkbox"/>	All	<input type="checkbox"/>	<input type="checkbox"/> Unlimited <input type="text" value="10"/>

Figure 4-67 Edit Orchestration Settings page

Even if the logging level is set to Error, the Job Log still records a successfully completed execution entry.

The Log Synchronously field allows the run time to synchronize the writing of job instance events to the monitoring logs as the orchestration is running. By default, this setting is disabled, and orchestration jobs are not logged synchronously. When you enable Log Synchronously, the time it takes to process an orchestration job increases.

In summary, the logging level field determines how much log information is produced for a job instance execution. You can then read the generated log information in the Job Log page. The details recorded in the log can change according to the corresponding level that is selected.

Viewing log records

Job log records are viewed by selecting **Logs** → **Job Log**. Figure 4-68 on page 208 shows a list of job instance results. Selecting an entry in the list of entries and then clicking **Details** will show additional information for the selected entry. The amount of information shown depends on the log level. If the log level is set to All, the information includes all activities that ran and the contents of the variables. This pane can be quite useful for debugging orchestrations.

The screenshot shows the 'Results' section of the Job Log page. On the left, there's a sidebar with links like Home, Repository, Logs, Job Log, System Log, Notifications, Archived Logs, Security, Network, and System. The main area has a 'Jobs Filter' dropdown set to 'All' and a 'Key Search' input field. Below that is a table with columns for 'Configurations', 'Orchestrations', and 'Status'. The 'Status' column contains several entries: 'Errored' (highlighted with a red box), 'Completed in 0.21s', 'Completed in 1.52s', and 'Completed in 1.52s'. At the bottom of the table are 'Export', 'Refresh', and 'Settings' buttons. Below the table is a section titled 'Job Details - 9ACABBOA2DD74BF5E0EB96D6D8DE3811' with tabs for 'Summary', 'Job Keys', and 'Details' (which is selected). This section displays variables and events for an activity named 'FTP - Poll FTP Directory'. It shows two rows: one for 'FTP' (Elapsed: 0.012s) and one for 'Transform' (Elapsed: 0.000s). To the right of these rows is a 'Variable' table with entries: timestamp (output), filename (output), data (output), and dataRow.

Configurations	Orchestrations	Status
LogSampleProject	CreateServicesPeopleCata	Errored
LogSampleProject	CreateServicesPeopleCata	Errored
LogSampleProject	CreateServicesPeopleCata	Completed in 0.21s
LogSampleProject	CreateServicesPeopleCata	Completed in 1.52s
LogSampleProject	CreateServicesPeopleCata	Completed in 1.52s

Variable
timestamp (output)
filename (output)
data (output)
dataRow

Figure 4-68 Job Log page showing details for an errored job instance

One of the log levels listed in Table 4-3 on page 206 is the **Inline** level. When one orchestration calls another (called a *suborchestration*), using the **Inline** setting causes the job details for suborchestration to be logged inline with the job details of the calling orchestration.

Figure 4-69 shows detailed log information that includes data from both orchestrations. The highlighted area shows activities from the suborchestration.

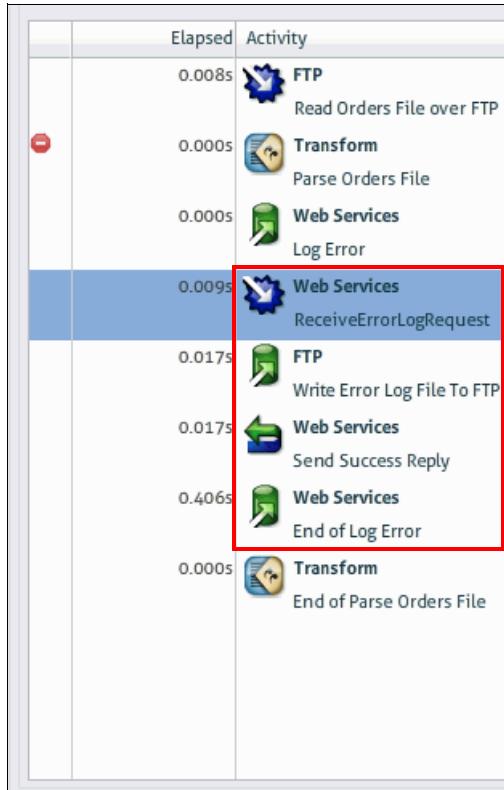


Figure 4-69 Activities from suborchestration highlighted

Click **Job Keys** in the Job Details section to show the job keys and their values, as shown in Figure 4-70.

Job Details - 81246007AE386CBA1EC4E427869B32DB	
Summary Job Keys Details	
Key	Value
Total	5
Filtered	4

Figure 4-70 Job Keys for a particular job instance

Finally, click **Summary** in the Job Details section to display general job instance information, including completion time, as shown in Figure 4-71.

Job Details - 81246007AE386CBA1EC4E427869B32DB	
Summary Job Keys Details	
Job ID:	81246007AE386CBA1EC4E427869B32DB
Configuration:	LogSampleProject {1.0} [Default]
Orchestration:	CreateServicesPeopleCatalog
Status:	Completed in 0.81s
Start Time:	10/05/2011 06:03:11 PM
End Time:	10/05/2011 06:03:11 PM

Figure 4-71 Summary panel with job instance detail data

Hint: You can search for a job log entry using the job ID in the Key Search field.

Exporting job logs

You can export orchestration log data from the job logs to an XML file for further analysis using any viewer tool that supports XML files. Example 4-4 shows one log entry that is extracted from an exported job logs XML file. The default name for the exported file is ia_export.xml.

Example 4-4 Job log entry in the exported log

```
<job ... status="1" startTime="2011-10-06T16:01:35.481Z"
path="LogSampleProject/1.0/Default/Orchestrations/CreateServicesPeopleCatalog"
id="9C21C2B28C105251C072E323515A3C69" endTime="2011-10-06T16:01:35.555Z">
  <ns3:event seqNum="0">
    <ns3:eventType>
      <taskStartLoggedEvent>
        <taskId>9C21C2B28C105251C072E323515A3C69</taskId>
        <sequenceNumber>0</sequenceNumber>
        <time>2011-10-06T16:01:35.481Z</time>
      </taskStartLoggedEvent>
    </ns3:eventType>
  </ns3:event>
...
</job>
```

The job logs can contain detailed information, depending on the log level set. Example 4-5 shows an example of a detailed job log entry with internal activity data exported to an XML file.

Example 4-5 Detailed job log entry

```
<ns3:eventType>
  <saveVarsLoggedEvent>
    <taskId>943F1E28D4D94C6EE670902110A73494</taskId>
    <sequenceNumber>4</sequenceNumber>
    <activityId>4</activityId>
    <variableName>dataRow</variableName>
    <variableName>Total</variableName>
    <variableName>Filtered</variableName>
    <variableName>JobInfo</variableName>
  </saveVarsLoggedEvent>
</ns3:eventType>
<ns3:variable name="dataRow">
  <ns3:value>122234 Giuliano,Builder,Sao Paulo,567 Leonardo,Engineer,Mexico
City,55</ns3:value>
</ns3:variable>
<ns3:variable name="Total">
  <ns3:value>0</ns3:value>
...
...
```

To create a job log export file, click **Export** on the bottom of the Job Log page, as shown in Figure 4-72 on page 211, to open the Export Jobs window.

Results		
Jobs Filter:	All	Key Search:
Configurations	Orchestratio	Status
LogSampleProject	CreateServicesPe	Completed in 0.9
LogSampleProject	CreateServicesPe	Completed in 0.3
LogSampleProject	CreateServicesPe	Completed in 0.4
LogSampleProject	CreateServicesPe	Errored
Export	Refresh	Settings

Figure 4-72 Export link for Job Logs

The Export Jobs window gives you the option of exporting or exporting and deleting the job logs, as shown in Figure 4-73. The Export and Delete option generates the export file and deletes the data from the job logs repository.

Only administrators can export logs.

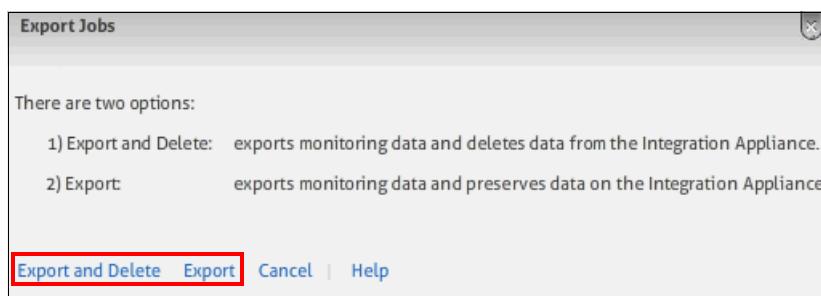


Figure 4-73 Export Jobs window

Purging job logs

The orchestration job log entries can be purged and archived to clean and free up the job log. From the WMC, you can create a configuration that purges and archives job logs automatically. Click **Settings** on the bottom of the Job Log page to open the Job Log Settings window with the Purge and Archive links, as indicated in Figure 4-74.

Cast Iron Live: This functionality is currently not available with Cast Iron Live.



Figure 4-74 Purge and archive job logs

The Purge action deletes job log entries based on criteria, as defined in Figure 4-75 on page 212.

Job Log Settings

Purge [Archive](#)

Trigger Conditions

Purge data when one of the following conditions is satisfied:

Available disk space: %

Schedule: Variable Fixed
 days hours minutes since last purge

Completed job count: jobs

Errored job count: jobs

Frequency

Configure how often trigger conditions are checked for:

Check trigger conditions every: hours minutes seconds

Job Scope

Purge only those jobs that satisfy these conditions:

Job status: Completed Any

Jobs older than: months days hours minutes

Jobs older than the most recent: jobs

Figure 4-75 Purge settings

The purge settings are divided into the following sections:

- ▶ Trigger Conditions: Settings that determine the conditions that trigger the action to start purging data. These conditions include disk usage space or a scheduled date or number (variable or fixed) of job instance occurrences.
- ▶ Frequency: Determines how often the trigger conditions are checked for.
- ▶ Job Scope: Sets a filter that allows you to select the job log entries that are suitable to be purged.

Job log entries can be archived before the job log is purged using the Archive feature (as described in the next section). If you do not enable the archiving settings, all the purged data is deleted.

Archiving job logs

For auditing purposes, save the job logs before you purge the data. You can archive purged logs locally on the appliance, or send the logs to an FTP server. Click **Archive** in the Job Log Settings window, as shown in Figure 4-76 on page 213.

Cast Iron Live: This functionality is currently not available with Cast Iron Live.

The screenshot shows the 'Job Log Settings' page with the 'Archive' tab selected. It includes sections for 'Archive Settings' and 'FTP Settings'. Under 'Archive Settings', there are checkboxes for 'Archive purged monitor data to a local file' (checked), 'Replace exported file' (checked), 'Transmit archive to FTP Server (after a purge)' (checked), and 'Transmit archived utilization data to FTP Server (at midnight)' (unchecked). Under 'FTP Settings', fields are provided for 'Host' (192.168.1.100), 'Port' (21), 'Username' (castironftp), and 'Password' (*****). A checkbox 'Hide password text' is checked. Below these, there's an 'Archive Download' section with a dropdown menu 'Select archive to download' containing 'Monitor data for 2011-10-14 00:08' and a 'Download' button.

Figure 4-76 Archive settings

You can download the local stored files by selecting the file in the Select archive to download section and clicking **Download**, as shown in Figure 4-77.

The screenshot shows the 'Archive Download' interface. It has a 'Select archive to download' dropdown menu containing 'Monitor data for 2011-10-14 00:08' and a 'Download' button below it.

Figure 4-77 Download archived log

Hint: Enable the Replace exported file option to avoid excessive disk consumption. Running out of disk space also triggers the purge action more often.

For more information about Job Logs purge and archive, visit:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.appliance.doc/Working_with_Logs/purgingJobLogs.htm

Tip: For troubleshooting purposes, there is a postmortem log. You can generate and download the postmortem log from the Commands page. For more information, refer to 4.7, "Commands" on page 225.

Some internal logs are also available by selecting **Logs → Archived Logs**.

4.3.4 Notifications

WebSphere Cast Iron can generate alert notifications, allowing you to monitor the system for certain events. Email and SNMP are the target destinations for notifications.

You can access the notifications page by selecting **Logs → Notifications**, as shown in Figure 4-78 on page 214.

Home Repository Logs Job Log System Log Notifications	<p>Notifications</p> <p>Policies (0) Email (inactive) SNMP (active)</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Policy Name</th><th style="text-align: right; padding: 2px;">Sy</th></tr> </thead> <tbody> <tr> <td style="height: 40px;"></td><td style="text-align: right; height: 40px;"></td></tr> </tbody> </table>	Policy Name	Sy		
Policy Name	Sy				

Figure 4-78 Notification page

Before configuring the policies for notification generation, you must configure the target destinations. You can configure email or SNMP. If the target destinations are not configured and enabled, notifications are not created.

To configure the email, click **Email**. Figure 4-79 shows the email settings. Click **SNMP** to set up the SNMP target destination.

Notifications

Policies (0) [Email \(inactive\)](#) [SNMP \(active\)](#)

Enable Email: (Send Email Notifications via SMTP)

Host: smtp.ibm.com

Port: 25

Sender's Email: notification@castiron.com

Subject: Notification Alert from Cast Iron

SMTP Requires Authentication:

Userid:

Password:

Hide Characters

Save

Figure 4-79 Email setup for notification

You can download the WebSphere Cast Iron Management Information Base (MIB) file to import it into a SNMP server or a monitor tool, as shown in Figure 4-80.

Notifications

Policies (0) [Email \(active\)](#) [SNMP \(active\)](#)

Enable SNMP: (send SNMP Notifications via:)

SNMP Host: 192.168.116.1

Trap Community: castiron

Save **Download MIB**

Figure 4-80 SNMP setup

After you configure the notification destinations, either email or SNMP, notice that their status changes to active, as indicated in Figure 4-81.

Next, you must create policies. Policies are the actual alert configuration. When creating a policy, you can determine the run time systems and severity level for which you want the system to generate notifications. Click **Logs** → **Notifications**, and then click **New Policy** at the bottom of the page to create a policy, as shown in Figure 4-81.

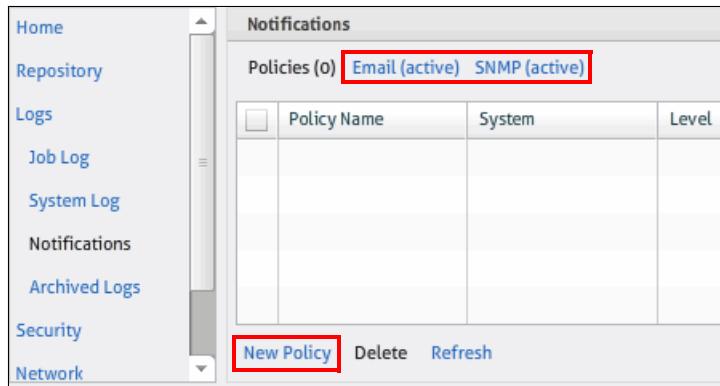


Figure 4-81 New Policy link

In the configuration for the New Policy, you must select the minimum notification level and the system level the notification engine will monitor to generate notification alerts. Figure 4-82 shows an example of how to set the name, log level, and system. In the Email pane, you can enter email addresses to receive this notification.

Although the target destinations can be active, for each policy you determine whether or not this destination is to be used. Figure 4-82 shows that both Email and SNMP are being used as destinations for this policy.

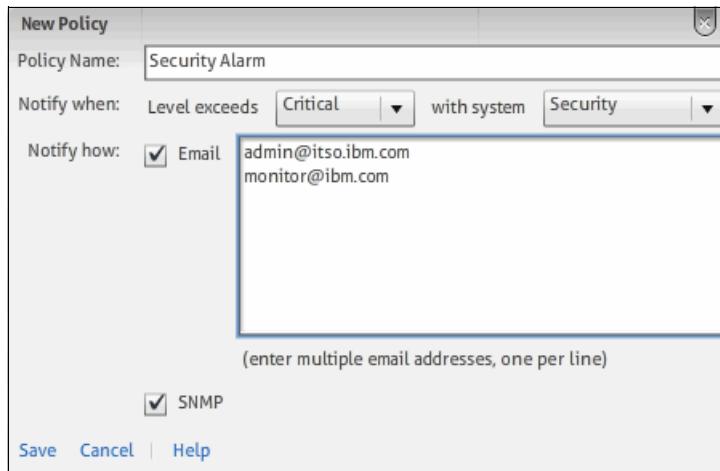


Figure 4-82 New Policy settings

After saving the policy, you can view the list of all policies in the notifications page. You can delete a policy by selecting the policy and then clicking **Delete**, as shown in Figure 4-83 on page 216.

Notifications					
Policies (1) Email (active) SNMP (active)					
	Policy Name	System	Level	Email (active)	SNMP (active)
<input checked="" type="checkbox"/>	Security Alarm	Security	Critical	admin@itso.ibm.	✓

New Policy **Delete** Refresh

Figure 4-83 Deleting a notification policy

Hint: Refer to the CLI documentation to learn more about getting SNMP traps that are not related to notification alerts. Refer to information about the `mgmt snmp` command at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.cli.doc/CLI_about_CLI.html

Cast Iron Live: For Cast Iron Live, you cannot generate SNMP notifications.

4.4 Network

Chapter 2, “Installing and setting up WebSphere Cast Iron integration” on page 27 discusses how to configure network settings through the Command Line Interface. The WMC also provides administrators a page to change network settings. Select **Network** in the menu to access the network page settings, as shown in Figure 4-84.



Figure 4-84 Network page link

The Network Settings page displays the current network settings and allows you to change several network parameters, including the management and data interface configuration, routes, host name, DNS, and time server among others. Click **Edit**, shown in Figure 4-85 on page 217, to change these configurations.

Cast Iron Live: This functionality is not available with Cast Iron Live.

The screenshot shows the Network Settings page with the following configurations:

- Data Network:** Use DHCP: checked, IP: 192.168.116.136, Subnet: 255.255.255.0, Broadcast: 192.168.116.255.
- Management Network:** Use DHCP: checked, IP: 192.168.116.135, Subnet: 255.255.255.0, Broadcast: 192.168.116.255.
- Gateway:** Interface: Management, Source: DHCP, Manual IP: 192.168.116.2.
- Host Name:** DHCP Management selected, Host Name: ci-192-168-116-135.
- Domain:** DHCP Management selected, Domain: localdomain.
- DNS:** DHCP Management selected, DNS: 192.168.116.2.
- Time Server:** DHCP Management selected.
- Routes (0):** Click on New Route to configure network routes. Buttons: New Route, Delete.

At the bottom left are **Edit** and **Refresh** buttons.

Figure 4-85 Network Settings page

You can change the configuration using a four-step wizard, as shown in Figure 4-86. The last step is to restart the appliance to apply your changes. At the bottom of the screen, **Previous** and **Next** allow you to move between the wizard steps.

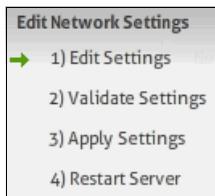


Figure 4-86 Four steps wizard for network settings

You can find more information at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.appliance.doc/Managing_Integration_Appliances/specifyingNetworkSettings.htm

Hint: If a TimeServer is set, you cannot change the time or time zone using the CLI.

4.5 The staging database

For performance reasons, using a local database as a temporary data repository is a good option. WebSphere Cast Iron provides an internal database for such purposes, called the

Staging DB. The Staging DB can be used by any database-related activity from Studio. It is frequently used by Data Quality activities, such as Lookup, Sort, and Merge.

When you need to merge data flowing inside the orchestration with data from a database table that does not change often, use the Staging DB.

To configure activities in Studio to use the Staging DB, create a database endpoint that is configured to Local, and specify the appliance data IP address and a valid user name and password, as shown in Figure 4-87. The user name and password must be for a user with Publisher privileges on the Integration Appliance. The staging database must be started using the WMC before it can be accessed.

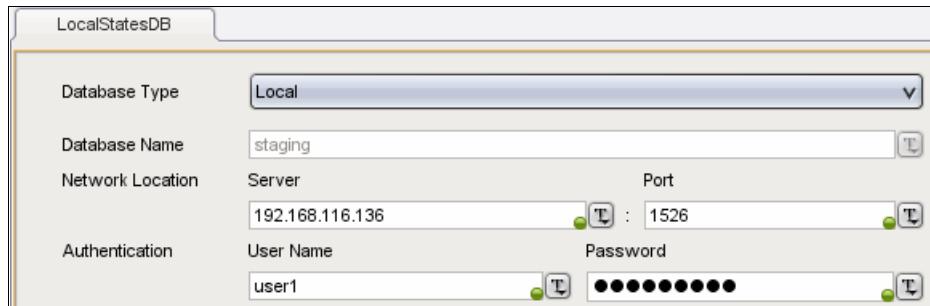


Figure 4-87 Endpoint defined as Local to use the Staging DB

Cast Iron Live: This functionality is not available with Cast Iron Live.

In the WMC, you can manage the Staging DB through the **System → Staging DB** menu, as shown in Figure 4-88.

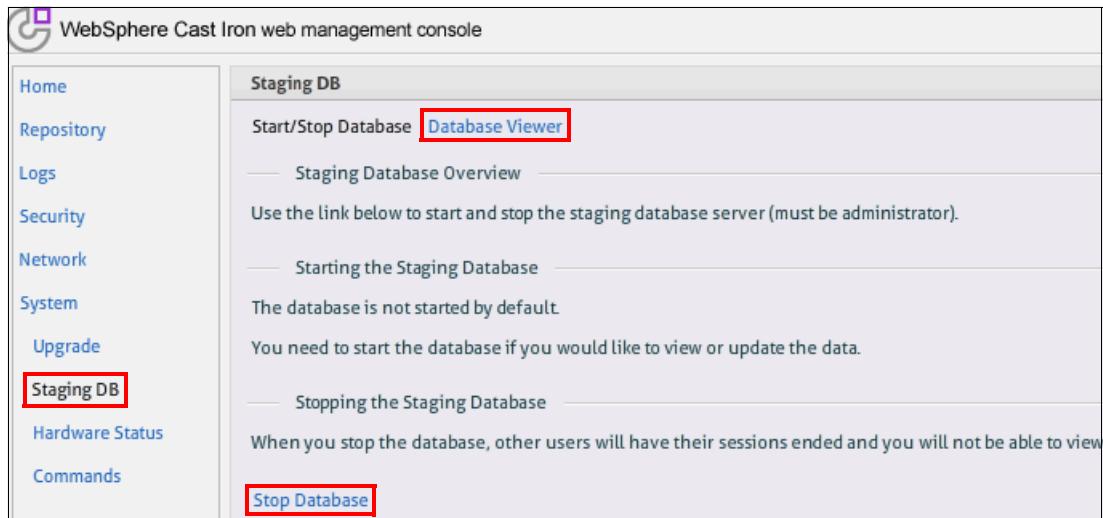


Figure 4-88 Staging DB menu

On the Staging DB page, you can start and stop the database and access the Database Viewer. By default the Staging DB is not started. The **Start Database** and **Stop Database** link at the bottom of the page is a toggle switch. When the database is started, you have the Stop Database link.

Only administrators are allowed to start or stop the Staging DB.

After the Staging DB is started, click **Database Viewer** to manage the database content. The Database Viewer allows you to create database tables and data and to validate queries. Only administrators or publishers are allowed to manage database table structures.

Click the APP folder inside the Schema pane to create a new database table, as shown in Figure 4-89.

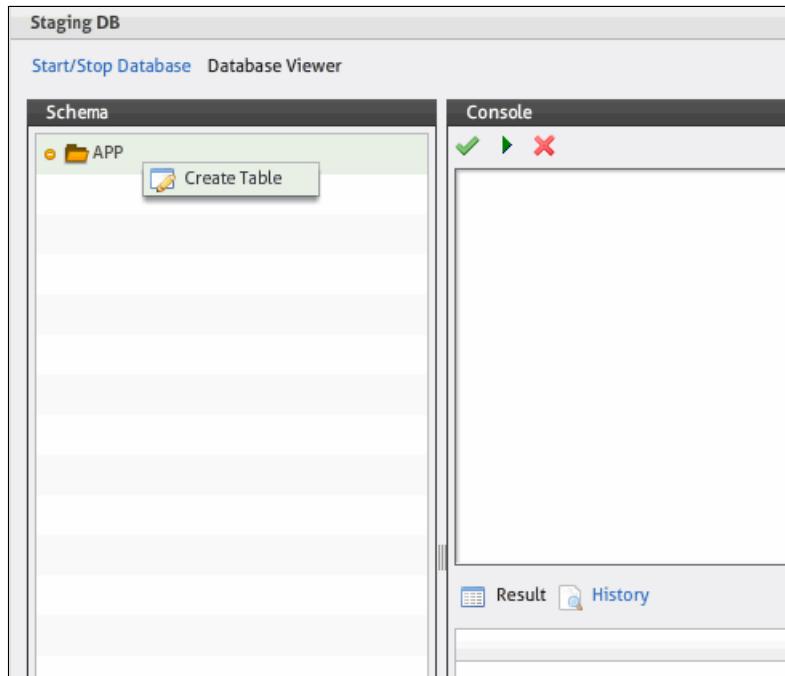


Figure 4-89 Creating a database table

The Create Table page, shown in Figure 4-90, provides an intuitive interface to create the new database table structure.

Create Table

Table Name: states Insert Column Delete Column

Column Name	Data Type	Length	Precision	Scale	Allow nulls?	Primary Key
country	VARCHAR	20			<input type="checkbox"/>	<input checked="" type="checkbox"/>
statecode	VARCHAR	2			<input type="checkbox"/>	<input checked="" type="checkbox"/>
statename	VARCHAR				<input type="checkbox"/>	<input type="checkbox"/>
	SMALLINT					
	TIME					
	TIMESTAMP					
	VARBINARY					
	VARCHAR					

Save **Cancel**

Figure 4-90 Insert columns into a database table

After the database table is created, its structure is displayed in the left pane. There are two other panes in this page—the Console and the Results pane. Using these two panes, you can issue and validate SQL statements to the database and check the results, as shown in Figure 4-91.

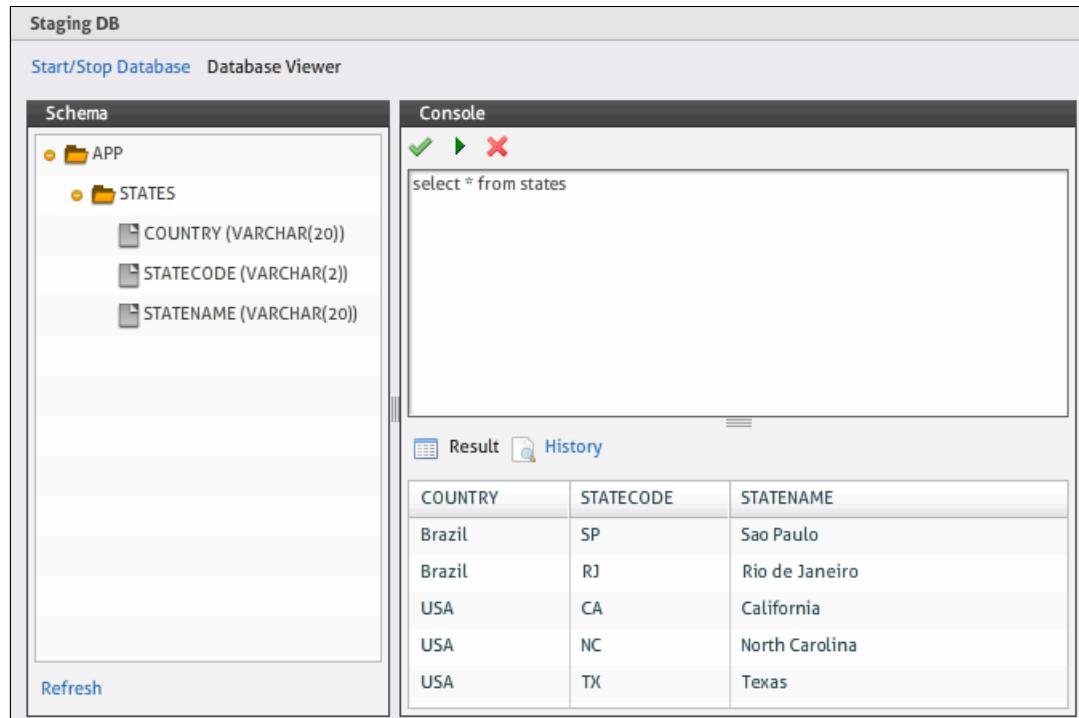


Figure 4-91 Console and Results panel

Database scripts: In “Database scripts” on page 190, you read that database scripts are created for you. If you are using a local database (the Staging DB), the database assets are created in the Staging DB.

4.6 Resource utilization

Monitoring hardware status and utilization is helpful for identifying bottlenecks and low resource availability. This type of information is available to all users and is not limited to the administrator.

Cast Iron Live: This functionality is not available with Cast Iron Live.

The dashboard page (**Home → Dashboard**) of the WMC, among other views, provides an overview of resource utilization. It displays a small graph in the top-right corner with information that is related to disk and memory usage, garbage collector activity, and the number of running jobs, as illustrated in Figure 4-92 on page 222.

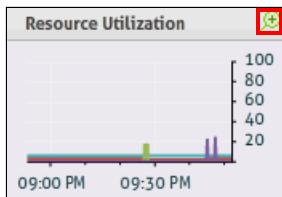


Figure 4-92 Resource Utilization pane on the dashboard

Click the plus sign (+) highlighted in Figure 4-92 to open an expanded Resource Utilization page. You can also access this page by clicking **Home** → **Utilization**. Figure 4-93 displays the expanded Resource Utilization window.

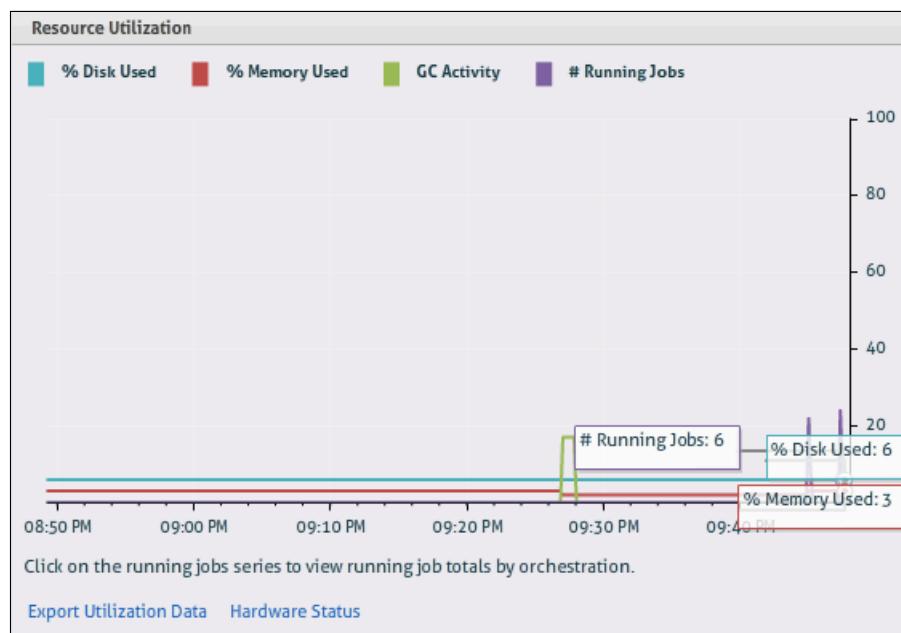


Figure 4-93 Resource utilization window

The % of Disk Used data allows you to identify whether WebSphere Cast Iron is running out disk space. If WebSphere Cast Iron runs out of disk space, you can take actions, such as turning off the persistence option (as described in 3.4.3, “Orchestration properties” on page 99) for orchestrations that no longer save job instance variables data into the disk. You can also change the logging level to record less data or purge logs.

The % of Memory Used data allows you to monitor memory utilization inside WebSphere Cast Iron. A high level of memory utilization might imply an out of memory error. To avoid this type of error, you can change the Max Simultaneous Jobs field for an orchestration to limit the number of parallel job instances that can run simultaneously. You can also turn on the persistence for an orchestration to save job instance data in the disk using less memory. In addition, you can schedule downtime to allow an orchestration that uses more hardware resources (a batch job for example) to run when more hardware resources are available. In this case, the scheduled downtime can bring down other orchestrations for a certain amount of time, freeing resources to the orchestration in need of more memory.

The GC Activity data displays the garbage collector activity. The garbage collector is a process within the run time Java virtual machine (JVM) that returns memory that is held by completed jobs and variable data back to the pool of memory used by orchestrations, as shown in Figure 4-94 on page 223. This garbage collection gives the orchestration engine a

constant supply of free memory to be used by new orchestrations. When the memory utilization is over the 80% threshold, the garbage collection is more intensive and can affect the orchestrations response times.

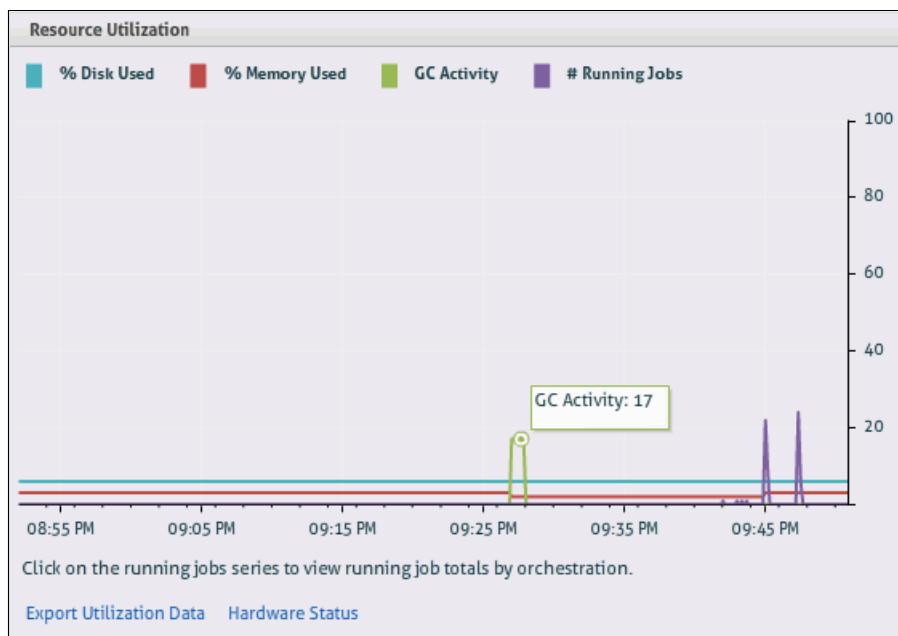


Figure 4-94 The GC Activity

Finally the “# Running Jobs” data graphically displays the number of jobs instances at a point in time indicated by the horizontal axis (time line). Click in the # Running Jobs line to obtain details about the orchestration that ran at that certain point in time, as shown in Figure 4-95.

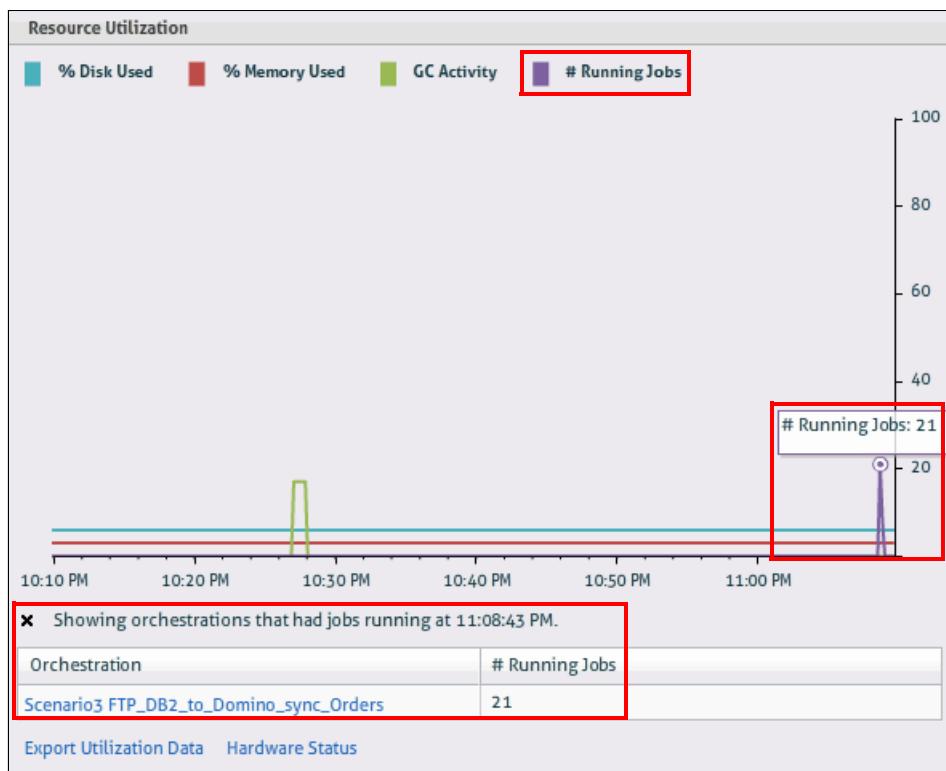


Figure 4-95 Orchestration details for Running Jobs

You can view the hardware status by clicking **Hardware Status** located on the bottom of the Resource Utilization windows, as shown in Figure 4-96. You can also view this page by selecting the **System → Hardware Status** menu.

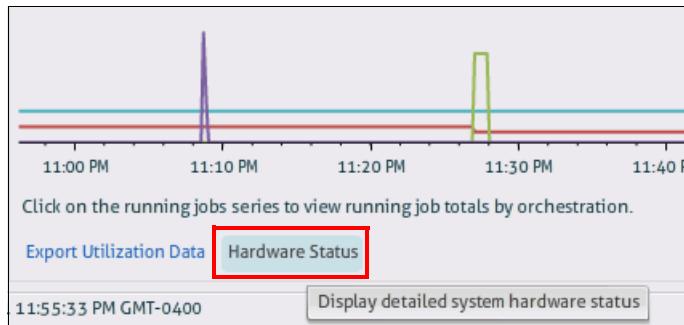


Figure 4-96 Hardware Status link

The Hardware Status shows the resource utilization in a bar graph. This page displays CPU, RAM, and disk consumption as a percentage, as shown in Figure 4-97.

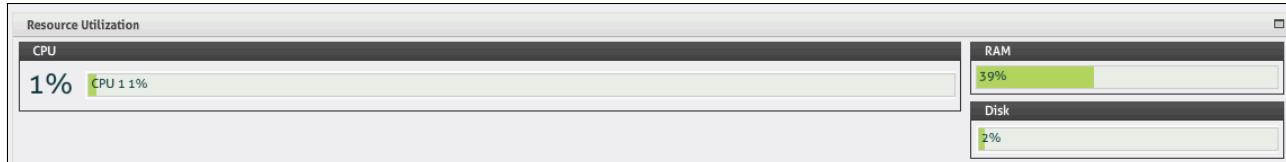


Figure 4-97 Hardware Status bar

For further analysis, you can also download the results from the graph as a comma-separated value (CSV) file. Click **Export Utilization Data** to download the file, as shown in Figure 4-98.



Figure 4-98 Downloading utilization data

Example 4-6 shows an example of this file.

Example 4-6 Utilization data

```
Time,% Disk Used,Memory Used,% Memory Used,GC Count,Total Running Jobs
Tue Oct 18 01:44:43 GMT 2011,6,37666120,2,122,0
Tue Oct 18 01:44:53 GMT 2011,6,37666120,2,122,0
Tue Oct 18 01:45:03 GMT 2011,6,39876848,3,122,22
Tue Oct 18 01:45:13 GMT 2011,6,43099880,3,122,8
Tue Oct 18 01:47:13 GMT 2011,6,43099880,3,122,0
Tue Oct 18 01:47:23 GMT 2011,6,44162904,3,122,28
Tue Oct 18 01:47:33 GMT 2011,6,46233864,3,122,6
Tue Oct 18 01:47:43 GMT 2011,6,46233864,3,122,0
Tue Oct 18 01:48:53 GMT 2011,6,46233864,3,122,0
```

4.7 Commands

Through the **System → Commands** menu, Figure 4-99, an administrator can issue several system commands to WebSphere Cast Iron. The list of commands can vary, depending on the Integration Appliance configuration (stand-alone versus high-availability pairs) and the Integration Appliance state (active or standby).

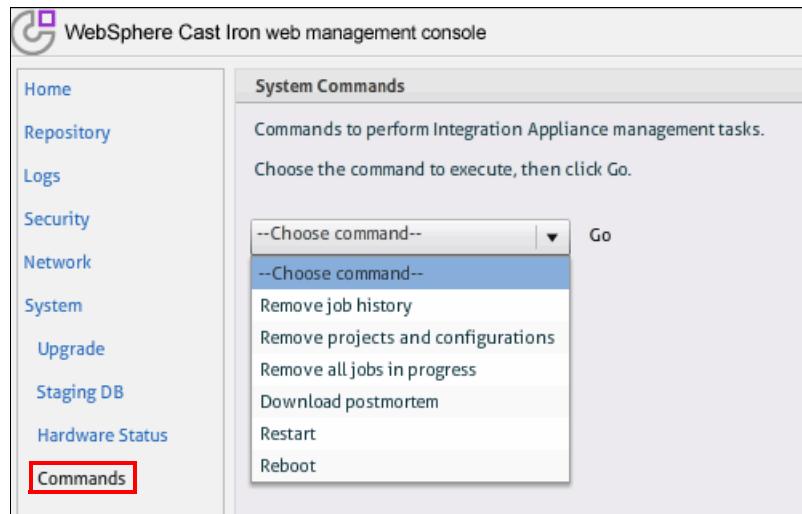


Figure 4-99 Commands menu

Cast Iron Live: This functionality is not available with Cast Iron Live.

Table 4-4 lists a description of these commands.

Table 4-4 System commands

Command	Description
Remove job history	Removes orchestration monitoring data.
Remove project and configurations	Removes currently deployed project or projects data.
Remove all jobs in progress	Stops and removes all currently running project data.
Download postmortem	Generates an archive of information that is gathered from Integration Appliance logs, orchestration logs, and transaction stores. This information can be used to debug Integration Appliance issues.
Restart	Restarts the Integration Appliance routing subsystem (the runtime engine).
Reboot	Shuts down all routing services and reboots the Integration Appliance.

To run a command, select the command and then click **Go**. Figure 4-100 shows a reboot command in progress.



Figure 4-100 Reboot command in progress

4.8 Security

The WMC contains a Security menu, as shown in Figure 4-101. This menu offers several tasks to manage the appliance from the security perspective. You can use this menu to manage users and groups, authentication servers (LDAP and Kerberos), and also certificates.



Figure 4-101 Security menu

Notice that the WebSphere Cast Live security menu contains only Users and Groups and certificates. There is a dedicated menu to manage Secure Connectors, as shown in Figure 4-102.



Figure 4-102 System menu

You can find more details about Security in Chapter 5, “Security” on page 231.

4.9 Appliance and connector upgrades

When an upgrade is released for the Integration Appliance, you can download the update from IBM Fix Central at:

<http://www.ibm.com/support/fixcentral>

When you download an update, be sure to make the following selections:

1. For product group, select **WebSphere**.
2. For Product select, **WebSphere Cast Iron Cloud Integration**.
3. For Installed Version, select the version that you currently have.

You are then presented with a selection of downloads. Choose the correct file for your Integration Appliance, and then apply it. For the physical appliance, there is a .scrypt2 file and, for the hypervisor, there is a .vcrypt2 file.

This section explains how to upgrade the hypervisor (virtual appliance). The process is identical for the physical appliance using the .scrypt2 file.

If you have an HA configuration, upgrade only the primary Integration Appliance.

Important: When upgrading the WMC, the system logs are not persisted. To retain system log information, export system logs before you upgrade.

4.9.1 Verifying the current version

To verify the current WebSphere Cast Iron version, select **System → Upgrade** to load the System Summary pane shown in Figure 4-103, which contains detailed information.

System Summary	
Model	Cast Iron vA3000, Revision A
Version	Cast Iron Operating System 6.1.0.1.0000 (Thu Sep 15 14:16:52 UTC 2011)
Serial Number	VMWHWBWWYLOK4DV6P

Figure 4-103 System Summary pane

With WebSphere Cast Iron Cloud, this information is accessible through the Settings menu, as shown in Figure 4-104.



Support Information	
Version	6.0.0.3_12494_437
gdinizdemoraишwsbp (Tenant)	B235D2D0C65EC0129F64DDA736E14E1
Production (Environment)	B34465BE08FEE8F69D13B5495F2E451D

Figure 4-104 Version information for Cast Iron Live

4.9.2 Upgrading the Integration Appliance

To upgrade the Cast Iron Hypervisor Edition:

1. Download the .vcrypt2 file, as shown in Figure 4-105.

<input type="checkbox"/> 1. fixpack: 6.1.0.1-WS-WCI-20110916-1648_H2.exe 6.1.0.1-WS-WCI-20110916-1648_H2.exe	19-Sept-2011
<input checked="" type="checkbox"/> 2. fixpack: 6.1.0.1-WS-WCI-20110915-1504_H2.vcrypt2 6.1.0.1-WS-WCI-20110915-1504_H2.vcrypt2	19-Sept-2011
<input type="checkbox"/> 3. interim fix: 6.1.0.1-WS-WCI-20110915-1504_H2.tar 6.1.0.1-WS-WCI-20110915-1504_H2.tar	19-Sept-2011
<input type="checkbox"/> 4. fixpack: 6.1.0.1-WS-WCI-20110915-1504_H2.scrypt2 6.1.0.1-WS-WCI-20110915-1504_H2.scrypt2	19-Sept-2011
<input type="checkbox"/> 5. fixpack: 6.1.0.1-WS-WCI-20110915-1504_H2.ova 6.1.0.1-WS-WCI-20110915-1504_H2.ova	19-Sept-2011

Figure 4-105 Downloading the .vcrypt2 file

2. Log in as an administrator to your Integration Appliance.
3. Click **System**, and then click **Upgrade** if it is not already selected.
4. Click **Upgrade Operating System**, as highlighted in Figure 4-106.

Home	System Summary	
Repository	Model	Cast Iron vA3000, Revision A
Logs	Version	Cast Iron Operating System 6.1.0.0000 (Mon Aug 22 11:38:16 UTC 2011)
Security	Serial Number	VMWQETOM4E65RFFI
Network	Upgrade Operating System Update Connector Libraries	
System		
Upgrade		
Staging DB		
Hardware Status		
Commands		

Figure 4-106 Upgrade page

5. Click **Browse**, and select the .vcrypt2 file that you downloaded. Click **Upgrade**, as illustrated in Figure 4-107.

Upgrade Operating System

Select upgrade file.
After the Operating System is upgraded, the appliance will reboot

Upgrade File: [Browse](#)

[Upgrade](#) [Cancel](#)

Figure 4-107 Upgrading Operating System page

6. The Integration Appliance completes the following tasks:

- Upload File
- Upgrade Operating System
- Reboot

When these tasks complete, which can take several minutes, the WMC login window opens, as shown in Figure 4-108.

Tip: Some browsers are restrictive in supporting self-signed certificates and might block browser functions, such as file upload, which is required for installing module providers and uploading projects. If you have this problem, either install a trusted certificate before starting the upload or use a different browser.

7. Enter the user name and password to log in to the WMC.

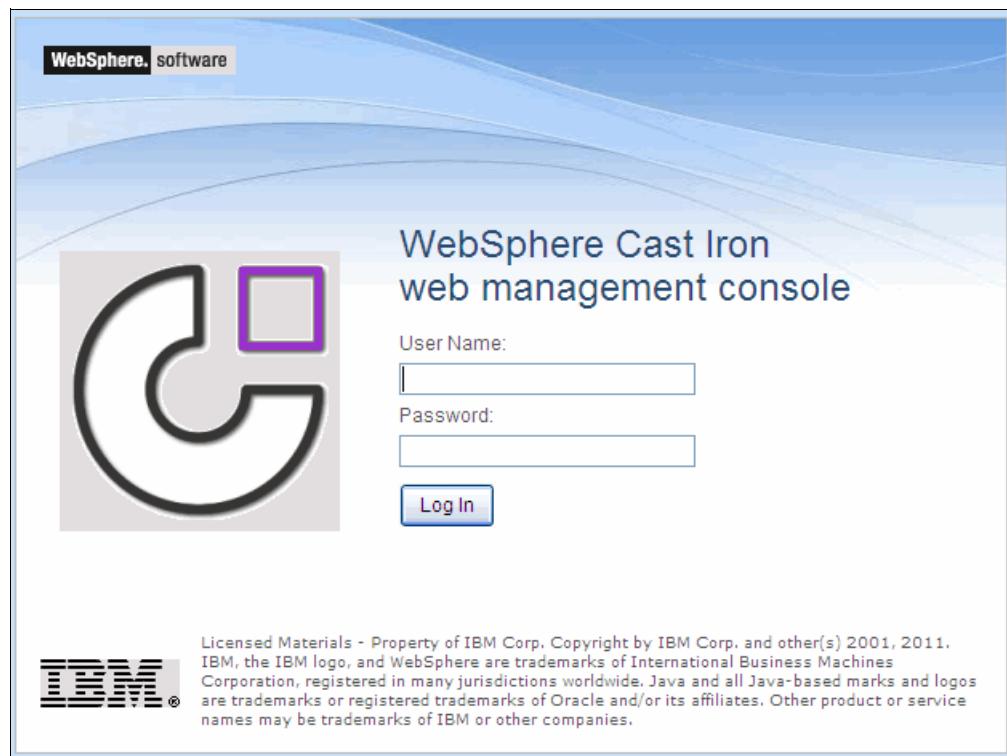


Figure 4-108 Login page after the upgrade

After the upgrade, it is recommended that you check the version number in the System Summary.

Cast Iron Live: Maintenance for Cast Iron Live is performed by IBM.

4.9.3 Update connector libraries

If your project uses one or more connectors that require additional libraries, you must install the additional libraries before you deploy the project configuration. Deploying the project configuration before installing the additional libraries generates runtime errors. For a

complete list of connectors that require third-party libraries, open the **System → Upgrade → Update Connector Libraries** page, as shown in Figure 4-109.

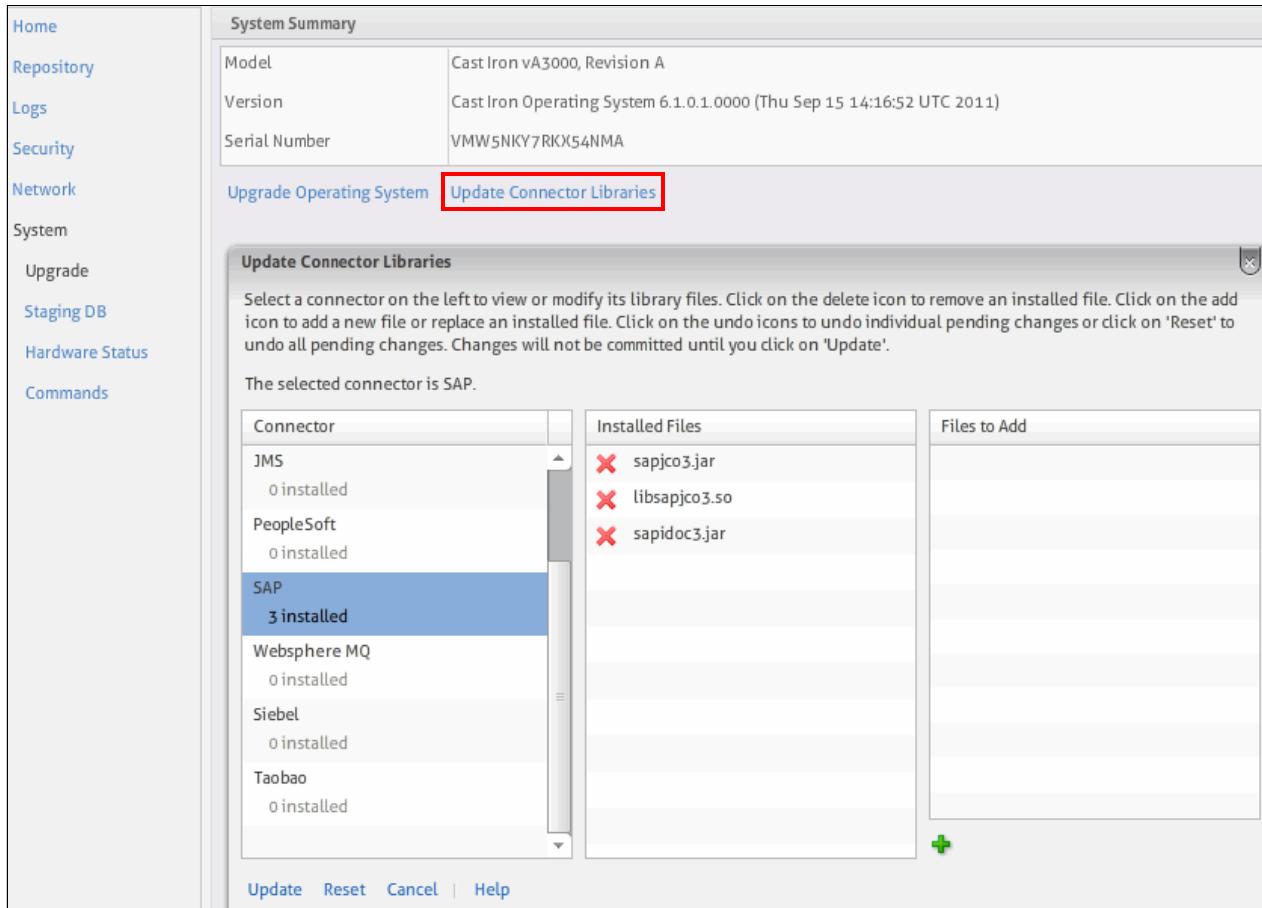


Figure 4-109 Updating connector libraries

Cast Iron Live: In Cast Iron Live, you can open this window by clicking **System → Connector Libraries**.

Detailed steps to update an SAP library in your system are described in 2.6.5, “Installing third-party libraries” on page 73. For each connector, a different file set is provided, but the process to update connector libraries does not change.

For more details about connector libraries, go to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.appliance.doc/Managing_Integration_Appliances/installing3rdPartyLibraries.htm



Security

This chapter provides an overview of the security features that are included in IBM WebSphere Cast Iron. It shows how to address the main security aspects at various levels for all form factors and discusses the security configuration steps that are required when a project is assembled.

This chapter is not intended to be a guide to securing appliances, applications, or networks. That information is beyond the scope of this book. This chapter is intended to provide security specialists with information about the capabilities of IBM WebSphere Cast Iron Cloud Integration. This information can be factored into the larger scope of corporate security procedures.

This chapter contains the following sections:

- ▶ WebSphere Cast Iron users and groups
- ▶ Granting project configuration permissions
- ▶ Permissions to publish projects from Studio
- ▶ Configuring WebSphere Cast Iron to use LDAP authentication
- ▶ Working with certificates
- ▶ Securing communication with endpoints
- ▶ Monitoring security
- ▶ Additional security considerations

5.1 WebSphere Cast Iron users and groups

Users that access WebSphere Cast Iron to publish or manage projects must log in to the runtime with a user ID and password. A *user* is a person with permission to perform a task in the WebSphere Cast Iron runtime. Users permissions are defined by the group of which they are a member.

WebSphere Cast Iron provides the following built-in groups for users that define specific levels of privileges in the runtime:

- ▶ admin
- ▶ publisher
- ▶ user

A user must be a member of one of these groups.

WebSphere Cast Iron Live: WebSphere Cast Iron Live includes the additional roles of the tenant administrator and the environment administrator.

Users are assigned to a built-in group and can also be assigned to a custom group that you create. Custom groups provide several users access to a project. They do not define additional permissions to the three built-in groups.

Initial admin user: WebSphere Cast Iron has a default administrative user called admin with a password of !n0r1t5@C. This user is a member of the admin group and can be used to manage users and groups that are defined to WebSphere Cast Iron.

Table 5-1 shows the built-in groups and the permissions a user has as a member of the group.

Table 5-1 List of permissions by group

Permissions	Groups		
	admin	publisher	user
▶ Create, edit, and delete users ▶ Create, edit, and delete custom groups ▶ Add users to any group ▶ Remove users from any group ▶ Create and deploy project configurations for projects that any publisher publishes ▶ View orchestration job details for any project configuration ▶ Edit permissions that are set for any project configuration	Allowed	Not allowed	Not allowed
▶ Create, deploy, or delete project configurations for any project that you publish ▶ Start and stop orchestrations for project configurations you deployed ▶ Grant permissions to other users for project configurations you created	Allowed	Allowed	Not allowed
▶ Monitor alerts and orchestrations ▶ Create and edit projects	Allowed	Allowed	Allowed

5.1.1 Viewing and updating your user profile

User *profiles* contain information about each user. To view your user profile, log in to the WMC, and select Settings, as shown in Figure 5-1.



Figure 5-1 The Settings link in the WMC

The resulting list displays your user name, display name, email, and group memberships. You can also change your display name, password, or email. For example, Figure 5-2 shows the details for user2.

Settings		
▼ User Profile		
User Name	user2	
Display Name	user2	Edit
Password	*****	Edit
Email	user2@ibm.com	Edit
User Groups	user	

Figure 5-2 User profile in the appliance

To update the display name, password, or email, click **Edit**.

5.1.2 Creating a new user

In the appliance, only the admin user and users with administrator privileges can create a new user. To create a new user:

1. In the WMC, click **Security** to open the pane shown in Figure 5-3 on page 234. By default, the Users and Groups section opens.

The screenshot shows the 'WebSphere Cast Iron web management console' interface. On the left, a sidebar lists navigation options: Home, Repository, Logs, Security (which is highlighted with a red box), Users and Groups, Certificates, LDAP, Kerberos, Network, and System. The main content area has two tabs: 'Users' and 'Groups'. The 'Users' tab is active, displaying a table with columns: Name, Display Name, Status, Email, and Groups. One row is present for 'admin'. Below the table are 'New User' and 'Delete' buttons. The 'Groups' tab is also visible, showing a table with columns: Name, Display Name, and Users. It lists three groups: 'admin' (Administrator Group), 'publisher' (Publisher Group), and 'user' (User Group). Below the table are 'New Group' and 'Delete' buttons. A 'Refresh' button is located at the bottom of the users section. At the very bottom of the page, a status message reads 'Last refreshed: 10/18/2011 02:33:13 PM GMT-0200'.

Figure 5-3 Users and Groups section

2. To create a user, click **New User**, as shown in Figure 5-4.

This screenshot shows a close-up of the 'Users' table from Figure 5-3. The 'New User' button at the bottom left of the table is highlighted with a red box. The table itself contains one row for 'admin'.

Figure 5-4 Creating a new user

The New User window opens, as shown in Figure 5-5. Complete the following fields:

Name	The user name that is used for log in
Display Name	A name to associate with the user
Email	An email address for the user
Password	A password that conforms to the specified rules
Confirm Password	Configuration of the password

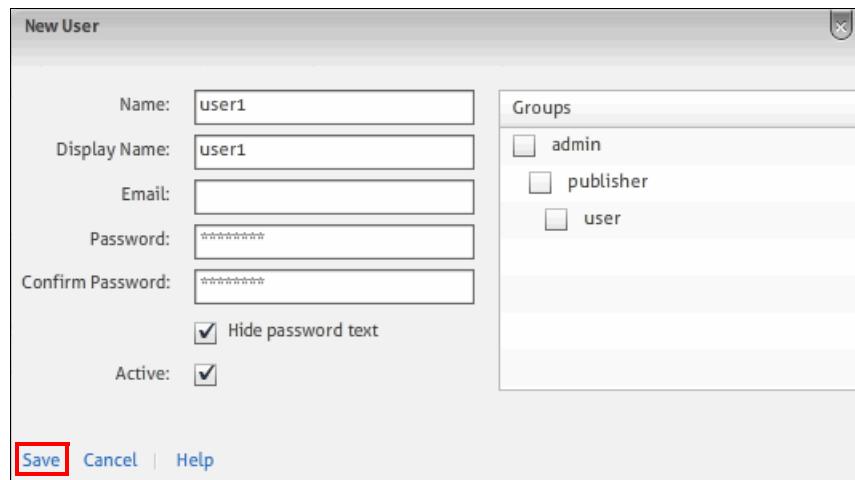


Figure 5-5 Creating user1

Password rules: The password must contain characters from three of the following four categories:

- ▶ Uppercase letters
- ▶ Lowercase letters
- ▶ Numeric characters
- ▶ Punctuation (for example !, \$, #, or %)

Add the tenant name to the user name and display name, for example, if the tenant name is ibm.com, the new user name is newuser@ibm.com. This naming convention can help you to identify common user names across multiple tenants in the cloud.

Clearing the Hide password text field shows the password in clear text.

Clearing the Active option prevents the user from being able to log in to the WMC.

You can select a group for the user in the right column.

Group selection: A new user is placed in one of the built-in groups regardless of whether you select a group as follows:

- ▶ If you select no groups, the user is added to the user group.
- ▶ If you select a group other than the user group, the user is added to that group and to the user group.
- ▶ If you select a built-in group higher in the hierarchy than the user group, for example the publisher group, the new user is added only to that group.

If you do not select a group, the user is placed in the user group by default.

WebSphere Cast Iron Live: The WebSphere Cast Iron Live environment includes a field called Libraries Environment. This field allows you to select the environment from which the user's projects can retrieve additional libraries and files for endpoints. The environment that you select must match the environment of the primary group of which the user is a member.

3. Click **Save**.

The new user is added to the list of users, as shown in Figure 5-6. This user is placed in the user group by default because you did not specify a group.

	Name	Display Name	Status	Email	Groups
<input type="checkbox"/>	admin	Administrator	Active		admin
<input type="checkbox"/>	user1	user1	Active		user
<input type="checkbox"/>	user2	user2	Active		user

Figure 5-6 Adding the new user

5.1.3 Logging in with a new user ID

When you log in to the WMC with a new user ID, you are prompted to read and accept the Terms of Use and Software License Agreement to access the main console. The license agreement spans three panels, and you must navigate through these panels as you read the agreement. After you read the text in a panel, click **Accept** to continue to the next panel. Figure 5-7 shows the first panel.

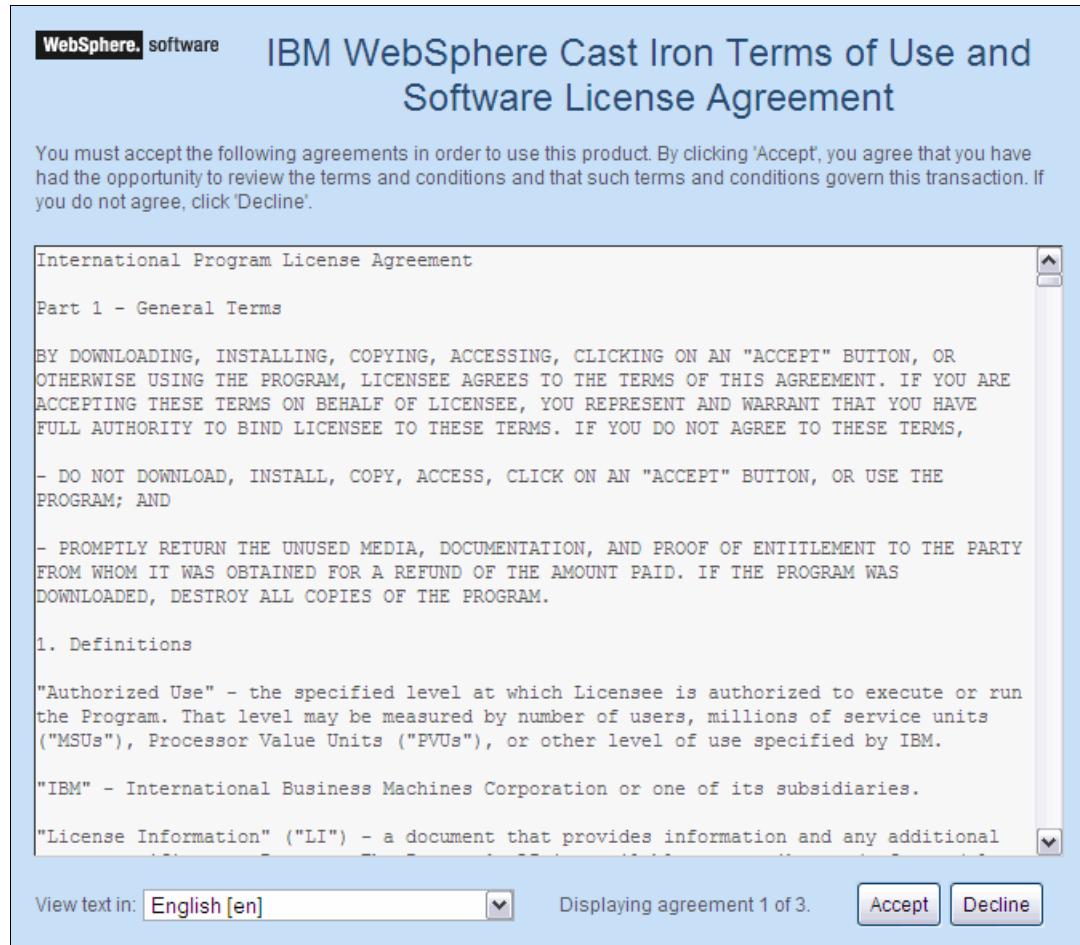


Figure 5-7 Terms of Use and Software License Agreement: Panel 1 of 3

5.1.4 Changing your password

On the appliance, only you and the administrator can change your password. In Cast Iron Live, only you and the tenant administrator can change your password.

To change your password:

1. Log in to the WMC.
2. Click the **Settings** link at the top-right corner in the WMC to access your user profile.

3. In the Settings pane, click **Edit** to the right of the Password field, as shown in Figure 5-8, to change your password.



Figure 5-8 Settings pane to change the password

4. Enter your old password. Enter and confirm your new password, as shown in Figure 5-9. Click **Save**.

The screenshot shows a 'Password' pane with three input fields: 'Old password' (containing '*****'), 'New password' (containing '*****'), and 'Confirm password' (containing '*****'). Below these fields is a checked checkbox labeled 'Hide password text'.

Figure 5-9 Password pane with passwords

Password rules

When you create or change a password, the password must meet the following requirements:

- A new password cannot contain parts of an old password.
- The password must contain characters from three of the following four categories:
 - Uppercase letters
 - Lowercase letters
 - Numeric characters
 - Punctuation (for example !, \$, #, or %)

Changing the password of another user

If you have administrator privileges, you can change the password of any user. To change the password for another user:

1. Log in to the WMC as a user with administrator privileges, for example admin.
2. Click **Security** to open the Users and Groups section.

3. In the Users section, click the user to edit it, as shown in Figure 5-10.

	Name	Display Name	Status	Email	Groups
<input type="checkbox"/>	admin	Administrator	Active		admin
<input type="checkbox"/>	user1	user1	Active		user
<input type="checkbox"/>	user2	user2	Active		user

New User Delete

Figure 5-10 User2 details

4. The Edit User window opens with the user profile information. Enter and confirm the password, as shown in Figure 5-11.

Edit User

Name: user2
Display Name: user2
Email:
Password: *****
Confirm Password: *****
 Hide password text
Active:

Groups

admin
 publisher
 user

Save Cancel | Help

Figure 5-11 Changing the user2 password

5. Click **Save**. A message displays that the save was successful.

Using the Command Line Interface to change a password: You can use the Command Line Interface (CLI) to change the password of a user. You do not need to know the current password for user2, but you need the admin password. The passwords that you type do not display on the console.

For example, to change the password for user2, use the following command from the CLI:

```
ci-9-42-171-225/Standalone> auth set user user2
```

Enter user2's old password or admin's password:

Enter new password:

Re-enter new password:

Successfully changed password

When Lightweight Directory Access Protocol (LDAP) is used for authorization and authentication for users and groups, this command can be used only for the administration account.

5.1.5 Creating a new group

You can create a new custom group to tailor permissions for a group of users. Only the admin user or others with administration privileges can create custom groups.

To create a custom group:

1. In the left menu of the WMC, click **Security** to open the Users and Groups section.
2. In the Groups section pane, click **New Group**, as shown in Figure 5-12.

	Name	Display Name	Users
	admin	Administrator Group	admin
	publisher	Publisher Group	
	user	User Group	user1,user2

Figure 5-12 Create new group

3. In the New Group window, enter the name for the group and a display name, as shown in Figure 5-13. You can also select the users to add to the group now by selecting each user from the list on the right.

New Group

Name:

Display Name:

User

admin
 user1
 user2

Save Cancel | Help

Figure 5-13 Creating the itso1 group

- Click **Save**. The new group is listed in the Security Groups section, as shown in Figure 5-14.

Groups			
	Name	Display Name	Users
<input type="checkbox"/>	admin	Administrator Group	admin
<input type="checkbox"/>	its01	ITSO Group 1	
<input type="checkbox"/>	publisher	Publisher Group	
<input type="checkbox"/>	user	User Group	user1,user2

New Group Delete

Figure 5-14 Groups section pane listing the its01 group

5.1.6 Assigning users to groups

You can assign users to groups using one of the following methods:

- Add an existing user to a new custom group when you create it.
- Add a new user to an existing group when you create the user.
- Update an existing group with an existing user.

Previous sections explained how to create users and groups and how to add a user to a group. This section explains how an administrator can add existing users to existing groups:

- In the left menu of the WMC, click **Security** to open the Users and Groups section.
- In the Groups section, click the group to which you want to add users. A window opens that allows you to edit the group. Select the users to add to the group, as shown in Figure 5-15, and then click **Save**.

Edit Group

Name:	<input type="text" value="its01"/>	<input checked="" type="checkbox"/> User
Display Name:	<input type="text" value="ITSO Group 1"/>	<input checked="" type="checkbox"/> admin
		<input checked="" type="checkbox"/> user1
		<input checked="" type="checkbox"/> user2

Save Cancel | Help

Figure 5-15 Updating the its01 group with the user1 and user2 users

3. A message displays that the group saved successfully. Now the Groups section shows the group updated with the users added, as shown in Figure 5-16.

Groups		
	Name	Display Name
	Users	
	admin	Administrator Group
	its01	ITSO Group 1
	publisher	Publisher Group
	user	User Group
		user1,user2

Figure 5-16 The its01 group updated with the user1 and user2 users

5.1.7 Deleting a user

In the appliance, only the admin or users with administrator privileges can delete an user.

To delete a user:

1. In the WMC, click **Security** to open the Users and Groups section.
2. In the Users section, select the user that you want to delete, and then click **Delete** as shown in Figure 5-17.

Users					
	Name	Display Name	Status	Email	Groups
	admin	Administrator	Active		admin
	publisher1	publisher1	Active		publisher
	user1	user1	Active		its01,user
	user2	user2	Active		its01,its02,user
<input checked="" type="checkbox"/>	user3	user3	Active		its02,user

Figure 5-17 Deleting user3

3. Confirm that you want to delete the user by clicking **Yes**, as shown in Figure 5-18.

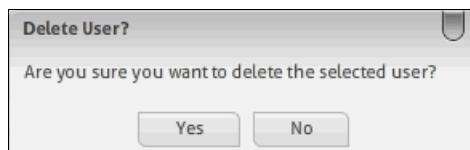


Figure 5-18 Confirmation to delete a user

A message displays indicating that the user was deleted successfully.

Deleting a user:

- ▶ Notice that deleting a user does not affect any groups. Deleting the user removes the user from any group membership.
- ▶ You cannot delete the admin user.

5.1.8 Deleting a group

You cannot delete the built-in groups. Only the admin or users with administrator privileges can delete a custom group. To remove a group:

1. Click **Security** to open the Users and Groups section.
2. In the Groups section, select the group that you want to delete, and click **Delete**, as shown in Figure 5-19.

Groups			
	Name	Display Name	Users
<input type="checkbox"/>	admin	Administrator Group	admin
<input type="checkbox"/>	itso1	ITSO Group 1	user1,user2
<input checked="" type="checkbox"/>	itso2	ITSO Group 2	user2
<input type="checkbox"/>	publisher	Publisher Group	publisher1
<input type="checkbox"/>	user	User Group	user1,user2

New Group **Delete**

Figure 5-19 Deleting group itso2

3. Confirm that you want to delete the group by clicking **Yes**, as shown in Figure 5-20.

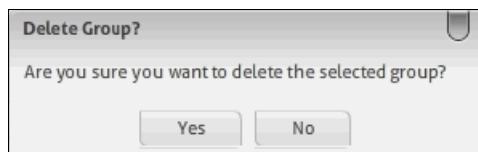


Figure 5-20 Confirmation to delete a group

A message displays indicating that the group deleted successfully.

Users in deleted groups: Deleting a custom group does not delete the users in the group. Users can belong to multiple groups and always belong to one built-in group. If you delete a custom group, the users still have membership in the built-in group.

5.2 Granting project configuration permissions

By default, permissions to project configurations are granted only to the publisher of the project and to the admin group. If you want to grant permissions to another group you must explicitly do so.

Granting a group permission to a project configuration in the WebSphere Cast Iron runtime gives the users in that group full access to the project configuration. They can edit the project configuration, delete it, or create new project configurations.

You must have administrator privileges or be the publisher of a project to grant permissions to a project configuration.

To assign permissions for a project:

1. Click a project configuration. The Configuration Details page displays. In the Permissions section, click **Edit**, as shown in Figure 5-21.

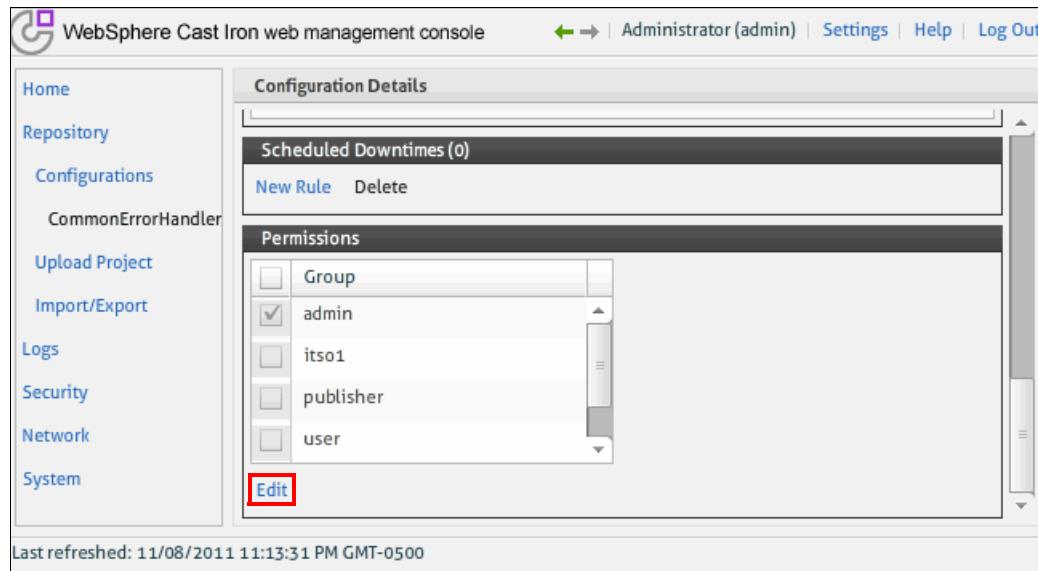


Figure 5-21 Permissions section in the Configuration Details panel

2. Select the users or groups to which you want to grant permissions, as shown in Figure 5-22.

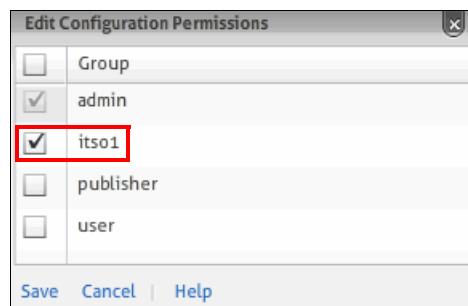


Figure 5-22 Select the groups to which to give permission

3. Click **Save**.

5.3 Permissions to publish projects from Studio

Only users with publisher or admin privileges can publish projects into the appliance run time or WebSphere Cast Iron Live. Other users can only create and edit projects and cannot publish any project.

For example, user1 attempts to publish the SFDC_to_DB2_upsert_Products project using his credentials, as shown in Figure 5-23.

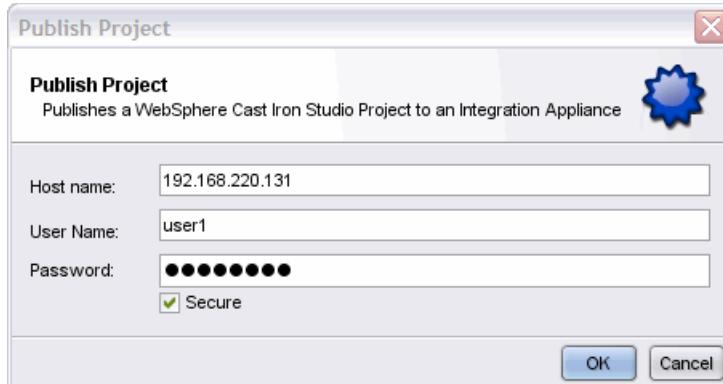


Figure 5-23 Publishing project with user1 credentials

As soon user1 clicks **OK**, an authentication failure message displays, as shown in Figure 5-24.

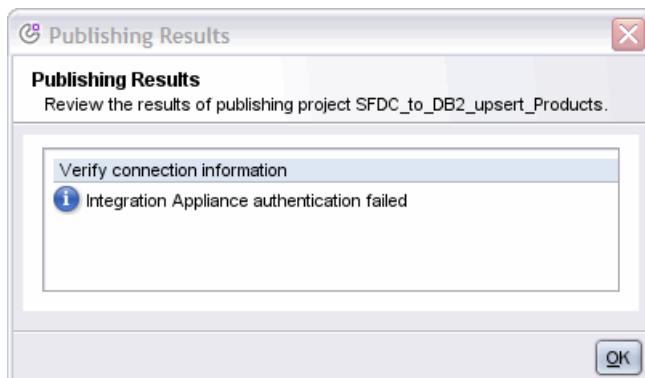


Figure 5-24 Authentication failure for user1

However, if publisher1 credentials are used, as shown in Figure 5-25, the project will publish successfully.



Figure 5-25 Publishing project with publisher1 credentials

Figure 5-26 shows the progress of the project as it publishes.

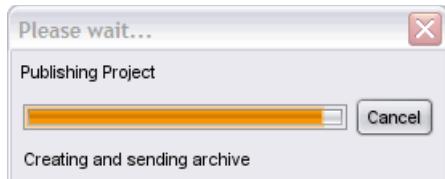


Figure 5-26 Publishing project

If the project publishes successfully, a message displays, as shown in Figure 5-27. Click **OK** to continue.



Figure 5-27 Project successfully published

5.4 Configuring WebSphere Cast Iron to use LDAP authentication

The Integration Appliance provides an internal user repository that can be used to store and manage user and access information. Although this repository might be sufficient in some cases, a more robust security solution is to use a remote LDAP repository to manage information. Typical implementations of LDAP repositories allow you to store a wealth of information about users and their access rights. WebSphere Cast Iron can access an LDAP repository for the authentication of users and authorization of users to resources.

You might consider using an LDAP repository for the following reasons:

- ▶ You want to use your existing user repository instead of setting up a new one.
- ▶ You have complex organizational structures that must be reflected in the repository.
- ▶ You have multiple WebSphere Cast Iron appliances and want to have one central repository.

(If you have just two physical appliances in a high availability (HA) configuration, synchronization between these two appliances occurs automatically.)

- ▶ You have complex password rules that cannot be fulfilled by WebSphere Cast Iron.
- ▶ You have complex monitoring requirements for authorization.

The admin user: When using an LDAP implementation, the admin user is still defined locally. If your access to the LDAP server fails, you can still log in to the Integration Appliance.

If users other than the admin user are defined in both the local user repository and the LDAP repository, the LDAP information overrides the local information.

Important: LDAP users cannot access the staging database.

Configuring WebSphere Cast Iron to use LDAP requires an understanding of LDAP and of the structure of your LDAP repository. These steps show only the options that are available.

To configure an LDAP repository to the Integration Appliance:

1. Log in to the WMC console as admin.
2. In the navigation pane, select **Security** → **LDAP**, as shown in Figure 5-28.

The screenshot shows the 'LDAP Configuration' page in the WebSphere Cast Iron web management console. The left sidebar has a 'LDAP' icon with a cursor arrow pointing to it. The main area contains several configuration sections:

- Host Information:** Fields for 'Host' (text input), 'Port' (text input), and 'Base DN' (text input with a 'Fetch DNs' button).
- Authentication:** A dropdown menu for 'Mechanism' with a 'Fetch Supported' button, and a link to 'Advanced Settings >>'.
- Security Options:** Radio buttons for 'Use Secure Connection (SASL)' (selected), 'Use Simple Authentication', and 'Anonymous Bind'.
- Credentials:** Fields for 'Principal' (text input) and 'Password' (text input with a 'Hide password text' checkbox checked).
 - User Authentication:** Sub-section with fields for 'User Container RDN', 'User Search Filter', and 'User Principal DN'.
 - Group Mapping:** Sub-section with fields for 'Admin Group Filter', 'Publisher Group Filter', and 'User Group Filter'.
 - Authentication Mode:** Radio buttons for 'Composed DN Mode' (selected) and 'Searched DN Mode'.

Figure 5-28 LDAP configuration settings

3. Select the **Enable LDAP** option.
4. In the Host Information section, complete the following information:
 - a. Enter the host and port information for the LDAP server.
 - b. Click **Fetch DNs** to select the base Distinguished Name (Base DN) of your company.
5. In the Credentials section, enter the user ID of an LDAP admin user or a user who can run directory searches (if the LDAP server requires it), as shown in Figure 5-29.

The screenshot shows the 'Credentials' section of the LDAP configuration page. It includes fields for 'Principal' (containing 'cn=its0,dc=its0,dc=com') and 'Password' (containing '*****'). A 'Hide password text' checkbox is checked.

Figure 5-29 LDAP credentials

6. In the Security Options section, choose one of the following options:
 - Use Secure Connection (SASL)

Enables Simple Authentication and Security (SASL) mechanisms that are supported by the LDAP directory server. This option is the most secure option.
 - Use Simple Authentication

Enables simple authentication, but does not encrypt credentials.
 - Anonymous Bind

Enables anonymous binding to the LDAP directory server. This option is the least secure option.
- If you select SASL, configure the Authentication section.
7. In the Authentication Mode section, select the mode:
 - Composed DN Mode
 - Search DN Mode
 8. In the User Information section, complete the following fields:
 - User Container RDN®

Configure this field when the composed DN mode is selected, which means you can specify the immediate ancestor of the users. For example, if User DN = uid=itsouser,ou=itso,dc=xyz,dc=com, the User Container RDN is ou=itso because dc=xyz,dc=com is the Base DN. This is typically left blank when the search DN mode is selected.
 - User Search Filter

This field is required only if the search DN mode is selected. This entry accepts an LDAP search filter specification and also accepts a user name in the form of \$USERNAME. Figure 5-30 illustrates this configuration.

User Authentication	
User Information	
User Container RDN:	<input type="text"/>
User Search Filter:	<input type="text"/> cn=\$USERNAME
User Principal DN:	<input type="text"/> \$DN

Figure 5-30 LDAP user information

- User Principal DN

This field specifies the DN of the user that the LDAP directory server binds after the search. You can specify \$USERNAME or \$DN to represent the user name and Distinguished Name. Where \$DN represents (RDN+BaseDN) for the Compose DN Mode and searched DN for Search DN Mode.

- Specify in the Group Mapping section the filters for the three roles (administrators, publishers, and users). These are standard LDAP filters, for example, many LDAP servers use the `isMemberOf` field to indicate to which group the user belongs, as shown in Figure 5-31.

Group Mapping	
Admin Group Filter:	(isMemberOf=cn=admins,dc=itso,dc=com)
Publisher Group Filter:	(isMemberOf=cn=publishers,dc=itso,dc=com)
User Group Filter:	

Figure 5-31 LDAP group mapping

Figure 5-32 shows a configuration of WebSphere Cast Iron that is connected to an LDAP server that requires directory search authentication. Searching for users is done with search DN mode, which means that the LDAP search starts from the root of the directory information trees (DITs).

LDAP Configuration	
<input checked="" type="checkbox"/> Enable LDAP	
General	
Host Information Host: 192.168.47.1	
Port: 389	
Base DN: dc=itso,dc=com	
Security Options <input type="radio"/> Use Secure Connection (SASL)	
<input checked="" type="radio"/> Use Simple Authentication	
<input type="radio"/> Anonymous Bind	
Authentication	
Mechanism:	<input type="text"/> Fetch Supported
Advanced Settings >>	
Credentials Principal: cn=itso,dc=itso,dc=com	
Password: *****	
<input checked="" type="checkbox"/> Hide password text	
User Authentication	
User Information User Container RDN:	
User Search Filter: cn=\$USERNAME	
User Principal DN: \$DN	
Authentication Mode <input type="radio"/> Composed DN Mode	
<input checked="" type="radio"/> Searched DN Mode	
Group Mapping	
Admin Group Filter: (isMemberOf=cn=admins,dc=itso,dc=com)	
Publisher Group Filter: (isMemberOf=cn=publishers,dc=itso,dc=com)	
User Group Filter:	

Figure 5-32 LDAP configuration pane

The user group filter is the first filter that is evaluated. Then, the user is checked for publisher or administrator privileges.

The user group filter determines who gets to log in to the WMC. This filter is the first of the three group filters that are evaluated for WMC login. With the user group filter, the following cases occur:

- Case 1: If the group filter is blank, any user who inputs valid credentials is allowed to log in, meaning that this filter is satisfied for all users.

- ▶ Case 2: If it is non-blank, only the users satisfying this filter are allowed to log in. If this filter fails, the user is denied WMC access, even if valid login credentials are provided.

The other two filters are used for granting privileges and are consulted only if the user group filter is satisfied; otherwise, they are ignored:

- ▶ The publisher group filter determines whether the user is granted publisher privileges, as follows:
 - Case 1: If blank, publisher privileges are never granted.
 - Case 2: If non-blank, users satisfying this filter are granted publisher privileges.
- ▶ The admin group filter determines whether the user is granted admin privileges:
 - Case 1: If blank, admin privileges are never granted.
 - Case 2: If non-blank, users satisfying this filter are granted admin privileges.

For further information, visit:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.appliance.doc/Security/enablingLDAP.htm

LDAP configuration can also be performed using the Management API interface, as described at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.api.doc/ci00060.html

5.5 Working with certificates

Digital certificates establish secure communication between two endpoints. A certificate contains information that includes the following important items:

- ▶ An alias name for the certificate
- ▶ To whom the certificate is issued
- ▶ Who issued the certificate
- ▶ Valid from and valid to date

The contents of a certificate are signed by a CA, which is a trusted third-party organization or company that issues the digital certificates. The certificate authority typically verifies the identity of the individuals who are granted the unique certificate. Certificates can also be self-signed certificates for use with a peer in a controlled environment.

In the context of WebSphere Cast Iron, certificates secure an inbound or outbound connection. In the WMC, go to **Security** → **Certificates** to view the certificates that are used by WebSphere Cast Iron. Figure 5-33 shows several certificates.

The screenshot displays the 'Certificates' interface in the WebSphere Cast Iron Integration Appliance. It is divided into three main sections:

- Key Store (1)**: Contains one certificate named "factory supplied identity". The table includes columns for Alias, Issued To, Issued By, From Date, and To Date. The certificate details are: Alias: factory supplied identity, Issued To: C=US,ST=CA,L=Mountain View, Issued By: C=US,ST=CA,L=Mountain View, From Date: 09/27/2011, To Date: 09/26/2013.
- Trust Store (78)**: Contains three certificates: "addtrustclass1ca", "addtrustexternalca", and "addtrustclass1ca". The table includes columns for Alias, Issued To, Issued By, From Date, and To Date.
- Settings**: Shows SSL Usage Type (Client SSL, Server SSL over data NIC, Server SSL over mgmt NIC) and their corresponding Certificate Aliases (factory supplied identity). It also indicates VPeer and VHost settings and Cipher Strength (Strong).

Figure 5-33 Certificates in the WebSphere Cast Iron Integration Appliance

The certificates are stored in a secure store. WebSphere Cast Iron includes the following stores:

- The *Key Store* contains the certificates that WebSphere Cast Iron uses to authenticate itself.
There is currently only one certificate in the Key Store, a certificate with the name *Factory Supplied Identity*. This certificate is created during the first start of the Integration Appliance and valid for two years. You can have more than one certificate in the Key Store.
- The *Trust Store* contains the certificates that are used to verify trust.
The certificates are created mainly by Trusted CA and used to verify that a retrieved certificate is valid. The certificates are often called signer certificates because these certificates are signed by a CA. You can add your own certificates to the Trust Store.

You can specify the certificate to use in the following situations:

- For inbound SSL communication on the management network, for example when you access the WMC.
- For inbound SSL communication on the data network, for example, when you call an orchestration with an HTTP Receive Request using HTTPS.
- For outbound SSL communication on the data network when the run time calls an external service on the data network and provides a client certificate.

This scenario is also called *mutual SSL* because the server and the client must authenticate each other.

Use cases where certificates are used are discussed in more detail in 5.6, “Securing communication with endpoints” on page 262.

5.5.1 Managing certificates

You can import, delete, and generate certificates using the WMC. You might need to manage certificates for the following reasons:

- ▶ You want to replace the default certificates with your own certificates.
- ▶ You want to add new signer certificates to the trust store.
- ▶ Certificates expired.
- ▶ You want to use different certificates for different endpoints.

Adding a new signer certificate

If you receive a new signer certificate from a CA, import it into the Trust Store so that outbound SSL connections can use it. Follow these steps:

1. In the WMC, go to **Security → Certificates**, and in the Trust Store section click **Import**, as shown in Figure 5-34.

Key Store (1)		
<input type="checkbox"/>	Alias	Issued To
<input type="checkbox"/>	factory supplied identity	C=US,ST=CA,L=Mountain View,O=Cast Iron Systems,OU=Cast Iron Systems

Trust Store (76)		
<input type="checkbox"/>	Alias	Issued To
<input type="checkbox"/>	addtrustclass1ca	CN=AddTrust Class 1 CA Root,OU=AddTrust TTP Net
<input type="checkbox"/>	addtrustexternalca	CN=AddTrust External CA Root,OU=AddTrust External

Figure 5-34 Import a signer certificate

2. Specify the certificate alias and the file to import, as shown in Figure 5-35. Because the signer certificate does not contain a private key, it is not secured using a password. If the certificate is not signed by a trusted CA, select the **Bypass Chain-of-Trust checks** option. Click **Import**, as shown in Figure 5-35.

Import Certificate

Alias: Company1CA

Trusted: (Bypass Chain-of-Trust checks)

Content: From file:
company1_ca_signercert.cer

From clipboard:
[empty text area]

|

Figure 5-35 Import a signer certificate details

The imported certificate displays in the Trust Store list, as shown in Figure 5-36.

Key Store (1)						
	Alias	Issued To	Issued By	From Date	To Date	
<input type="checkbox"/>	factory supplied identity	C=US,ST=CA,L=Mountain View,O=Cast	C=US,ST=CA,L=Mountain View	12/16/2010	12/15/2012	
Generate Delete Import						
Trust Store (77)						
	Alias	Issued To	Issued By	From Date	To Date	
<input type="checkbox"/>	company1ca	CN=Company1,O=Company1CA,C=US,ST=CA,L=Mountain View,O=Cast	CN=Company1,O=Company1CA,C=US,ST=CA,L=Mountain View	10/21/2011	10/18/2021	

Figure 5-36 Signer certificate successfully imported into Trust Store

- Verify the content of the certificate by clicking the Alias name. Information about the certificate displays, as shown in Figure 5-37.

Certificate Details - company1ca	
Issued To	
Common Name (CN)	Company1
Organization (O)	Company1CA
Country (C)	US
State (ST)	State1
Locale (L)	City1
Serial Number	E2:35:4A:5B:D2:3E:19:49
Issued By	
Common Name (CN)	Company1
Organization (O)	Company1CA
Country (C)	US
State (ST)	State1
Locale (L)	City1
Validity	
Issued On	10/21/2011
Expires On	10/18/2021
Fingerprints	
SHA1 Fingerprint	29:52:9D:D6:99:CC:34:FC:14:16:1C:E8:70:10:17:15:C1:5D:E4:27
MD5 Fingerprint	3D:FC:5A:73:E0:E1:4A:1B:C2:A0:06:11:99:64:B5:F6

Figure 5-37 Signer certificate details

Adding certificates to the Key Store

You can import a certificate in the Key Store or in the Trust Store. To insert a certificate into a Key Store:

- In the WMC, go to **Security → Certificates**, and in the Key Store section click **Import**, as shown in Figure 5-38.

Home	Certificates	
	Key Store (1)	
		Alias
	<input type="checkbox"/>	factory supplied identity
		C=US,ST=CA,L=Mountain View
	Generate	Delete
	Import	
Trust Store (78)		

Figure 5-38 Import a certificate into the Key Store

- Specify the certificate alias, the password, and the file to import, as shown in Figure 5-39. If the certificate is not yet signed by a trusted CA, select the **Bypass Chain-of-Trust checks** option. Click **Import**.

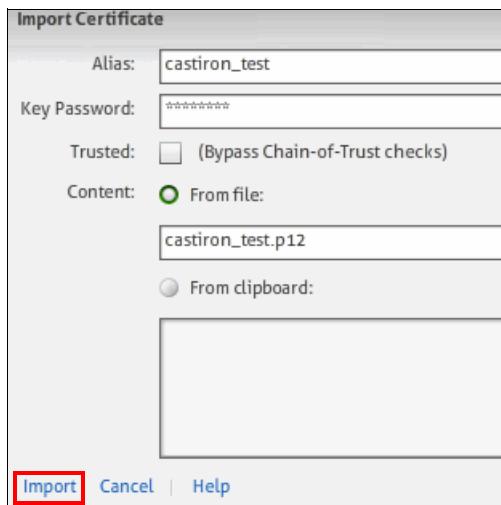


Figure 5-39 Import options for a certificate to be placed into Key Store

If you receive the message shown in Figure 5-40, the certificate might be corrupted. You must create the certificate again.

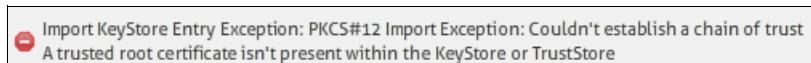


Figure 5-40 Error during import in Key Store

If certificate corruption is not the problem, select the **Bypass Chain-of-Trust checks** option, and then try the import again.

After the certificate is imported, it is listed in the Key Store list, as shown in Figure 5-41.

Key Store (2)		
	Alias	Issued To
<input type="checkbox"/>	castiron_test	C=US,ST=CA,L=CA,O=test,CN=castiron_test
<input type="checkbox"/>	factory supplied identity	C=US,ST=CA,L=Mountain View,CN=factory supplied identity

Figure 5-41 Successfully imported certificate in the Key Store

- Verify the content of the certificate by clicking the Alias name. A window displays information about the certificate, as shown in Figure 5-42.



Figure 5-42 Details of the newly imported certificate

Creating a self-signed certificate

To create a self-signed certificate:

- In the WMC, go to **Security → Certificates**, and in the Key Store section click **Generate**, as shown in Figure 5-43.

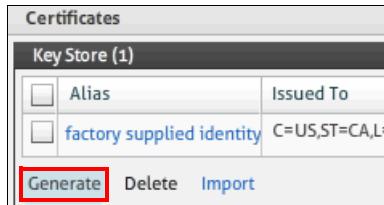


Figure 5-43 Generate self-signed certificate

- Enter the values for your company. Figure 5-44 shows sample data. Click **Generate**.

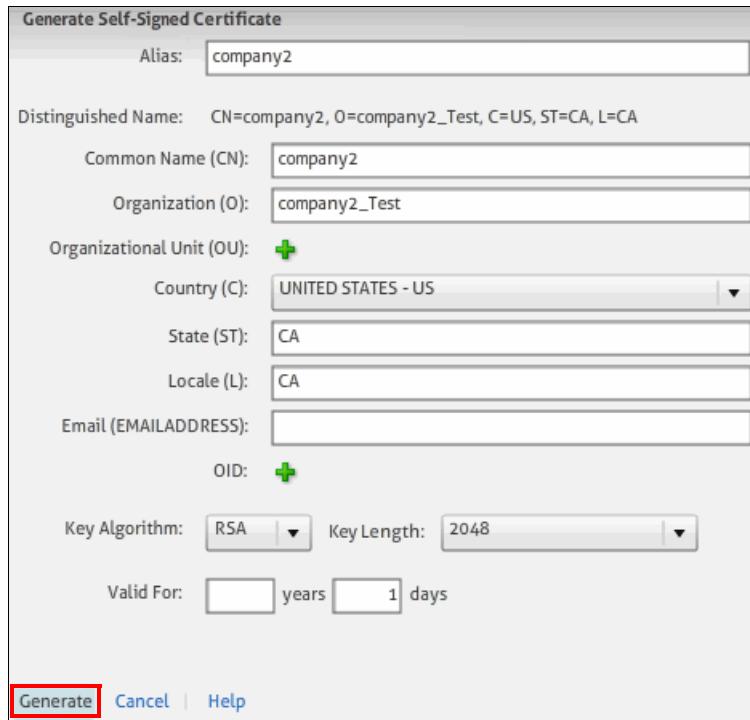


Figure 5-44 Self-Signed certificate sample data

- The content of the certificate displays and is available for download or copy, as shown in Figure 5-45. Because you do not need the certificate outside of WebSphere Cast Iron, click **Close**.



Figure 5-45 Content of the self-signed certificate

As shown in Figure 5-46, the self-signed certificate is placed into the Key Store.

Certificates					
Key Store (2)					
Alias	Issued To	Issued By	From Date	To Date	
company2	C=US,ST=CA,L=CA,O=company2_Test,CN	C=US,ST=CA,L=CA,O=company2_Test,CN	10/21/2011	10/22/2011	
factory supplied identity	C=US,ST=CA,L=Mountain View,O=Cast I	C=US,ST=CA,L=Mountain View,O=Cast I	12/16/2010	12/15/2012	

Figure 5-46 The self-signed certificate in the Key Store

Creating a certificate trusted by a certificate authority

To have a certificate signed by a CA, a certificate request must be sent to the CA. The CA then sends back a certificate that is signed by the CA.

First, you must create a certificate request. To create a certificate request, use an existing certificate, and create a request for that certificate. If you want to create a new certificate, generate a self-signed certificate, and then send the request for that certificate. The following example uses a self-signed certificate with the alias *company2* that was created in “Creating a self-signed certificate” on page 255.

To create this certificate request:

1. In the WMC, go to **Security** → **Certificates**, and in the Key Store section click the **company2** certificate alias, as shown in Figure 5-47.

Certificates					
Key Store (2)					
Alias	Issued To	Issued By	From Date	To Date	
company2	C=US,ST=CA,L=CA,O=company2_Test,CN	C=US,ST=CA,L=CA,O=company2_Test,CN	10/21/2011	10/22/2011	
factory supplied identity	C=US,ST=CA,L=Mountain View,O=Cast I	C=US,ST=CA,L=Mountain View,O=Cast I	12/16/2010	12/15/2012	

Figure 5-47 Select certificate for certificate request

- Verify that the certificate contains the correct information for the request, and then click **Generate CSR** to generate a certificate signer request, as shown in Figure 5-48.



Figure 5-48 Generate certificate signer request

The certificate signer request displays similar to that shown in Figure 5-49.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIC1zCCAX8CAQAwUjERMA8GA1UEAxMIY29tcGFueTIxFjAUBgNVBAoMDWNvbXBhbnkyX1Rlc3QxCzABgNVBAcTakNBMQswCQYDVQQIEwJDQTElMAkGA1UEBhMCVVMwggiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQCVv1xMCKTZgULSSrc08oZp8NnTW2Y4uP4NFjmIQRIP0Jsl+yei6hkmU
x8dx46bn5KeBk3ujMwfWmoR92ZZOjlij0DMgxZlehZ+Qe3mFF/AyV6MiSgCqN2Qp45XBAWUUuKMy
ujMRZvwRckD0X/9m7XN9mPOGGfEamMeB6t7D5PmK0kwD6EyfkDNODSPg5uFU/zU1MwDrJ9tAmzZH
LUNaSDBZJKz9h6QqhqyOX21p5y5in3Ym360RZPne9rGw1k408V+AQ3QGUa0Qa9ARup3kOLj+EhWm
d8dSw2w1RVpi+mvQBCnVxoDs7/kuXgNR+L8sprco5wGPzzm1wAwff1ZTAgMBAAGgADANBgkqhkiG
9w0BAQQFAAOCAQEAj1P+ffMHOyVRp49TUgkj11IxkI+NPFHKIpILHLWJYFxAaiyi00XuLP6/yw8
Yi8M/oE/JY+MbR706CfV9L+rbeNVcpxxXwAkGIK+nNAjChVdc0CAgwMvpUCIo01wMXwRp82aNFOne
elMAK7JYSV3n8bW7ntGP79suHzJZ8dUQ2Z2hQyynbopJUcoirZtHJe0dqadKnqIkPDrtEOFoS67
EPPOmsvFhskw1G8HJ5ooI9bjUvg7zRNg2o8aeYhItPZUyx1W+OkA0W1qe6mVwPkqxzpA8fG2AA
3Trh/66V/GXkDuru+F2taus4yLpTQ9ejJR1P22FgccJ5/e7tg+ikUw==
-----END NEW CERTIFICATE REQUEST-----
```

At the bottom, there are buttons for **Copy** (highlighted in red), **Download**, and **Close**.

Figure 5-49 Sample certificate signer request

You can download the request as a .pem file (for example, company2-csr.pem) using the Download function, or you can copy the text to initiate the request. Depending on the CA, you are requested either to send the text through email or a portal or to send the .pem file. After you save the request, click **Close**.

- Send the certificate request to the CA.

When the signed certificate is returned, import it using the WMC:

1. In the WMC, go to **Security** → **Certificates**. In the Key Store section, click the **company2** certificate alias, as shown in Figure 5-47 on page 257.
2. Verify that the certificate contains the correct information for the request, and click **Upload** to upload the certificate received by the CA to the run time, as shown in Figure 5-50.

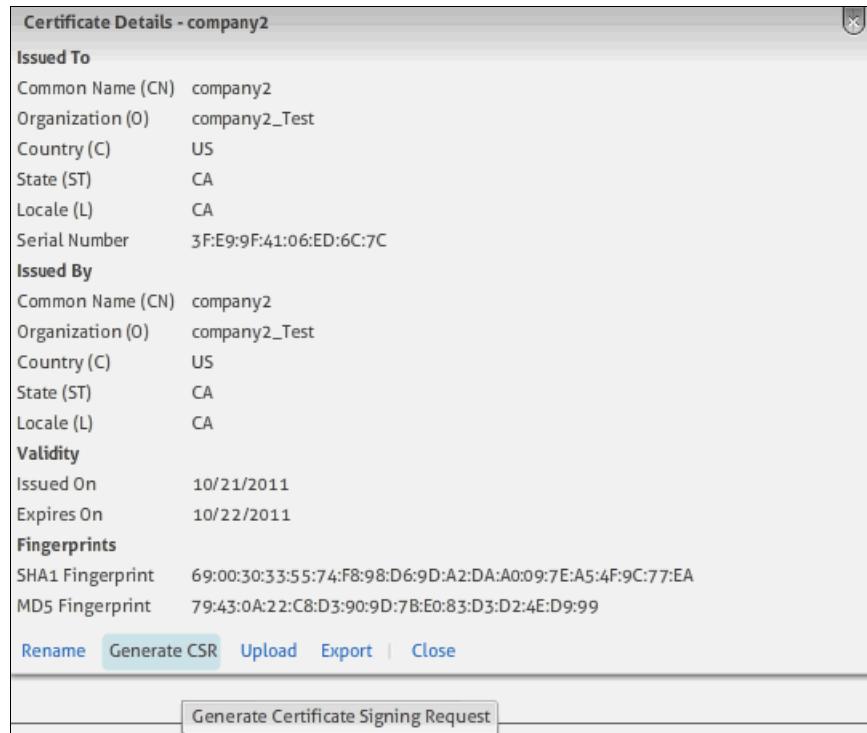


Figure 5-50 Upload the CA certificate

3. In the Update Certificate window, specify the name of the certificate file, and then click **Import**. If the signer certificate of the CA that signed your certificate is not in the Trust Store, you might have to select the **Bypass Chain-of-Trust checks** option, as shown in Figure 5-51.

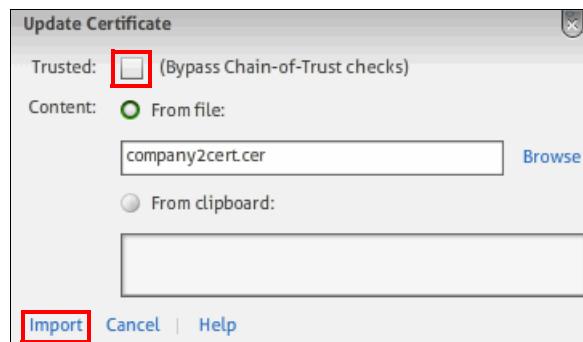


Figure 5-51 Update certificate

4. You are routed back to Figure 5-50 (still with the old values). Click **Close**.

- Back on the main window, the certificate is updated. As shown in Figure 5-52, the Issued By column now shows that CA and the To Date changed to 10/23/2012. Compare the values shown here to those shown in Figure 5-46 on page 257.

Certificates					
Key Store (2)					
	Alias	Issued To	Issued By	From Date	To Date
<input type="checkbox"/>	company2	C=US,ST=CA,L=CA,O=company2_Test,CN=company2	CN=Company1,O=Company1CA,L=CA	10/24/2011	10/23/2012
<input type="checkbox"/>	factory supplied identity	C=US,ST=CA,L=Mountain View,O=Cast I	C=US,ST=CA,L=Mountain View,O=Cast I	12/16/2010	12/15/2012

Figure 5-52 Updated certificate

Assigning a certificate to an SSL usage type

In “Creating a self-signed certificate” on page 255 and in “Creating a certificate trusted by a certificate authority” on page 257, we discussed how to create an individual certificate. This section explains how to use it.

To use a certificate, you must assign it to a specific SSL usage type. The following usage types are defined in the WMC:

- ▶ Client SSL for outbound mutual SSL with another server
- ▶ Server SSL over a data network interface card (NIC) for inbound SSL communication on the data network
- ▶ Server SSL over a mgmt NIC for inbound SSL communication on the data network

Tip: Server SSL over a data NIC can be specified globally in the WMC but can be overwritten in an activity or endpoint, for example in an HTTP Receive Request or in a web service Provide Service endpoint.

The following steps show you how to replace the global certificate in the WMC. This example shows how to replace the certificate for the management network, but the approach for the other usage types is the same. To replace the global certificate in the WMC:

- In the WMC, go to **Security → Certificates**, and view the Settings section. The default factory supplied identity certificate is assigned to all three SSL usage types. Click **Edit**, as shown in Figure 5-53.

Settings					
SSL Usage Type	Certificate Alias	VPeer	VHost	Cipher Strength	
Client SSL	factory supplied identity	✓		Strong	
Server SSL over data NIC	factory supplied identity			Strong	
Server SSL over mgmt NIC	factory supplied identity	✓	✓	Strong	
Edit					

Figure 5-53 Assigned certificate

- Click the drop-down menu to the right of the Server SSL over mgmt NIC option to display a list of all certificates that are available for assignment. Select the **company2** certificate, as shown in Figure 5-54.

Edit Settings				
SSL Usage Type	Certificate Alias	VPeer	VHost	Cipher Strength
Client SSL	factory supplied identity	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Strong
Server SSL over data NIC	factory supplied identity	<input type="checkbox"/>	<input type="checkbox"/>	Strong
Server SSL over mgmt NIC	factory supplied identity -- none -- company2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Strong

Figure 5-54 Assign Server SSL over mgmt NIC certificate

- Click **Save**.

The company2 certificate is now listed for usage type Server SSL over mgmt NIC, as shown in Figure 5-55.

Settings				
SSL Usage Type	Certificate Alias	VPeer	VHost	Cipher Strength
Client SSL	factory supplied identity	✓		Strong
Server SSL over data NIC	factory supplied identity			Strong
Server SSL over mgmt NIC	company2	✓	✓	Strong

Figure 5-55 New server SSL over mgmt NIC certificate has been assigned

- Log out of the WMC.
- Restart the network using the **net restart** CLI command.
- Log in again and the new certificate is listed, as shown in Figure 5-56.

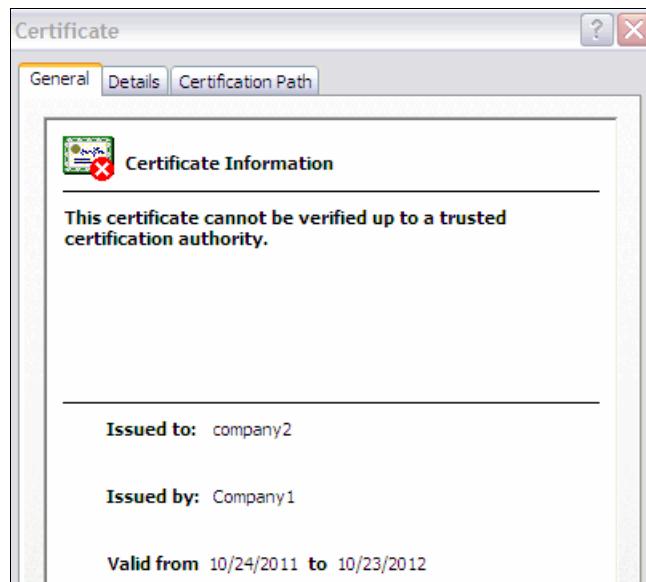


Figure 5-56 New certificate displays at login

5.6 Securing communication with endpoints

This section describes options for communicating with an endpoint in a secure mode. It differentiates between inbound communication and outbound communication. In an inbound scenario, the Integration Appliance is called by a client, and in an outbound scenario, the Integration Appliance calls a server.

A communication between client and server can be secured on two levels:

- ▶ The communication is encrypted on the transport level.
- ▶ The server and possibly the client must authenticate each other.

Both mechanisms are often combined to provide more security.

If you connect to a web page using HTTPS, the following events occurs (in this order):

1. The browser sends a request to the server and asks for authentication.
2. The server sends its certificate to the browser.
3. The browser verifies that the certificate is valid.
4. The channel between client and server is then encrypted using the key in the certificate.

Tip: Cast Iron Live includes an option to use the WebSphere Cast Iron Secure Connector to provide a secure connection to endpoints behind a firewall. For more details, refer to 5.6.1, “The WebSphere Cast Iron Secure Connector” on page 262.

5.6.1 The WebSphere Cast Iron Secure Connector

The WebSphere Cast Iron Secure Connector is a component available for WebSphere Cast Iron Live. It provides secure bidirectional data transfer between WebSphere Cast Iron Live and endpoints that are located behind a firewall. The Secure Connector consists of the following components:

- ▶ The Secure Connector component in WebSphere Cast Iron Live

You must configure the Secure Connectors that you want to use in WebSphere Cast Iron Live. Secure Connectors are defined in the scope of an environment.

- ▶ The Secure Connector component in the enterprise

You must install the Secure Connector program within your on-premise environment that you want to connect to WebSphere Cast Iron Live. You can download the program and configuration properties from WebSphere Cast Iron Live and then install them anywhere within your network when the connector has access to the cloud and to the endpoints to which you want the cloud to connect.

For information about installing the Secure Connector, refer to 2.6, “Installing the WebSphere Cast Iron Live Secure Connector” on page 60.

Figure 5-57 shows the architectural view. There are separate Secure Connectors for the Test and the Production cloud environments.

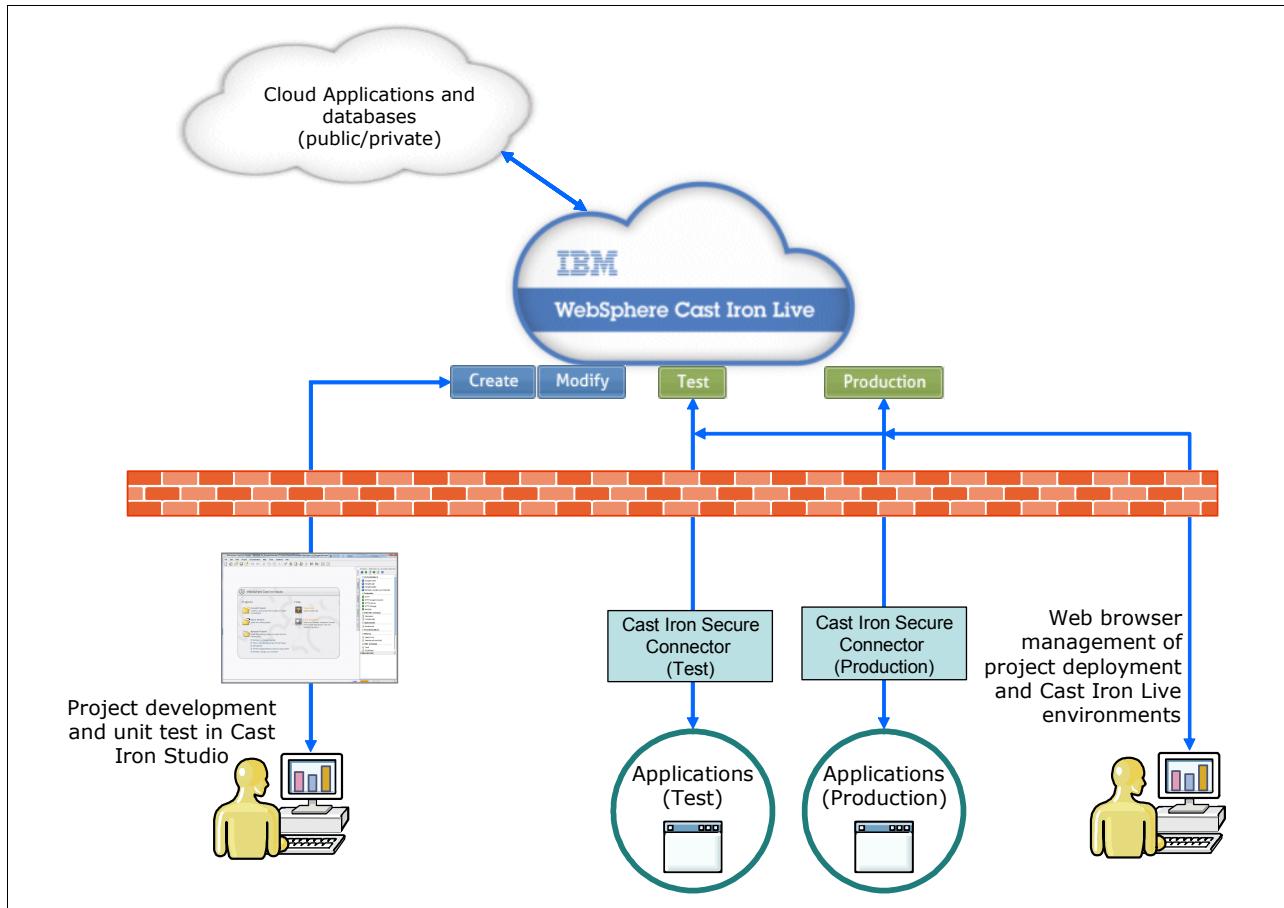


Figure 5-57 The Secure Connector architecture

Tip: WebSphere Cast Iron Studio cannot access the Cast Iron Secure Connector that is used by Cast Iron Live to access an external system running in a private network behind a firewall. Therefore, to configure access to an external system that is running in a private network, Studio must be provided with a direct connection in to this private network so that it can access the external system at development time.

How the Secure Connector works

The Secure Connector creates a secure channel between your WebSphere Cast Iron Live environment and your intranet. The Secure Connector in your local system uses SSL to connect to the WebSphere Cast Iron Live environment that is defined in the connector for bidirectional communication.

The Secure Connector includes the following features:

- ▶ The channel is initiated from your intranet but can be used for bidirectional communication.
- ▶ Management of the Secure Connector can be done only from the intranet. Although the initial configuration is done in WebSphere Cast Iron Live, the start and stop of a Secure Connector can be done only with access to the Secure Connector system.
- ▶ No ports for inbound communication have to be opened in the firewall.

- ▶ No route to an endpoint has to be defined in the intranet.
- ▶ The communication over a channel is secured with mutual SSL. The Secure Connector and Cast Iron Live must validate each other at start using a certificate that was exchanged during configuration and installation.
- ▶ The communication from the Secure Connector in the intranet to the endpoint in the intranet is done through a native protocol (for example BAPI call, JDBC, MQ, JMS, and so on).
- ▶ Each runtime environment has its own secure connector to ensure that there is no way to bridge different environments when not intended.
- ▶ Creating and modifying Secure Connector configurations needs administrative rights in the cloud environment.

Why use the Secure Connector

The Secure Connector can help you to overcome limitations caused by a firewall between the endpoint and the orchestration run time. It builds a secure channel between Cast Iron Live and your local network in the following scenarios:

- ▶ Outbound/consume
The orchestration must access an endpoint that is behind a firewall.
- ▶ Inbound/provide
The orchestration is started with an HTTP Receive Request starter activity, and you want to move the HTTP listener behind the firewall.

The Secure Connector helps you to bridge between the Internet and intranet. For example, if you implement an integration solution between SAP and Salesforce.com, the orchestration takes an HTTP request as trigger. It then reads data from SAP and stores the data into Salesforce.com.

If you want to run this integration in WebSphere Cast Iron Live, your job configuration must be able to communicate with the Salesforce.com and SAP endpoints and must be accessible through HTTP and HTTPS.

Accessing endpoints without the Secure Connector

The following list explains how to access endpoints without Secure Connector for comparison:

- ▶ Salesforce.com
Communication is easy because the orchestration and Salesforce.com are running in the cloud and both provide functionality to be accessible through the Internet.
- ▶ SAP
To make a BAPI call from WebSphere Cast Iron Live to an SAP system in the intranet without using the Secure Connector, you must complete the following steps:
 - a. Open the firewall to allow an inbound BAPI call to SAP.
 - b. Define a route so that the firewall knows where the SAP system is located.
 - c. Secure the communication as the BAPI call uses an unsecure network.
- ▶ HTTP Receive
You can have the HTTP Receive endpoint running in the cloud, but then everybody who wants to access the orchestration using HTTP and HTTPS must have WebSphere Cast Iron Live credentials to access the orchestration.

Alternatively, you can install a proxy in the intranet that receives the HTTP request, does the authentication to the cloud, and sends the request to the HTTP endpoint in the cloud.

In both cases, you must secure the communication even though the original HTTP call might not use HTTPS.

The requirement to open the firewall for inbound requests is challenging. Security is even more challenging if these requests then have to use ports other than the HTTP ports.

Accessing endpoints with the Secure Connector

To access endpoints with the Secure Connector, you have the following options:

- ▶ Salesforce.com

The communication is the same as without the Secure Connector.

- ▶ SAP

When the Invoke BAPI activity is processed, the secure call to the Secure Connector is done. Within the intranet, the Secure Connector uses SAP libraries to make a native BAPI call to SAP.

- ▶ HTTP Receive

When the orchestration is started, an HTTP listener is started on the system that is running the Secure Connector. An HTTP request sent to that listener is routed over a secure channel to WebSphere Cast Iron Live and is processed there as though it was a local inbound call. You do not have to specify any user in WebSphere Cast Iron Live to make the orchestration available to users in your intranet.

Configuring the Secure Connector at an endpoint

The configuration of a Secure Connector is similar for each endpoint. The endpoint configuration panel includes a Remote Endpoint Configuration section where you can select to use a Secure Connector by enabling the Endpoint Runs Behind Firewall option. You can also define the Secure Connector Name. The name must be valid in WebSphere Cast Iron Live. Configure the Secure Connector name as a configuration property, as shown in Figure 5-58.



Figure 5-58 Remote endpoint configuration

The installation and initial configuration of the WebSphere Cast Iron Secure Connector is discussed in 2.6, “Installing the WebSphere Cast Iron Live Secure Connector” on page 60.

Managing the Secure Connector

You manage the Secure Connector with the following actions:

- ▶ After you configure the Secure Connector in WebSphere Cast Iron Live and install it on the local system, you must start it on the local system. To start the Secure Connector, use the following command, or the shortcut in the Windows Start menu.

`<installation directory>|runclient osgi.bat start`

- ▶ You can monitor the status in WebSphere Cast Iron Live.

- ▶ The Secure Connector can be stopped only from the local Secure Connector installation. To stop the Secure Connector, use the provided shortcut or the following command:
`<installation directory>\runclient osgi.bat stop`
- ▶ You can configure the Secure Connector with regard to ports and so forth with the following command:
`<installation directory>\jre\jre\bin\java.exe -Dconfigure.agent=true -jar agent_configurator.jar`
- ▶ You can find the logs for troubleshooting in the `<installation directory>/logs` directory.

5.6.2 Authentication mechanisms

The following most common authentication mechanisms are available for use with endpoints in WebSphere Cast Iron, depending on the endpoint:

- ▶ Anonymous: The client does not send any authentication information to the server.
- ▶ Client or server authentication through a Certificate: The client or server sends a trusted certificate to the other party, which is then verified.
- ▶ Client authentication using Basic Authentication: The client sends the user name and password unencrypted to the server.
- ▶ Client authentication using Digest Authentication: The client sends the user name and password encrypted to the server. The encryption is done based on user, password, a realm and a nonce received by the server.
- ▶ Client authentication using NTLM: Hand-shaking protocol where the server sends a random number to the client. The client does some calculation based on that number, the user password, and other identification information.
- ▶ Client/Server authentication using Kerberos: Kerberos is a security protocol allowing both client and server to mutually verify each other's identity. An authentication service verifies the user's identity and grants an encrypted ticket.

Using certificates for authentication

To use certificates for authentication:

1. Create a certificate. You can also use a default certificate. Refer to 5.5, “Working with certificates” on page 250.
2. Configure the endpoint to use the certificate.

Using Kerberos for authentication

To use Kerberos for authentication:

1. Configure Kerberos in the appliance.
 - a. In the WMC, click **Security** → **Kerberos**. A panel displays, as shown in Figure 5-59.

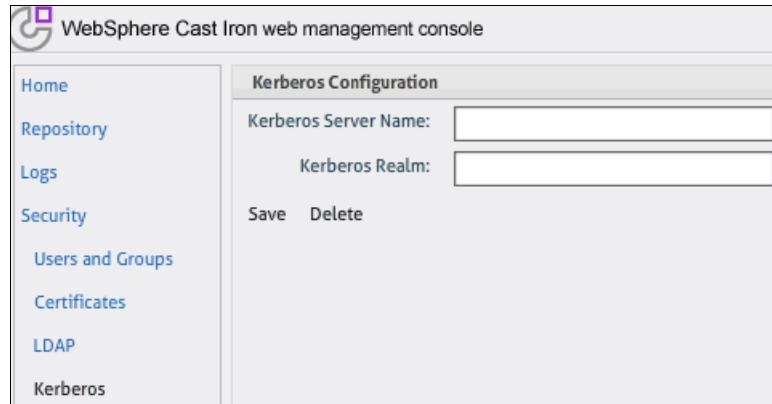


Figure 5-59 Kerberos configuration in WMC

- b. Enter the name or IP address of the Kerberos Key Distribution Center (KDC) server and the Kerberos realm.
- c. Click **Save**.

Tip: If you want to test Kerberos within Studio, specify the server and realm in Studio using **Edit** → **Preferences** → **SSL/Kerberos**.

2. Configure the endpoint to use Kerberos (see “Using authentication mechanisms for outbound endpoints” on page 271).

Configuring authentication mechanisms for inbound endpoints

This section describes the authentication mechanisms for inbound HTTP requests. For most endpoints, the settings and concepts are similar. Check the information center for details about a specific endpoint.

Figure 5-60 shows the HTTP endpoint for inbound configuration panel. Open the endpoint by double-clicking the entry in the Projects tab under Endpoints or by double-clicking the endpoint in the orchestration graph.

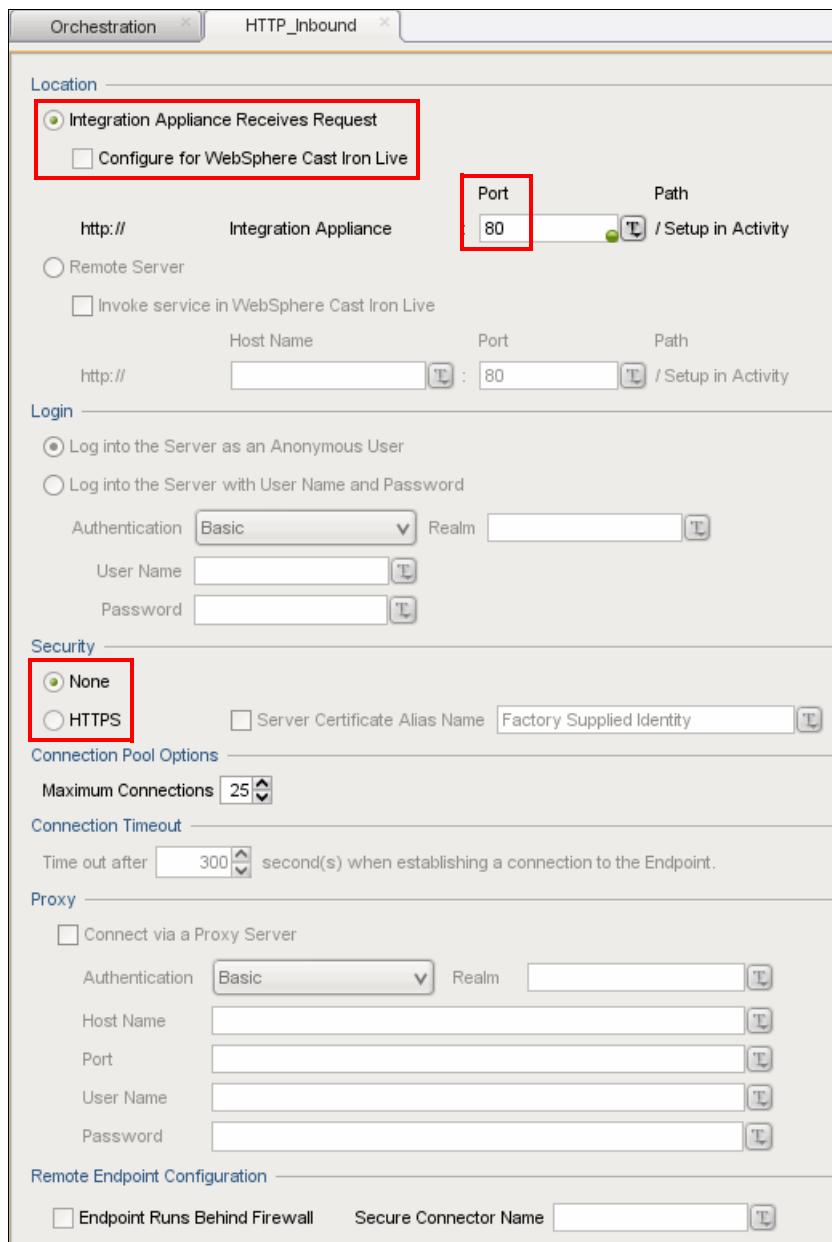


Figure 5-60 Security settings for HTTP inbound

The Login options are disabled because they are not available for HTTP inbound. In the Security section, you can specify the use of SSL, and if so, the certificate to use:

► **None**

The endpoint is provided only through HTTP without any security. No encryption or authentication takes place.

► **HTTPS**

The endpoint is provided through HTTPS.

Tip: If you switch from HTTP to HTTPS, the endpoint settings for the port stay the same and are not switched to port 443.

If you use HTTPS for an HTTP inbound endpoint, select the **Server Certificate Alias Name** option, as shown in Figure 5-61. This option specifies whether the Integration Appliance provides a specific certificate to the client. If you enable this option, you must insert a name for a valid certificate. The Integration Appliance has a default certificate for the data interface.



Figure 5-61 HTTP Inbound server certificate

You can define the server certificate for the data network as follows:

- ▶ If you do not enable the Server Certificate Alias Name option, the certificate that is specified in the WMC (or through the CLI) is taken.
- ▶ If you select the Server Certificate Alias Name option, the certificate that is specified at the endpoint level takes precedence over certificates that are specified at the Integration Appliance level.
- ▶ If you enable the Server Certificate Alias Name option and the name that is specified for the certificate is not valid, the orchestration will not start.

Tip: Configure the Server Certificate Alias Name option as a configuration property to simplify adding or replacing certificates.

Configuring the inbound endpoint for WebSphere Cast Iron Live

In the Remote Endpoint Configuration section, you can specify that you want to use the Secure Connector. This setting provides the option to run the HTTP endpoint in your local environment and is discussed in detail in 5.6.1, “The WebSphere Cast Iron Secure Connector” on page 262.

If you want to run the endpoint in WebSphere Cast Iron Live, other rules apply. Select the **Configure for WebSphere Cast Iron Live** option, as shown in Figure 5-62.

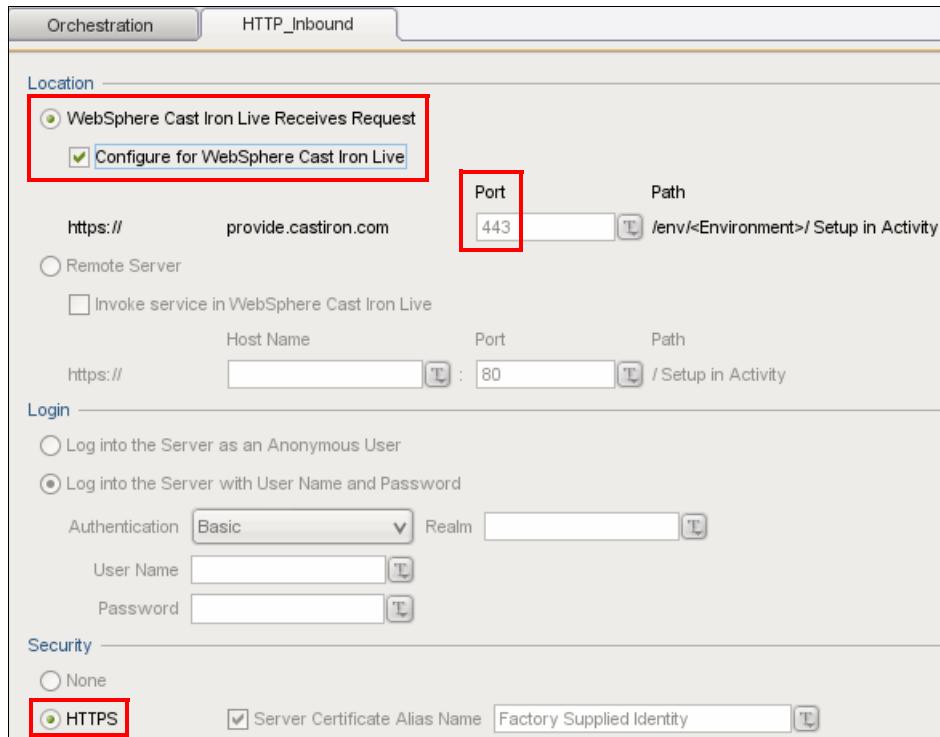


Figure 5-62 Security settings for HTTP inbound for WebSphere Cast Iron Live

When you select this option, HTTPS is selected automatically and the port changes to 443. You cannot change either the port or the certificate.

For additional information, visit:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.doc%2Fhttp_create_edit_endpoint.html

Using authentication mechanisms for outbound endpoints

This section describes the authentication mechanisms for HTTP outbound. For most endpoints, the settings and concepts are similar. Check the information center for details about a specific endpoint.

Figure 5-63 shows the HTTP endpoint for the outbound configuration panel.

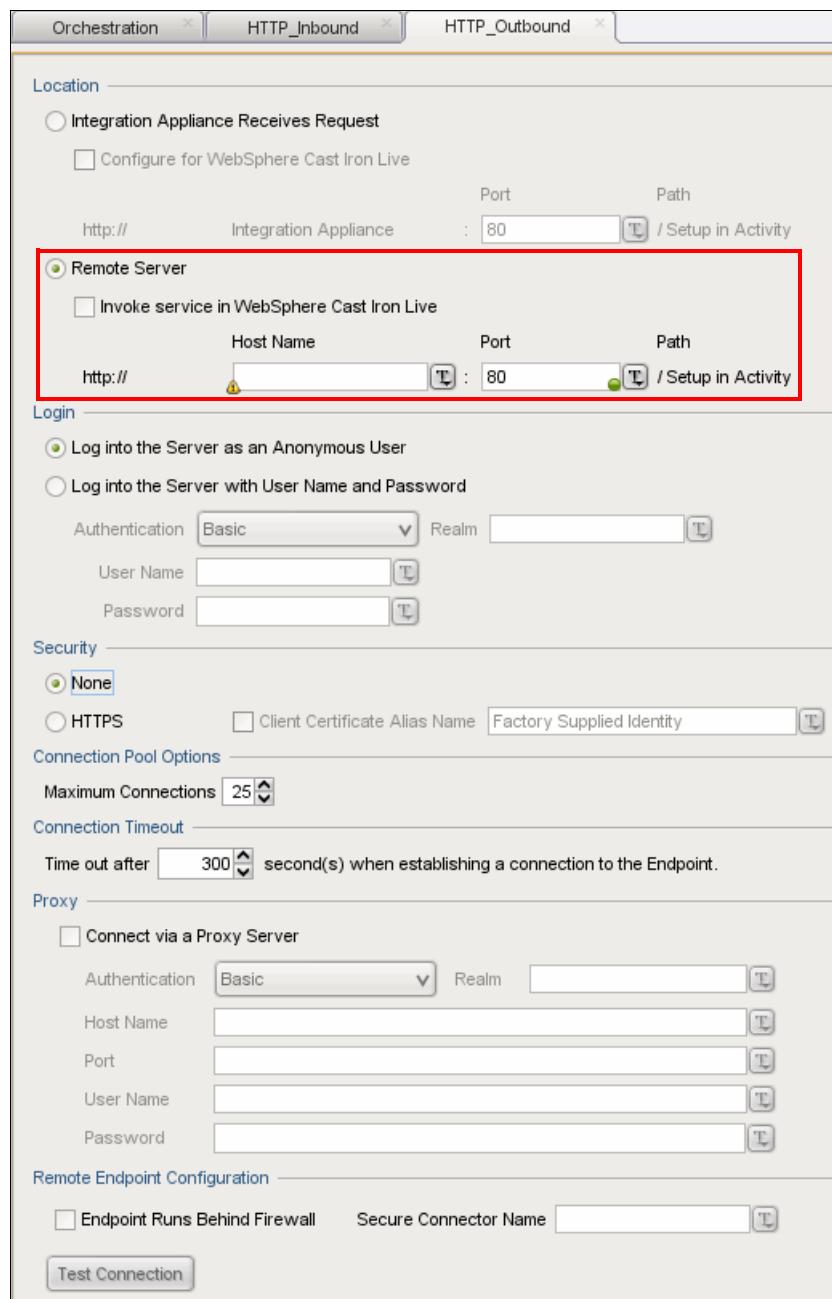


Figure 5-63 Security settings for HTTP outbound

In the Location section, select **Remote Server**, and specify the Host Name and Port of the remote server that hosts the service you want to call. Specify the URI in the HTTP Post activity. You can access the service from a browser to determine whether it needs a user ID and password.

Outbound settings include the following options for login:

- ▶ Log into the Server as an Anonymous user

This setting is the default setting and needs no further configuration. It works only if the service does not need a user ID and password.

- ▶ Log into the Server with User Name and Password—Basic Authentication

If the remote service requires a user name and password, you can use basic authentication to provide this information. Select **Basic** for authentication, and provide the user ID and password. Depending on the endpoint, you might also need to specify a realm as shown in Figure 5-64.

The screenshot shows a 'Login' dialog box. At the top, there are two radio button options: 'Log into the Server as an Anonymous User' (unchecked) and 'Log into the Server with User Name and Password' (checked). Below these, there is a dropdown menu labeled 'Authentication' set to 'Basic', and a text input field labeled 'Realm' containing 'Any realm'. Below the authentication section are two text input fields: 'User Name' containing 'RemoteServiceUser' and 'Password' containing 'remoteServicePassword'.

Figure 5-64 Calling a remote service using basic authentication

- ▶ Log into the Server with User Name and Password—Digest Authentication

This option is similar to basic authentication, as shown in Figure 5-65.

The screenshot shows a 'Login' dialog box. The 'Authentication' dropdown is set to 'Digest', and the 'Realm' input field contains 'pteServiceDigestRealm'. The other fields ('User Name' and 'Password') are identical to Figure 5-64.

Figure 5-65 Calling a remote service using digest authentication

- ▶ Log in to the Server with User Name and Password—NTLM Authentication

With NTLM, you do not provide the credentials to the remote server but to a third instance, the NTLM server. You must specify the NTLM user credentials and the NTLM domain, as shown in Figure 5-66.

The screenshot shows a 'Login' dialog box. The 'Authentication' dropdown is set to 'NTLM', and the 'Domain' input field contains 'NTLMDomain'. The 'User Name' field contains 'NTLMUser' and the 'Password' field contains 'NTLMPassword'.

Figure 5-66 Calling a remote service using NTLM authentication

- Log into the Server with User Name and Password—Kerberos Authentication

The configuration of Kerberos requires the following parameters:

- The Kerberos Key Distribution Center (KDC) server and the Kerberos realm are defined for the appliance as described in “Using Kerberos for authentication” on page 267.
- The Kerberos user name and password are defined per the endpoint in the Login section as shown in Figure 5-67.



Figure 5-67 Calling a remote service using Kerberos authentication

In the Security section, you can select HTTPS to use SSL, as shown in Figure 5-68.



Figure 5-68 Calling a remote service using HTTP or HTTPS

Tip: If you switch from HTTP to HTTPS, the endpoint settings for the port stay the same and are not switched to port 443.

If you use HTTPS for the HTTP inbound endpoint, select the **Client Certificate Alias Name** option. This option specifies whether the Integration Appliance provides a client certificate to the server. If you select this option, you must insert a name for a valid certificate.

The Integration Appliance has a default client certificate. You can define the client certificate for the data network as follows:

- If you do not select the Client Certificate Alias Name option, the certificate that is specified in the WMC (or through the CLI) is taken.
- If you enable the Client Certificate Alias Name option, the certificate that is specified at the endpoint level takes precedence over certificates that are specified at the Integration Appliance level.
- If you select the Client Certificate Alias Name option and the name that is specified for the certificate is not valid, the orchestration will not start.

Tip: Configure the Client Certificate Alias Name option as a configuration property to make it simple to add or replace certificates.

Configuring the outbound endpoint for WebSphere Cast Iron Live

In the Remote Endpoint Configuration section, you can specify that the remote HTTP endpoint is behind a firewall and must be accessed using Secure Connector. To configure this option correctly, you must specify the name of a secure connector that was defined in

WebSphere Cast Iron Live. For flexibility, use a configuration property instead of a fixed name, as shown in Figure 5-69.



Figure 5-69 Remote Connector configuration for outbound

If you want to access a server running in WebSphere Cast Iron Live, other rules apply. Select the **Invoke service in WebSphere Cast Iron Live** option, as shown in Figure 5-70.

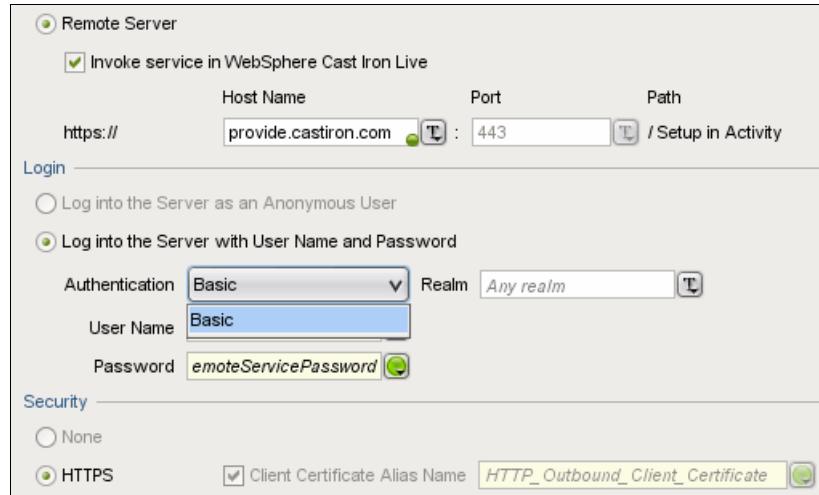


Figure 5-70 Invoke HTTP outbound service in WebSphere Cast Iron Live

The following changes apply:

- ▶ The host name is set to provide.castiron.com.
- ▶ Only HTTPS is allowed and the port is set to 443.
- ▶ Only Basic authentication is allowed.

For additional information, visit:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.doc%2Fhttp_create_edit_endpoint.html

5.7 Monitoring security

Security errors, such as authentication failures, are stored in the system log with a log level of Error. You can view the alerts in the System Log from the WMC by selecting **Logs → System Log**. You can filter the view to show security alerts by setting the System filter to Security, as shown in Figure 5-71.

System Log							
Minimum Level:		All Levels	System:	Security	Date:	All Dates	Resolved:
Level	System	Message		Job	When	Resolved On	Resolved By
Error	Security	Authentication failure for user: test, from host: unknown			10/19/2011 11:49:51	10/20/2011 12:38	user1
Error	Security	Authentication failure for user: admin, from host: unkno			10/16/2011 11:26:49		<input checked="" type="checkbox"/>

Figure 5-71 System Log with security alerts in WMC

For security reason, alerts and errors cannot be removed from the System Logs. However, you can mark them as Resolved and filter them so that an administrator can manage the number of alerts that they must view. The person that marks a record as Resolved is documented for the record with date and time. To resolve an alert, User rights are sufficient.

You can create a policy (**Logs → Notifications**) that indicates that a notification must be sent automatically when a critical security alert occurs. You can specify that security alerts only of a specific level trigger a notification and specify who receives the notification, as shown in Figure 5-72.

New Policy

Policy Name: Security Error Policy

Notify when: Level exceeds Error with system Security

Notify how: Email securityadmin@mycompany.com
(enter multiple email addresses, one per line)

SNMP

Save Cancel | Help

Figure 5-72 Security notification for security level Error and above

You can define more than one notification for a topic, for example to send security errors and above to the security officer but security warnings to the administrator.

Although all roles can see notification rules, only an administrator has the right to change them.

5.8 Additional security considerations

Although encryption and authentication achieve a level of security for endpoints, there are additional exposures to consider:

- ▶ Even if your Integration Appliance is running behind the firewall, take steps to restrict the access to the appliance.
- ▶ Take steps to protect your appliance from receiving malformed requests that might influence your environment.
- ▶ Take steps to ensure that a single client cannot cause an Integration Appliance to be busy all the time.

Security appliances, such as the WebSphere DataPower XG45, can help to address these topics. For more information, refer to:

<http://www-01.ibm.com/software/integration/datapower/XG45/>



Availability and scalability planning

Many companies today have applications that provide services continuously to consumers. With continuous service, a production environment must be planned with availability, scalability, and disaster recovery in mind to guarantee seamless operation to consumers.

This chapter provides information about how to configure your Cast Iron Integration Appliances for high availability (HA) and scalability, including:

- ▶ High availability techniques for Integration Appliances
- ▶ Scalability techniques for integration appliances
- ▶ Maintenance and updates
- ▶ Backup and restore

6.1 High availability techniques for Integration Appliances

A production environment experiences situations where critical applications are interrupted by factors beyond their control, such as natural disasters or by human mistakes (technical errors). Failure situations you might encounter include:

- ▶ Hardware failures, for example, the failure of a hard drive, CPU, power supplies, RAM, mother boards, and network interfaces.
- ▶ Runtime failures that can occur due to fatal errors in an orchestration.
- ▶ Communication failures that can occur because of issues with external hardware that controls the communication between different endpoints. Communication failures can also be generated by a network problem.

A highly available design can help ensure that critical services remain available during these occurrences. *High availability*, in general terms, means that an application can automatically recover from interruptions with no manual intervention; therefore, a consumer has access to its services as close to around the clock as possible. A highly available design of your infrastructure provides 24x7 access to applications in both unplanned outages and during periods of scheduled maintenance.

Providing 24x7 access to critical applications takes more than hardware. There must be processes in place to ensure the reliability of the systems, including the effective backup of the applications and their data and protecting the applications from attacks. Orchestrations must also be designed for quick recovery in a highly-available environment.

High availability for Integration Appliances can be approached at three levels. First are the high availability features inherent to the Integration Appliance itself. Second are the availability modes that the Integration Appliances can operate in. Third is the ability of orchestrations to tolerate failover situations.

6.1.1 Integration Appliance availability features

WebSphere Cast Iron Integration Appliance comes in two form factors. There are failover features that are specific to the form factor that is implemented. This section describes these features:

- ▶ WebSphere DataPower Cast Iron Appliance XH40 appliances

The physical appliances are designed with a redundant architecture, which means that their principal hardware components, such as CPU, hard disks, fans, and power supplies, are built in a high availability mode. A redundant architecture makes the device stable and reliable and ensures minimum appliance downtime. To avoid a single point-of-failure you can configure an HA scenario in conjunction with another physical appliance working in standby mode. In the event of a failure of the primary appliance, the standby appliance takes over and continues the processing.

See 2.3, “Configuring the physical appliance for high availability” on page 44.

- ▶ Cast Iron Hypervisor Edition (referred to as the virtual appliance)

Virtual appliances are based on virtualized environments, such as VMware and Xen, which provide their own mechanisms to deal with failures, for example RAID configuration, storage area network (SAN), sources of power supplies, and so on. If a virtual appliance fails, the virtualized environment can start another virtual appliance that takes over the processing.

Some hypervisors allow the movement of virtual machines during run time. For example, if the hypervisor detects that the hardware that is running a virtual appliance has a network problem, the hypervisor can move the virtual appliance to another hardware without stopping the appliance. Whether this type of feature is available depends on the hypervisor. So, with Cast Iron Hypervisor Edition, it is up to the hypervisor to make the underlying hardware and resources highly available.

Cast Iron Live: High availability features are built into WebSphere Cast Iron Live.

6.1.2 Integration Appliance high availability modes

Most high availability solutions require two instances, or, for Cast Iron, two Integration Appliances. Having two parallel Integration Appliances provides redundancy so that if one Integration Appliance fails, the service is still available. This redundancy can be provided using either an active/active or active/standby configuration.

Active/active

In an active/active configuration, two Integration Appliances are actively running project configurations. If a failure is detected in one appliance, the second appliance receives all new work and processes it. An active/active configuration can be achieved, for example, by using a load balancer or a shared input source, such as an FTP server. You can have multiple Integration Appliances running in parallel. Active/active on a Cast Iron Integration Appliance does not support the take over of work in progress.

Active/standby

In an active/standby configuration, only the primary Integration Appliance does work. The standby appliance or a central instance verifies that the primary appliance is still alive using a heartbeat or similar technique so that the standby appliance can take over if the primary is not responsive. Active/standby on a Cast Iron Integration Appliance can be set up to support the take over of work in progress. Configurations that support take over are described in this section.

Cast Iron supports active/standby using the following configurations:

- ▶ Cold standby

In a cold standby scenario, only one appliance is fully working. The other appliance is not started or is only partly started. If the primary appliance fails, the standby appliance is activated and can take over new work. Take over for work in progress might be achieved.

All Integration Appliances support a cold standby configuration. You can implement this configuration using the following methods:

- Both Integration Appliances are started

Both Integration Appliances are started, but the orchestrations are started only on the primary appliance. An external monitoring tool, such as an SNMP monitor, is used to monitor the state of the primary appliance. When an outage of the primary appliance is detected, the monitoring tool calls the management APIs to stop the orchestrations on the primary appliance and to start them on the standby appliance. The monitoring tool is not provided with Cast Iron and must be configured separately.

- Only the primary Integration Appliance is started

Virtual appliance only: This scenario is valid only for virtual appliances because it relies upon the functionality of the Hypervisor on which the virtual appliance is running.

The hypervisor can be configured to automatically start a new Cast Iron Hypervisor instance if the primary appliance fails. This new virtual machine can be on the same physical environment or on another physical environment if the original environment is unavailable. The hypervisor then attempts to restart the image of the primary appliance.

There are two possible situations:

- The image of the primary appliance can be restarted. In this case, the appliance can take over the new work and continue with the work in progress if the orchestrations use persistence.
- The image of the primary appliance cannot be restarted. In this case, a new image of the appliance is started and can take over the new work but cannot continue with the work in progress.

► Warm standby

You can implement a warm standby for all Integration Appliances.

In a warm standby scenario, both appliances are configured for active/active mode, but only the primary appliance receives work. This is not achieved by the Integration Appliances themselves, but instead requires external configuration, for example, if data is being sent to the Integration Appliances using HTTP, an intelligent Load Balancer is needed. This Load Balancer directs messages only to the primary appliance and monitors this appliance to detect its state. If the Load Balancer detects that the primary appliance has failed, the Load Balancer will direct traffic to the secondary appliance. This requires no additional configuration on the Integration Appliances.

A similar technique can be used for most protocols, for example:

- For FTP by using two FTP directories, one for each Integration Appliance
- For WebSphere MQ by using an MQ Cluster with one Queue Manager per Integration Appliance

The secondary appliance takes over new work as quickly as the protocol allows. However, there is no take over for work in progress.

► Hot standby

Physical appliance only: You cannot achieve hot standby with a virtual appliance.

This configuration consists of two physically identical Integration Appliances working together to handle transactions from and to separate endpoints. This configuration is called *HA pairs*. In this case, one Integration Appliance acts in the active role and the other appliance acts in the standby role. These two Integration Appliances are synchronized automatically.

The standby appliance is in charge of synchronization and pulls data from the active appliance. When the two appliances connect to each other for the first time, they synchronize immediately and determine which appliance assumes the active and standby roles. In this initial process, the active appliance can handle transactions, but failover cannot occur until the HA pair is synchronized completely. After synchronization is complete, data is replicated in both appliances, and failover can be established.

When the active appliance becomes unavailable, a failover process is triggered. The standby appliance takes the active role and resumes the processing exactly where it was at the failure time.

During the takeover, the appliance that was originally the standby appliance attempts to reboot the appliance that failed. If the failed appliance can be rebooted, it takes over the role of standby appliance and starts synchronizing with the other appliance.

Details of the HA Pairs states are at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=/com.ibm.websphere.cast_iron.HAOverview.doc/HA_about_IAStates.html

For information about how to configure an HA environment using physical Integration Appliances, refer to 2.3, “Configuring the physical appliance for high availability” on page 44.

Determining the HA configuration that suits your needs

When determining which mode to use, keep in mind the following considerations:

- ▶ Active/active is the approach to use for the best performance or throughput.
- ▶ Active/hot standby provides the most reliability and minimizes the risk of losing data.
- ▶ Active/warm standby is preferred if your orchestrations do not allow parallel execution because of resource locking or the need for sequencing.
- ▶ Active/cold standby can be helpful from a licensing point-of-view, if limited resources are available.

6.1.3 Orchestration design for availability

The design of an orchestration has an effect on how it performs in a highly available environment. The main design feature for HA in an orchestration is to enable persistence. When persistence is enabled for an orchestration, most activities in the orchestration automatically record their state to a persistence store on the Integration Appliance. If the Integration Appliance running an orchestration fails and restarts, it restarts the orchestrations, and the Job Instances continue from the last persistent activity that was invoked by each Job Instance. With this configuration, no in-flight messages are lost.

Not all of the activities in an orchestration can support persistence. You can find further information about persistent activities at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=/com.ibm.websphere.cast_iron.doc/enabling_persistence.html

As an example for persistence and failover, assume you have a scenario where SAP sends an IDoc with 1000 customer records to the Integration Appliance. The Integration Appliance reads the IDoc and creates a loop to update each customer to a database. Figure 6-1 on page 282 illustrates this orchestration.

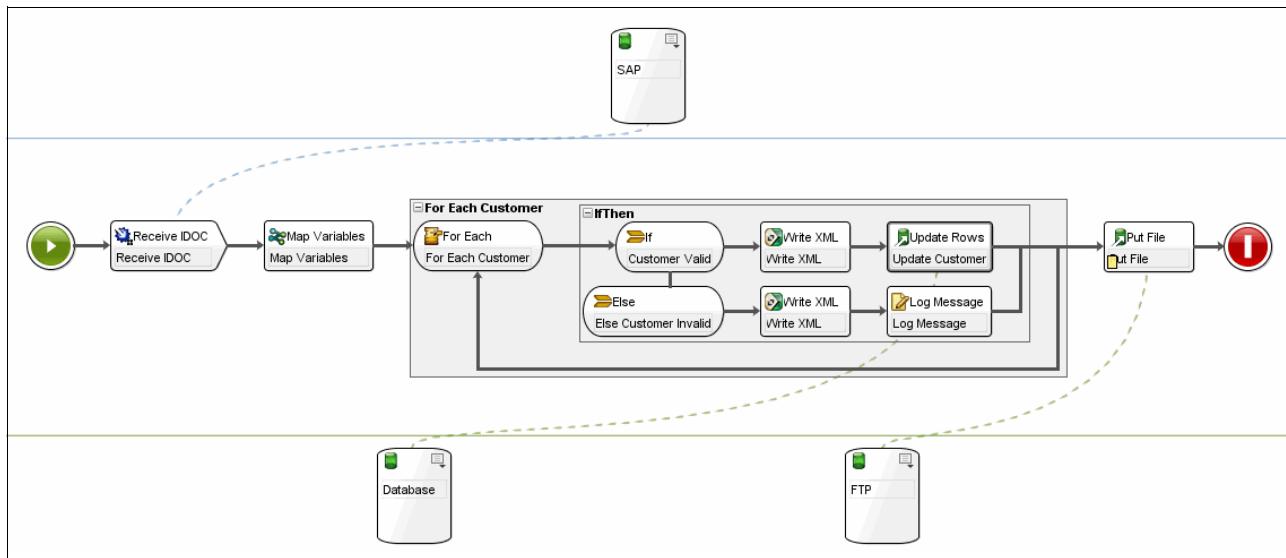


Figure 6-1 Example of a persistent scenario

If persistence for the orchestration is enabled or disabled, the following actions occur:

- ▶ If the Integration Appliance crashes just after reading the IDoc but before going into the loop:
 - If persistence is disabled, the IDoc request is gone.
 - If persistence is enabled, the orchestration restarts just after receiving the IDoc. In this case, all data is available.
- ▶ If the appliance crashes just after processing record 304 between Write XML and Update Customer:
 - If persistence is disabled, the records one to 303 were processed, and all further data is lost. Resending the IDoc means that 303 records are processed again, which can cause duplication in the database.
 - If persistence is enabled, the orchestration restarts at the last point of persistence, which is after the update just after record 303. All data is available, and the remaining customer records can be processed without any duplication.

The protocol used to start the Job Instance can also have an effect, for example, if a Job Instance was started using an HTTP Receive Request activity and the Integration Appliance fails, an HTTP error is sent back to the requester. This happens even if the orchestration is persistent. Thus, if the orchestration is persistent and the Job Instance restarts, the Job Instance can run to completion even though an error was sent back to the requester. Take care to design the orchestration to handle these sort of unexpected situations

6.2 Scalability techniques for integration appliances

Cast Iron Live: Scalability features are built into WebSphere Cast Iron Live.

With increased load you might encounter capacity problems, which can result in slow response times or loss of requests due to timeouts.

Scalability refers to the ability to increase processing capacity as the requirement for it grows. Planning for scalability includes selecting a platform that can grow dynamically, and investing in the additional resources (usually hardware) required for that growth. There are two approaches to scale a system. Vertical scalability refers to increasing the processing capability of the current system by adding CPUs, memory, hard disks, and so forth. Horizontal scalability refers to adding systems and spreading the application load across them.

Vertical scaling can be achieved on the Cast Iron Hypervisor Edition by scaling up the hardware that runs the hypervisor or by assigning more resources to the virtual appliance. Virtual appliances can also scale horizontally by starting multiple virtual appliances in parallel.

The physical Integration Appliance can be scaled horizontally, which means allowing one orchestration to run on multiple appliances simultaneously. However, because there is no replication between the machines, some considerations must be made with horizontal scaling.

Implementing horizontal scalability depends on the use of a load balancing mechanism to ensure the distribution of load across the appliances.

Figure 6-2 illustrates this concept.

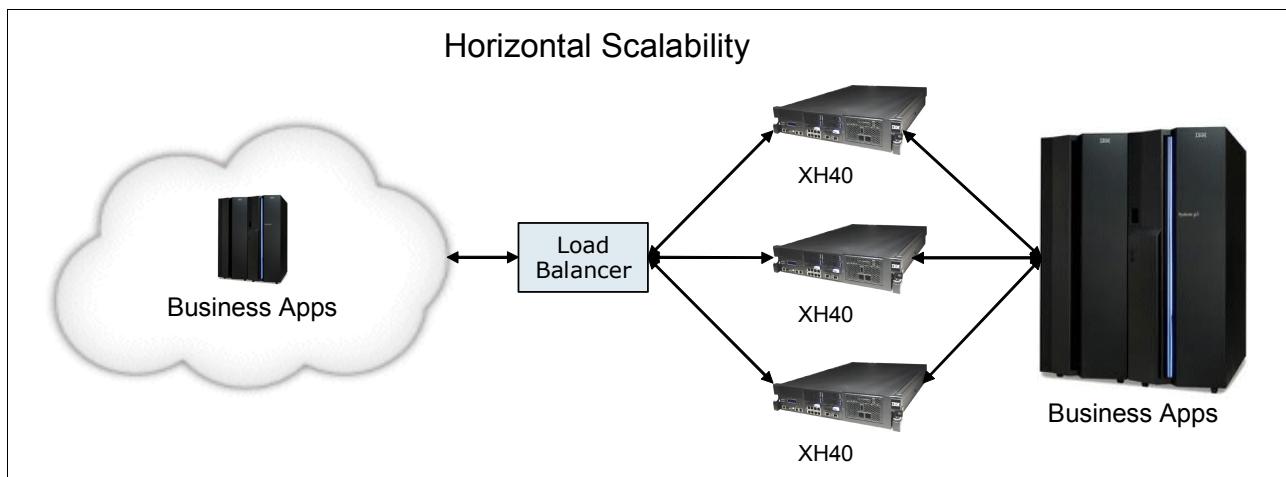


Figure 6-2 Horizontal scalability

The type of load balancing that can be implemented depends on the type of starter activity for the project configuration.

If data is pushed to Cast Iron (for example through an HTTP request), load balancing can be done by a load balancer.

If data is polled by Cast Iron and reading data is fast, you might consider having more than one orchestration listening to the same resource (file system, queue, server, and so forth); however, retrieving data takes a bit longer so this approach can cause conflicts. For example, if there is an FTP server with large files so that the retrieval of a file can take a minute or more, there is a risk that the same file can be read and processed by more than one orchestration. In this case, you need additional logic to do load balancing.

The following concepts apply to load balancing for orchestrations that are actively polling:

- ▶ Provide several sources for the orchestrations, for example JMS queues or FTP directories, so that separate orchestrations listen to different events.
- ▶ Provide different filter criteria for the poll activity, for example a file name pattern for an FTP poll, subject patterns for email polling, and so on.

Alternatively, if different appliances process load for several department or customers, consider replacing poll activities with scheduled jobs and separating the load by intelligent queries.

If these configurations for load balancing do not work, consider creating a dispatcher orchestration that acts as a load balancer. A dispatcher orchestration can take the requests and spray them in an intelligent way to the orchestrations in different appliances using round-robin or even more intelligent mechanisms, for example using the management API to query the number of current jobs.

Consider the following example. If you have an orchestration that polls an FTP server, you can create a dispatcher orchestration with a scheduler that retrieves a list of all files and then calls the orchestrations on separate appliances with different file names. You can do the same configuration, for example, with JMS or MQ, based on the message ID, with emails, and so forth. In addition, you can use a staging DB to store the status of assignments.

6.3 Maintenance and updates

In this chapter, maintenance describes the process of applying fixes to the appliance. These fixes can be, for example, fix packs for the appliance (these are provided by IBM) but can also be updates to orchestration running on the appliance.

Maintenance requires some downtime, for example, for a restart of the appliance after the application of a fix pack. If you can perform maintenance in a scheduled maintenance window, this makes the maintenance easier. But there might be business requirements that do not allow a downtime for maintenance.

A roll out of updates is the process of performing the maintenance on one appliance at a time. This is possible for applying fix packs but might not be possible for updates to the orchestration if these updates are major changes. For example if the inbound web services interface (Provide Service) of the orchestration changes in a way that makes the new version incompatible with the older one, a roll out update approach might cause problems.

Cast Iron Live: IBM performs maintenance of the appliance. You are responsible for updates to orchestrations.

6.3.1 Rolling out updates

If there are maintenance windows available, you can perform updates during that time. If no maintenance windows are available, make sure that you do a rollout update, meaning that you have a procedure in place that works, as follows:

1. Change the load balancing mechanism to make sure that the first appliance does not get any new load. New load is sent to the remaining Integration Appliances managed by the load balancer.

2. Stop the orchestrations on the first appliance so that they finish their current load. (You can stop an orchestration with an option to allow jobs to finish.)
3. Update the Integration Appliance.
4. Start the orchestrations and wait until all orchestrations are running.
5. Take the Integration Appliance back into the load balancing mechanism.
6. Repeat this process for the next appliance.

6.3.2 Scheduling down time

You can configure downtime rules in project configurations. These rules can stop or suspend all of the job instances for an orchestration.

You can configure rules to be triggered only once or to be repeated. Downtime rules are based by default on the GMT time zone of the system. So, be aware of time zone changes because downtime rules change too. You must be a user with administrative or publisher privileges to create downtime rules.

For information about scheduling down time, see 4.2.12, “Scheduling downtime” on page 194.

6.3.3 Minimizing downtime when updating an orchestration

You can create new versions of project configurations to minimize downtimes during upgrades. Creating a new version allows you to have the same orchestrations deployed simultaneously. You can stop one orchestration and start the other orchestration with minimal downtime.

If you instead deploy the same version, you must stop and undeploy the project configuration first before performing the update. This takes much longer than switching to a new version. In addition, by using a new version it is easy to switch back to the older version if you encounter any problem in the updated orchestration.

For further information about project configuration versions, refers to 3.3.4, “Versioning projects” on page 92.

6.4 Backup and restore

Backup and restore techniques are translated to export and import features in WebSphere Cast Iron. This section describes the two approaches for backup and restore that you can implement.

6.4.1 Using an image or a snapshot and restore for virtual appliances

Applicability: Using an image or a snapshot is a technique that can be used for virtual appliances. This is not applicable to physical appliances or to Cast Iron Live.

To create an image means to copy the virtual machine files at the file system level to another location. A snapshot is a mechanism provided by a hypervisor to take a snapshot of the

current runtime state of a virtual machine. The hypervisor allows you to move back to that snapshot later on.

Creating images while the system is running can be challenging or can require additional software, whereas snapshots can be done easily. Restoring an image or snapshot is easy but can be a challenge if the image or snapshot contains in-flight data (also known as *work in progress data*) that might no longer be valid and that can cause duplication or other issues.

Creating an image creates a large file because the entire virtual machine is saved. Creating a snapshot produces a smaller file but does not provide a real backup, as the snapshot is not saved on an external device. If the runtime image becomes corrupted at the file level, the snapshot might not be of use either. The image is not portable, meaning that you cannot use it to clone the environment. There are additional backup tools that are available for virtual machines to address disaster recovery.

6.4.2 Exporting or importing the Integration Appliance configuration

Applicability: Exporting or importing the configuration is available for both virtual appliances and physical appliances, but currently not supported for Cast Iron Live.

You can export the configuration of the appliance, including network settings, user settings, and projects, and then import the exported configuration into an existing or new Integration Appliance. During import, you can decide to import just the settings, just the projects, or both.

Use the export/import feature for backup and recovery or for migration purposes. This approach creates small images that are portable, meaning that you can use the exported configuration to create a clone for another environment.

Only users with administrative privileges can export and import repositories.

Exporting the configuration

These are the steps to export the projects and user settings of the Integration Appliance:

1. Log in to the WMC as a user with administrative privileges.
2. Click **Repository** → **Import/Export**, as shown in Figure 6-3.



Figure 6-3 Import/Export repository

- Click **Export**, and a message displays, as shown in Figure 6-4. In this message, click **Download Now**.

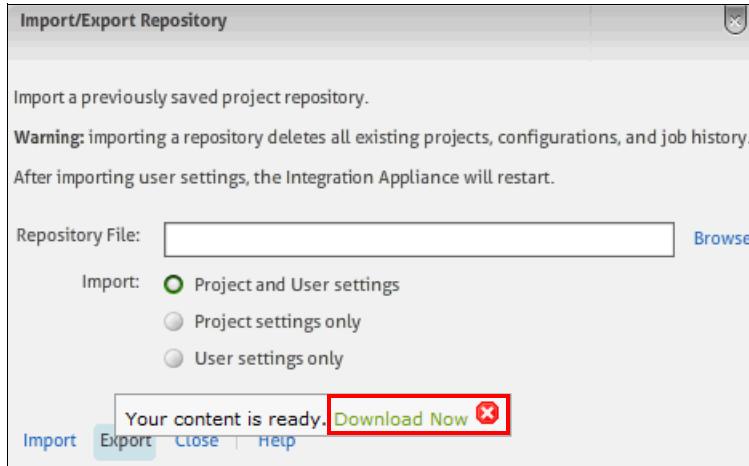


Figure 6-4 Export repository

A .tar.gz file is generated to be stored in a file system. This file contains two XML files:

- One file for user data (`user-data.xml`)
- One file for project data (`proj-data.xml`)

These files include data that refers to network configurations, users and groups, licenses, job log parameters, log levels, notifications, downtime rules, and passwords (system logs are not exported). These XML files are encrypted to protect the information. If you change the configuration of your Integration Appliance, we recommend exporting the current configuration before and after.

Virtual appliance considerations: Encryption of virtual appliance disks uses the VMware-assigned UUID of the virtual appliance. As a result, attempting to clone or copy a virtual appliance creates a non-functioning clone or copy because such operations create a new UUID.

To create a backup appliance, create a new a virtual appliance with the same configuration (for example standard or enhanced). Use the `config save/load` CLI command to export or import virtual appliance settings and WMC repository functions. Be aware, however, that these operations do not export or import any work in progress.

Also note that VMware VMotion provides a valid operation for migrating a virtual appliance to a new host. For information see:

<http://www.vmware.com/products/vmotion/overview.html>

For more information about exporting a virtual appliance repository, see:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=/com.ibm.websphere.cast_iron.VAuserguide.doc/VA_exportingVA.html

The exported system does not contain any installed third-party libraries. So, a package of assets that are required to recover an appliance must contain the following components:

- ▶ For a virtual appliance, a Cast Iron 6.1.x Hypervisor Edition image (A Cast Iron 6.1 version is already installed on a physical appliance)

- ▶ The fix pack of choice if not already installed
- ▶ Any installed third-party libraries
- ▶ The export file

Importing a configuration

To restore an exported repository into the Integration Appliance, import it using the following instructions:

1. Go to the Navigation page in the WMC.
2. Click **Repository** → **Import/Export**.
3. Click **Browse**, and select the .tar.gz file with the exported repository, and then select the settings to be imported, as shown in Figure 6-5.

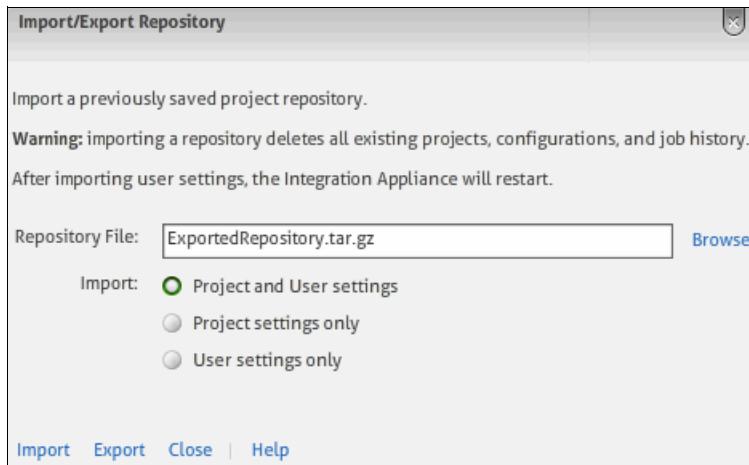


Figure 6-5 Settings to import

The available options to import a project repository, as shown in Figure 6-5, are:

- Project and User settings: Imports all the contents from the repository file.
- Project settings only: Imports only project information from the repository file.
- User settings only: Imports only user information from the repository file, such as network configurations, users and groups, licenses, job log parameters, log levels, notifications, downtime rules, and passwords.

Important: All of the projects in the Integration Appliance must be undeployed before you can import a project repository; otherwise, an error message displays.

Warning: When you import projects from a repository, the WMC deletes all existing projects, project configurations, and job history in the Integration Appliance. The system logs remain unaffected. To have the option to roll back to the older configuration, you must create an export of the older configuration before importing the new one. Import is supported from Cast Iron versions 3.7.1 to 6.x.

4. Click **Import**. The runtime imports the content of choice. After successful import, the appliance either restarts the runtime or reboots it. A message displays indicating that the Integration Appliance will be restarted or rebooted.

Depending on your browser, the file upload might initially fail due to certificate issues. If this happens, a warning message is displayed, similar to Figure 6-6, about the self-signed certificate that is used by default in the Integration Appliance. You can follow any of the options listed in the message, as shown in Figure 6-6. Click **OK**.



Figure 6-6 Import file failure

6.4.3 Recovering a physical or virtual appliance

To recover a physical or virtual appliance:

1. Install the new appliance.
2. Update the appliance to the fix pack of choice.
3. Install any required third-party libraries.
4. Import the export file.
5. Install the remaining projects (if the remaining projects are not in the export file or if you want to import without projects).

Tips:

- Be sure to test the recovery approach before going into production.
- To reset an Integration Appliance use the **system clean all** command.

For more information about the system clean command see:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=%2Fcom.ibm.websphere.cast_iron.cli.doc%2FCLI_system_commands.html

You can also perform an import and export using the command-line interface (CLI) or the Management API:

- The CLI uses an FTP server to upload or download the project repository file. For more details, refer to:
http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.cli.doc/CLI_config_commands.html
- The Management API uses a deployment web service interface that handles project repositories in XML format. This interface is preferred to create scheduled backup/restore scripts. For further information, refer to:
http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp?topic=/com.ibm.websphere.cast_iron.api.doc/ci00092.html

6.4.4 Recovering projects from the WMC

If you lose the source of a project but have the project uploaded to the runtime, you can recover projects from the WMC and import those projects back to a new Studio machine. When a project is published to the Integration Appliance, the source files of the project are also uploaded. You can recover these files and take them back to Studio.

To recover the project configurations:

1. Log in to the WMC as an administrative user or publisher.
2. In the Home page, select the Project Configuration to download, as shown in Figure 6-7.

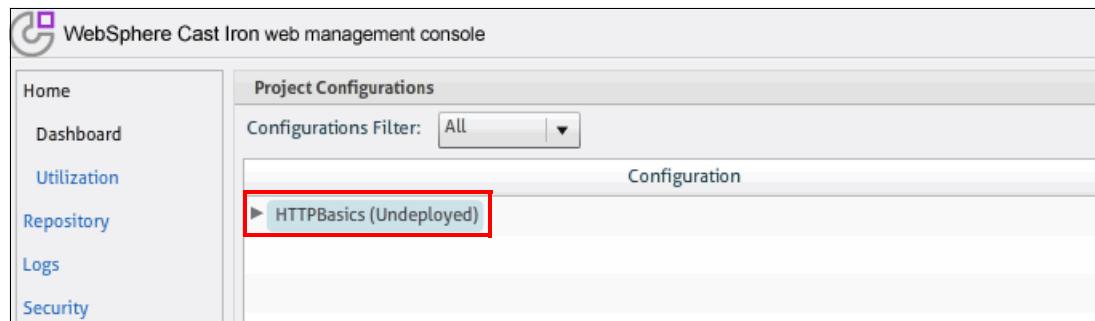


Figure 6-7 HTTPBasics project configuration

3. In the Configuration Details pane, click **Download**, as shown in Figure 6-8.

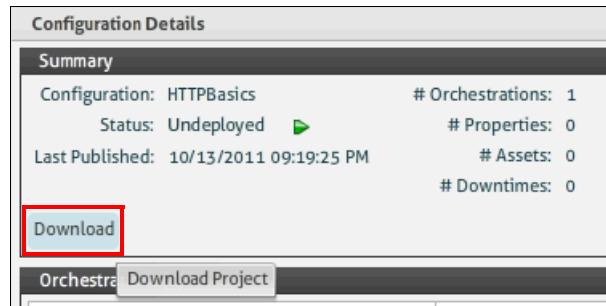


Figure 6-8 Download configuration project

4. Click **Download Now**, as shown in Figure 6-9.

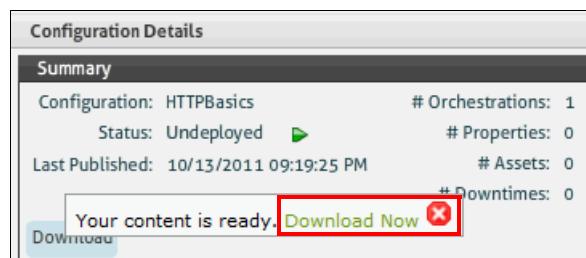


Figure 6-9 Download Now

5. A .zip file is generated. Save this file, and then extract the contents of the compressed file.
6. Open the Studio tool, and click **Open Project**, as shown in Figure 6-10 on page 291.

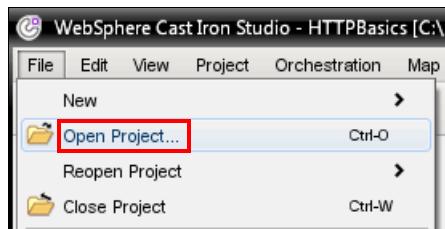


Figure 6-10 Open project in Studio

7. Select the .sp3 file in the extracted folder, and click **Open**, as shown in Figure 6-11.

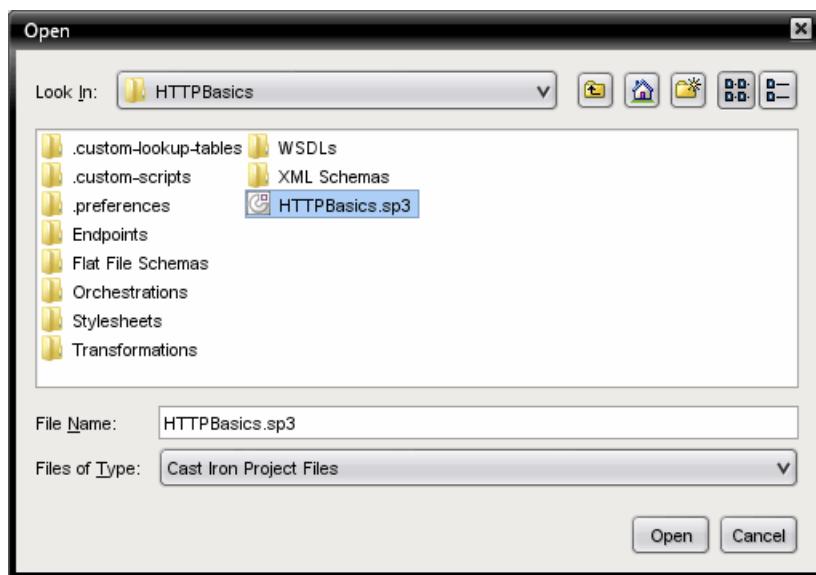


Figure 6-11 Open the .sp3 file

Figure 6-12 shows the project configuration recovered from the WMC.

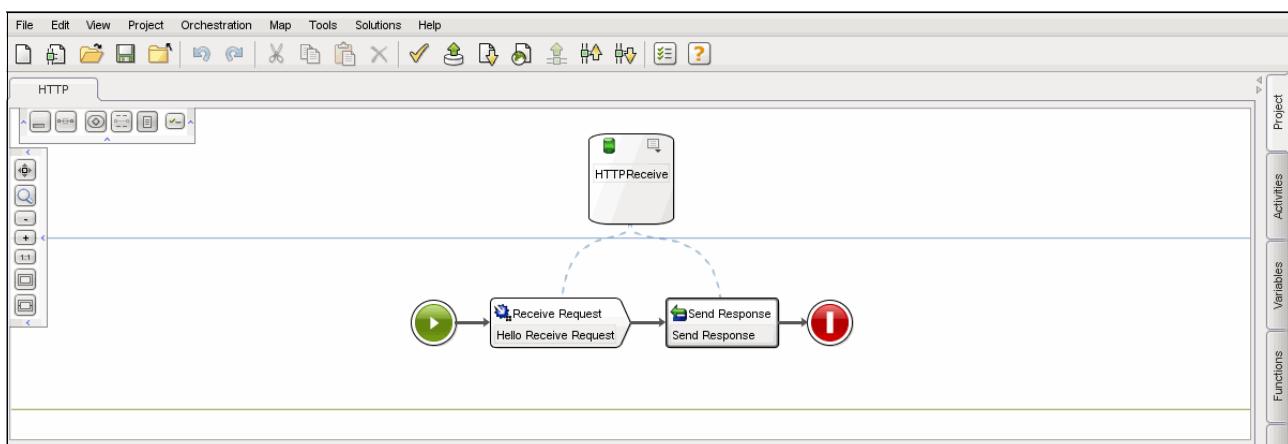


Figure 6-12 Project configuration recovered



Reusability with Template Integration Projects

Reusability is a key factor in the efficiency and ease-of-use for any product. In WebSphere Cast Iron, *Template Integration Projects (TIPs)* provide a way to store and reuse common orchestrations. TIPs are kept in a central repository, letting orchestration developers take advantage of the development efforts of many others. This feature provides a path for fast and easy integration development.

This chapter includes the following topics:

- ▶ Overview of Template Integration Projects
- ▶ Introduction to the TIP Configuration Editor
- ▶ Creating and modifying TIPs
- ▶ Uploading TIPs
- ▶ Searching and downloading TIPs
- ▶ Verifying the TIP
- ▶ Rating and reviewing TIPs

7.1 Overview of Template Integration Projects

TIPs are WebSphere Cast Iron Studio (referred to as *Studio*) projects that are built following preferred practices. These projects depict common integration patterns that can be deployed readily with minimal changes. After such a project is complete and tested, you can create a TIP from the project. Creating a TIP involves using the TIP Configuration Editor in Studio to create a series of configuration steps. These steps identify the variable points in the project that must be configured before using the TIP. When a TIP is used as the basis of a new project, the TIP Configuration Wizard in Studio takes the user of the TIP through these steps so the TIP can be customized for their orchestrations.

Users can store TIPs in the Cast Iron solutions repository. By the same token, users can search the repository and download TIPs for use. Each user of a TIP has the option of rating it, providing information to those searching TIPs for use.

TIPs that are uploaded are monitored by IBM and those depicting common use cases or scenarios are tested and IBM certified, guaranteeing that the solution provided by the TIP works as expected. TIPs that are IBM certified cannot be deleted from the Cast Iron solutions repository.

Figure 7-1 gives a brief overview of the creation and use of TIPs.

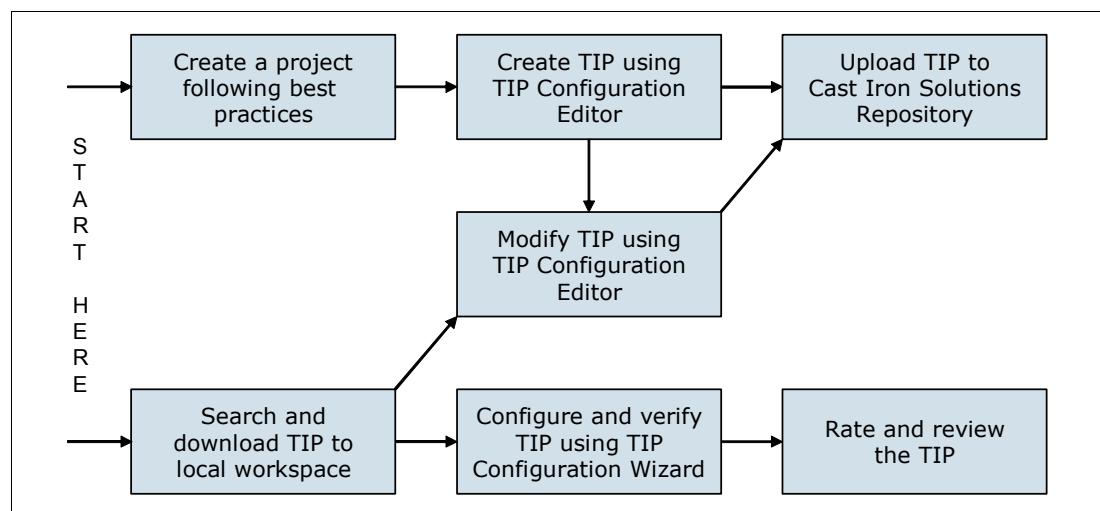


Figure 7-1 Overview of TIPs creation and use

Prerequisites

The following prerequisites are required to work with TIPs in the repository:

1. Make sure you have a Cast Iron community ID. If you do not have a community ID, you can request one at:
<http://community.castiron.com>
2. Click the **Preferences** icon in the Studio tool bar.

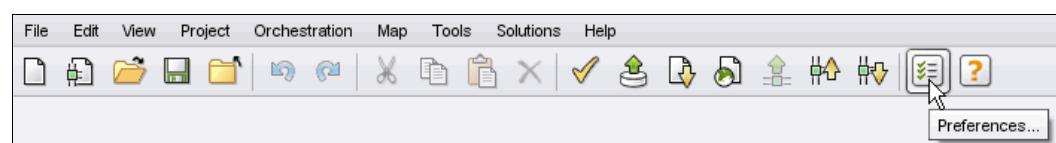


Figure 7-2 Preferences

3. Select **Session Login**, and enter your community user name, password, and the URL of the repository, as shown in Figure 7-3. Click **OK** at the bottom of the preferences window.

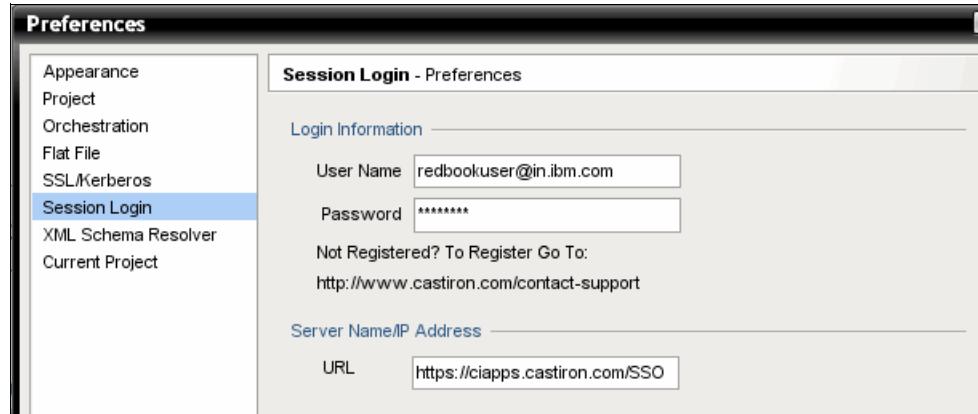


Figure 7-3 Session login details

4. Log in to the community by clicking **Login** in the status bar, as shown in Figure 7-4.

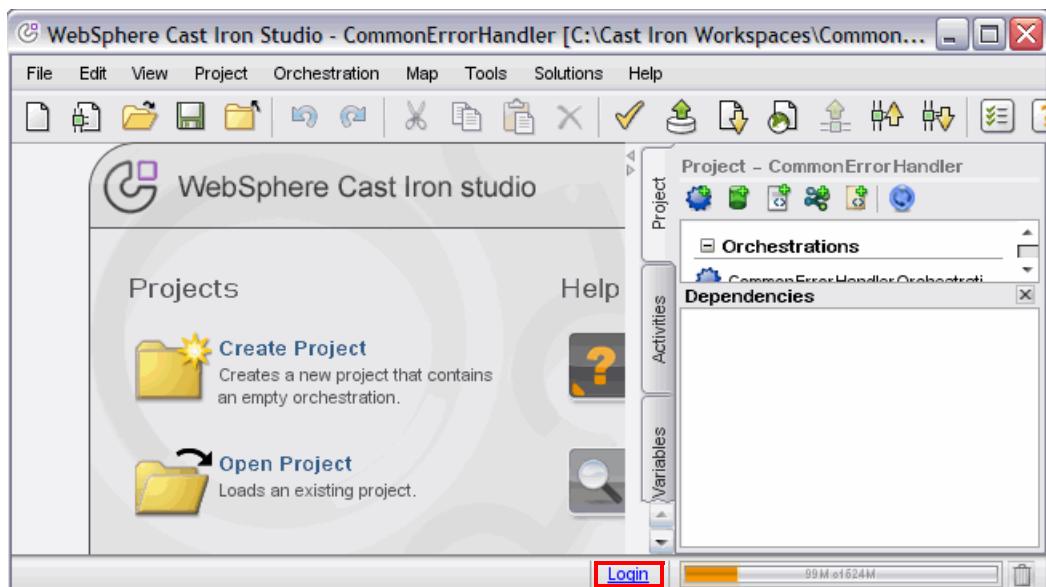


Figure 7-4 Login option

If your system has firewall protection installed, make sure to look for a prompt to allow the access.

5. The login uses the community user name and password you specified in the preferences, as shown in Figure 7-5 on page 296. Click **OK**.



Figure 7-5 Community login

7.2 Introduction to the TIP Configuration Editor

Using the TIP Configuration Editor you can create or modify the TIP's Configuration Wizard steps.

You can launch the Configuration Editor from the Studio Menu:

1. Open the project.
2. Select **Tools** → **Create Configuration Editor** or **Edit Configuration Editor**. When the editor opens, Figure 7-6, the Steps panel (on the left) and the Workspace panel display. The sections that follow describe these panels.

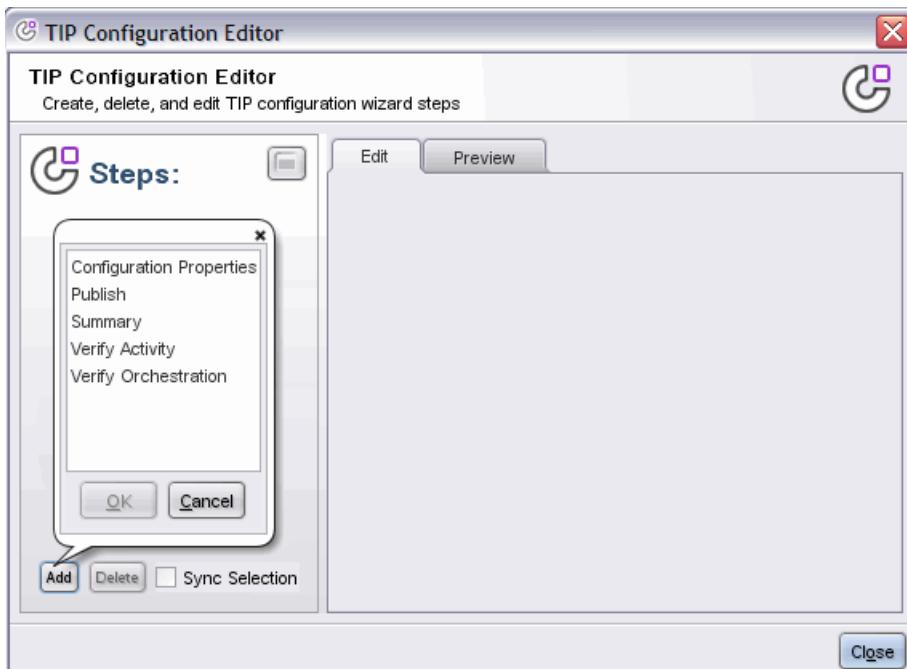


Figure 7-6 TIP Configuration Editor

7.2.1 Steps panel

The Steps panel lists the configuration wizard steps. Clicking **Add** in the Steps panel displays a list of wizard step types. To add a step, select the step type, and click **OK**.

The step types are:

- ▶ Configuration Properties

Specifies the configuration properties defined for the project (see “Creating configuration properties” on page 88) that you need to change as part of the initial configuration. In this step, you can add a subset of the configuration properties in the project that must have values provided before using the downloaded TIP.

- ▶ Publish

Specifies the steps required to publish the TIP to any form factor.

- ▶ Summary

Gives a brief overview of the scenario covered as part of the TIP.

- ▶ Verify Activity

Verifies the selected activity. Ideally this activity interacts with an endpoint, for example to test FTP connection.

- ▶ Verify Orchestration

This step tests the orchestration. You can start the orchestration within the step to verify that it works and provides the results that you expect.

When creating or modifying a TIP, you can change the order of the steps in the wizard by dragging and dropping the steps into the required position.

The TIP Configuration Editor has the following views:

- ▶ The *compact view* shows only the Steps panel, which provides a clear overview of the steps in the TIP.
- ▶ The *full view* shows the Steps panel and the Workspace panel. In the Workspace panel, you can view the details for the selected step in the Steps panel.

You can toggle between the views by clicking the  icon in the upper-right corner of the Steps panel.

When you open the TIP configuration editor, the orchestration is still open in the background in Studio. When you select the **Sync Selection** option at the lower-right corner of the Steps panel, the Studio automatically navigates to the entity in the orchestration or configuration page corresponding to the step in the TIP Configuration Editor that you are editing.

7.2.2 Workspace panel

The Workspace panel consists of the following tabs:

- ▶ The Edit tab is used to design and build each step of the TIP. Note that you can drag images directly into the Description and Summary Content fields for illustration. Selecting the **Show Markup** option at the bottom of the tab allows you to edit the underlying HTML source for the page. For example, you can add HTML tags (bold, italics, bullets, and so forth) to better convey the description, which can include instructions to perform as part of the current step.

- ▶ The Preview tab is used to view the actual page of the current TIP Configuration Wizard step. On this page, you can review the changes made in the Edit tab.

7.3 Creating and modifying TIPs

You must create TIPs from valid projects that were built using preferred practices. Ideally, the project addresses a useful, generic scenario.

The following steps show how to create a TIP for the common error handler project, shown in Figure 7-7, and discussed in Chapter 9, “Common error handlers” on page 349.

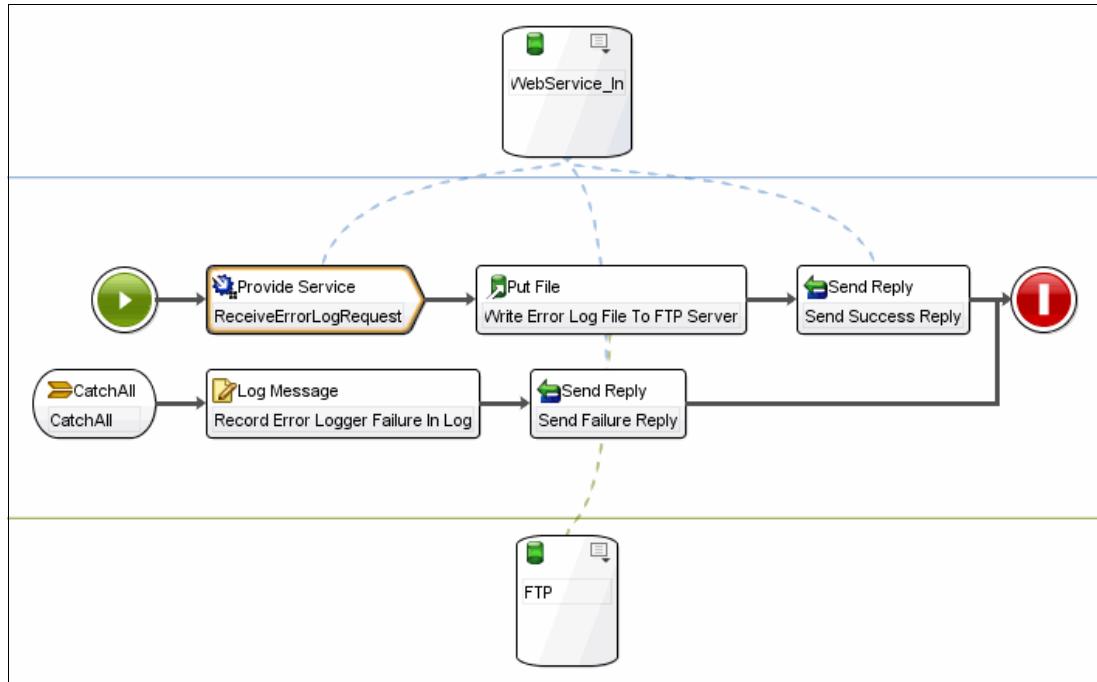


Figure 7-7 Common error handler project

1. Open the CommonErrorHandler project in Studio.
2. Launch the TIP Configuration Editor to create a TIP by selecting **Tools** → **Create Configuration Wizard**.
3. Add a Summary Step that gives an overview of the scenario (Figure 7-8 on page 299):
 - a. In the Steps panel, click **Add**. Select **Summary**, and then click **OK**.
 - b. Rename the step by typing **Introduction** into the Name field.
 - c. Select the **Show Markup** field so that you see the HTML markup in the Description and Summary Content areas.
 - d. Enter a description of the TIP in the Summary Content field, as shown in Figure 7-8 on page 299. To see how your page will look when the TIP configuration wizard is executed, click the **Preview** tab.

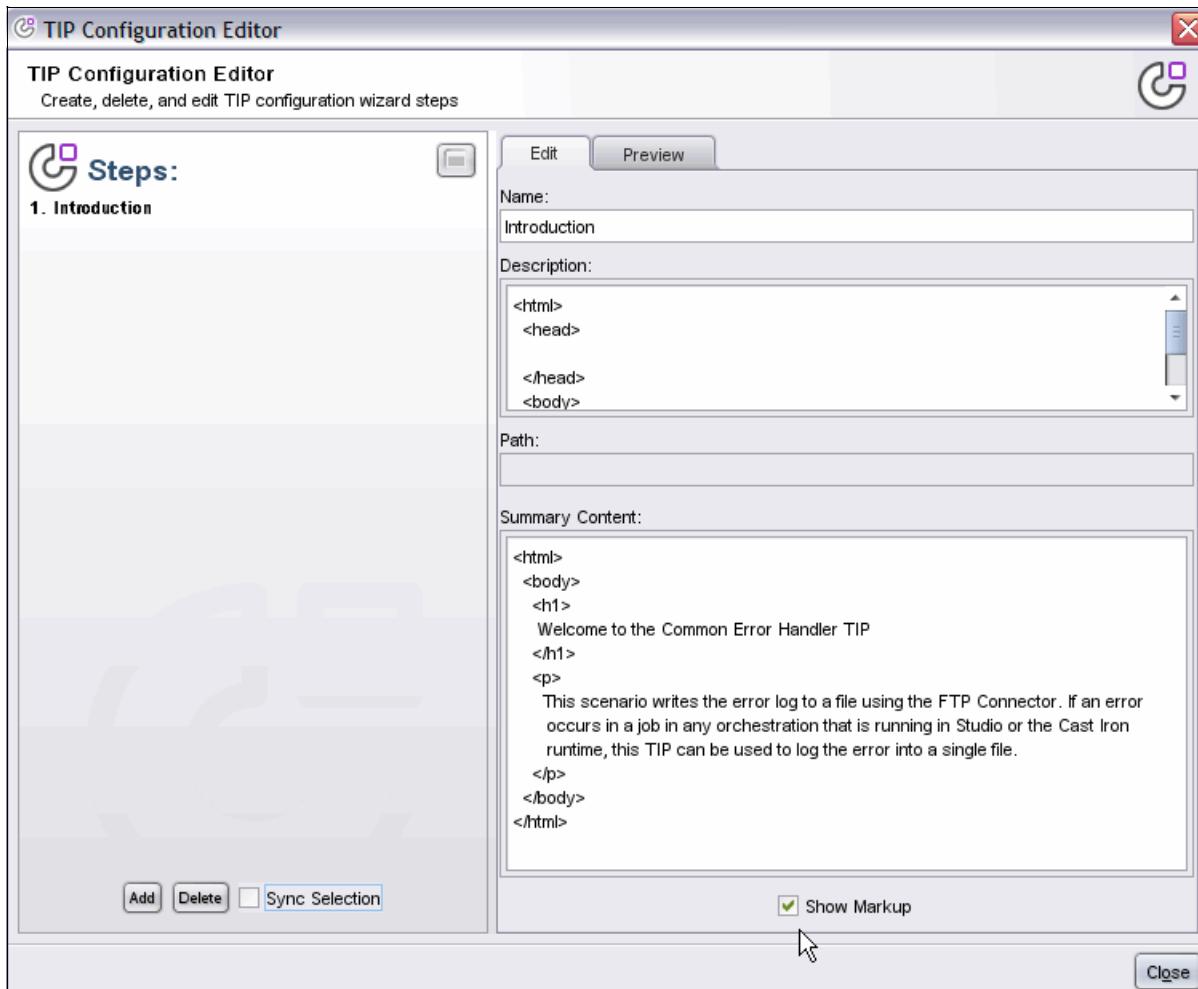


Figure 7-8 Introduction step

4. Add the Configuration Properties step to list the configuration properties that must be given a value by the user of the TIP.
 - a. In the Steps panel, click **Add**. Select **Configuration Properties**, and then click **OK**.
 - b. Click the **Preview** tab to review the configuration properties for the project.
 - c. Switch back to the Edit tab and select the **Show Markup** field so that you see the HTML markup in the Description area.
 - d. Enter the instructions for completing the configuration properties as a bulleted list using HTML tags in the Description field.

If the Restrict Properties To section is empty, all the configuration properties of the project are added to this step. You can list just the properties to be configured by adding those properties to the Restrict Properties To section, using the **Add** button. You must configure all the properties of the CommonErrorHandler project; therefore, leave the Restrict Properties To area empty, as shown in Figure 7-9 on page 300. To check how your page will look, click the **Preview** tab.

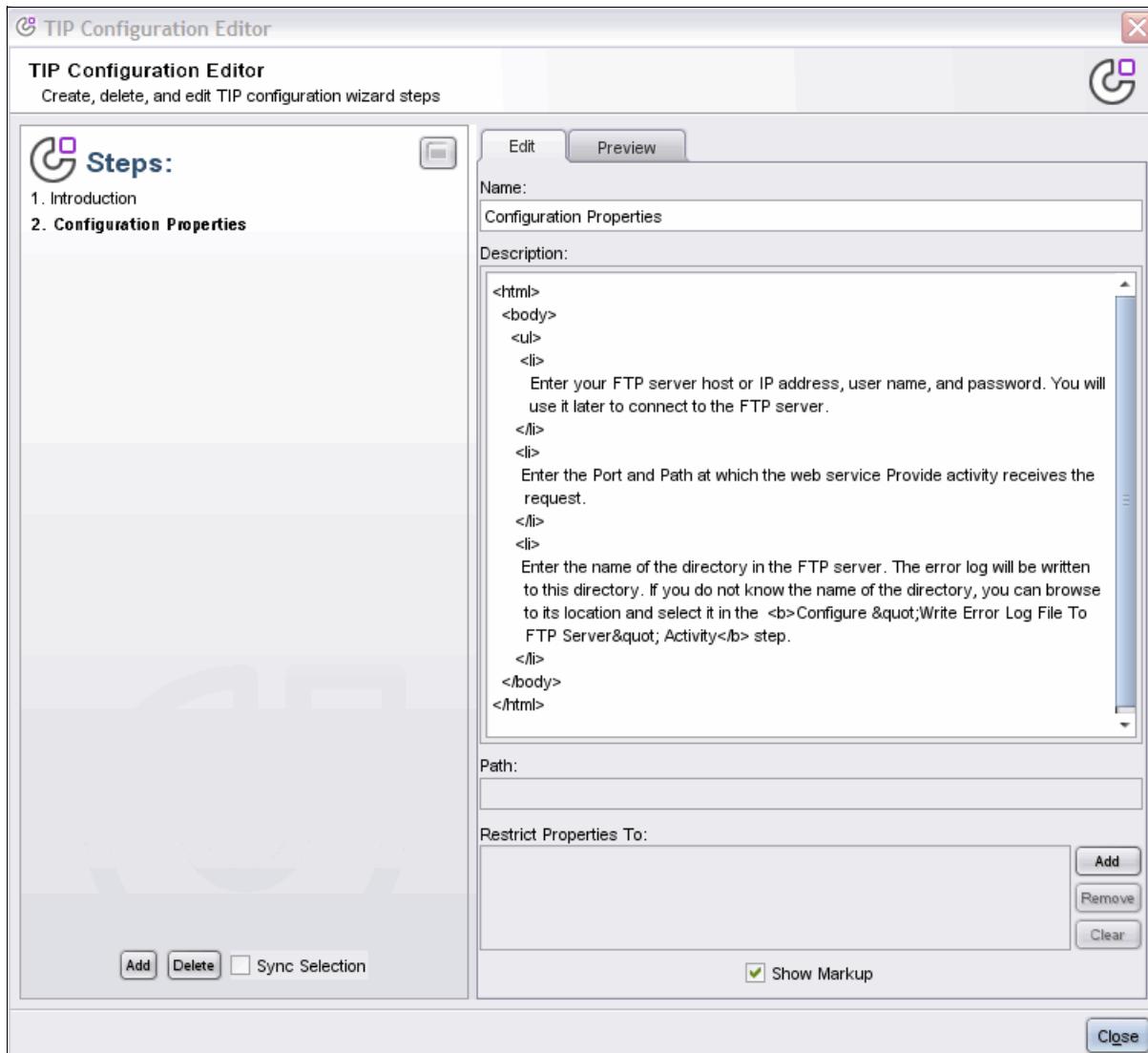


Figure 7-9 Configuration Properties step

5. Add a step to verify the connection to the FTP server:
 - a. Drag the **FTP Endpoint** from the Project tab, Figure 7-10 on page 301, to the Steps panel to add a step with the name of the endpoint, FTP.

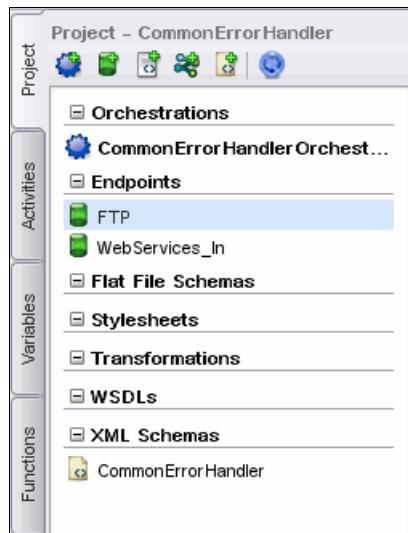


Figure 7-10 Select the FTP endpoint in the Project tab to drag to the steps

- b. Rename the step by typing Verify FTP Connection in the Name field.
- c. Select the **Show Markup** field so that you see the HTML markup in the Description area.
- d. Enter the instructions for this step in the Description field, shown in Figure 7-11.

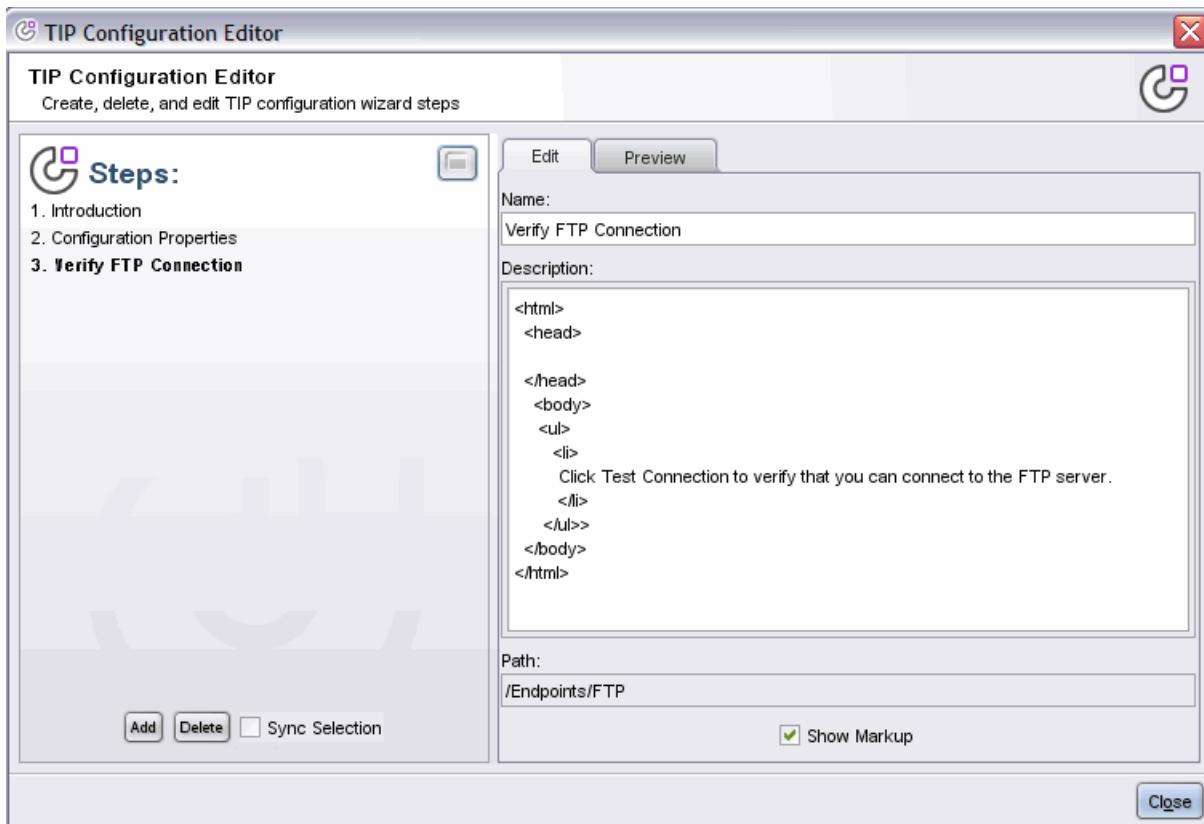


Figure 7-11 Verify FTP Connection step

6. Add a step to configure the activity that writes the error log to a file:
- Drag the Configure task from the checklist of the “Write Error Log File To FTP Server” activity in the orchestration to the Steps panel to add a step with the name Configure, as shown in Figure 7-12.

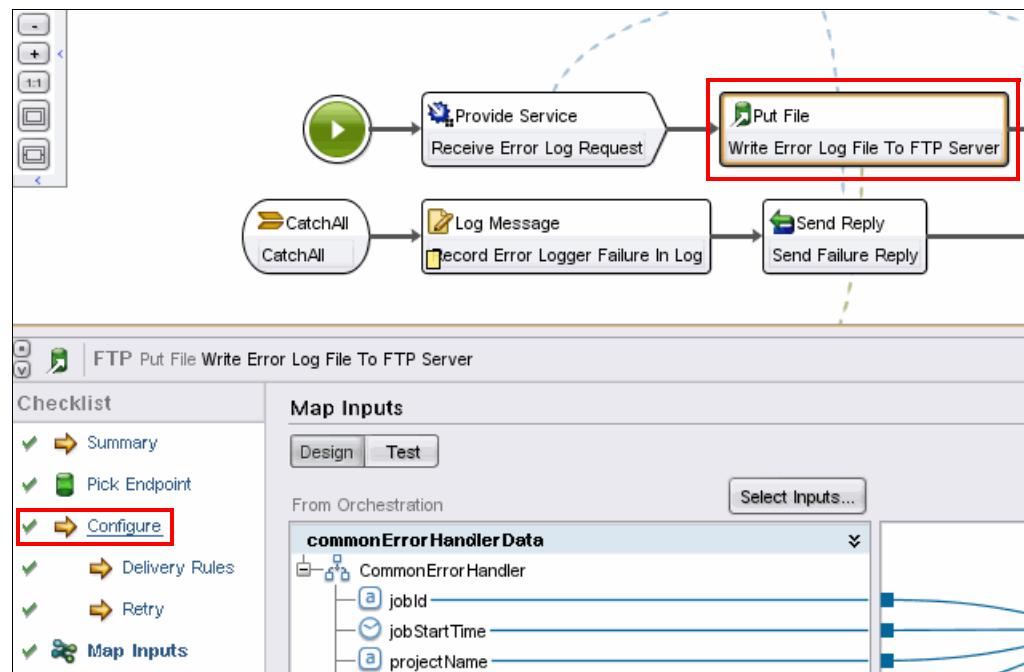


Figure 7-12 Select Configure of the activity to drag to the steps

- Rename the step by typing Configure “Write Error Log File To FTP Server” Activity in Name field.
- Select the **Show Markup** field so that you see the HTML markup in the Description area.
- Enter the instructions in the Description field, as shown in Figure 7-13 on page 303.

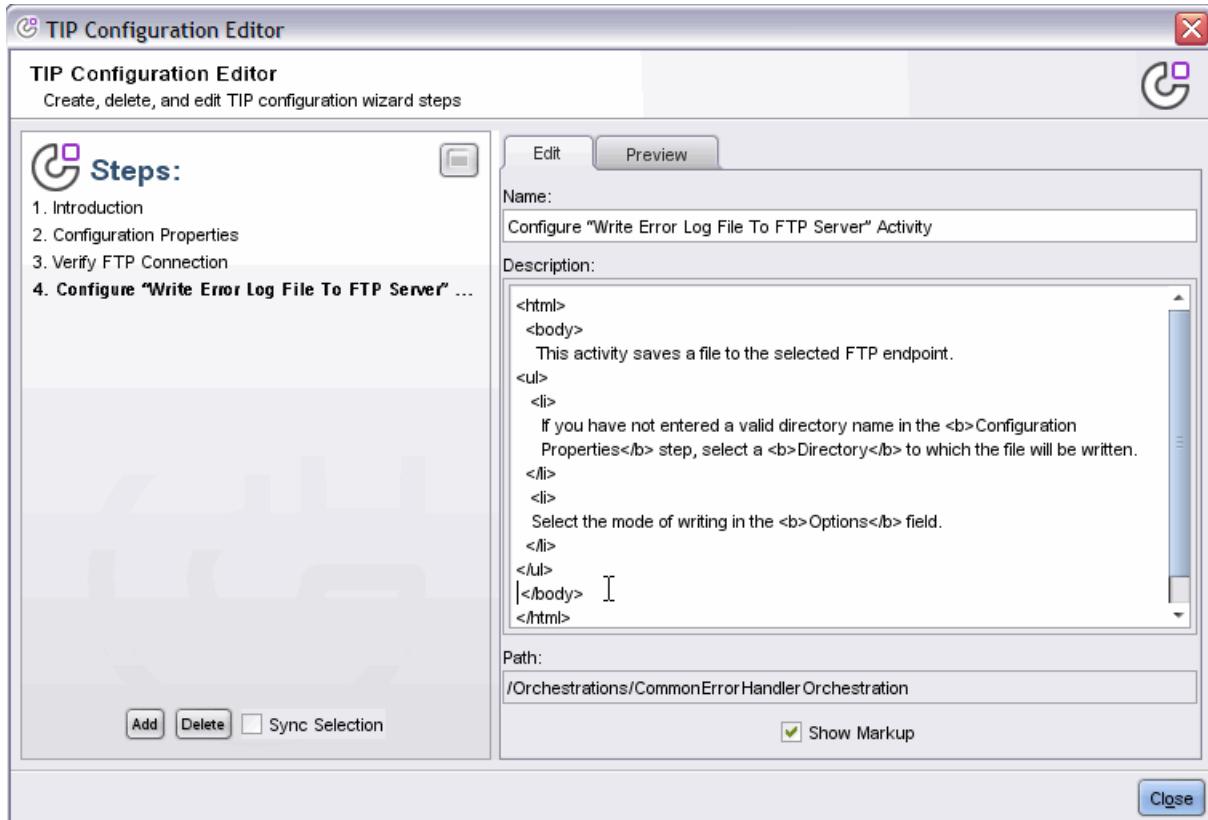


Figure 7-13 Configure "Write Error Log File To FTP Server" Activity step

7. Optionally, add a step to modify the mapping of the "Write Error Log File To FTP Server" activity. In this step, you can restrict the fields written to file by the mapping only the required fields:
 - a. Drag the Map Inputs task from the checklist of the "Write Error Log File To FTP Server" activity (Figure 7-14 on page 304) to the Steps panel (Figure 7-15 on page 304) to add a step with the name Map Inputs.

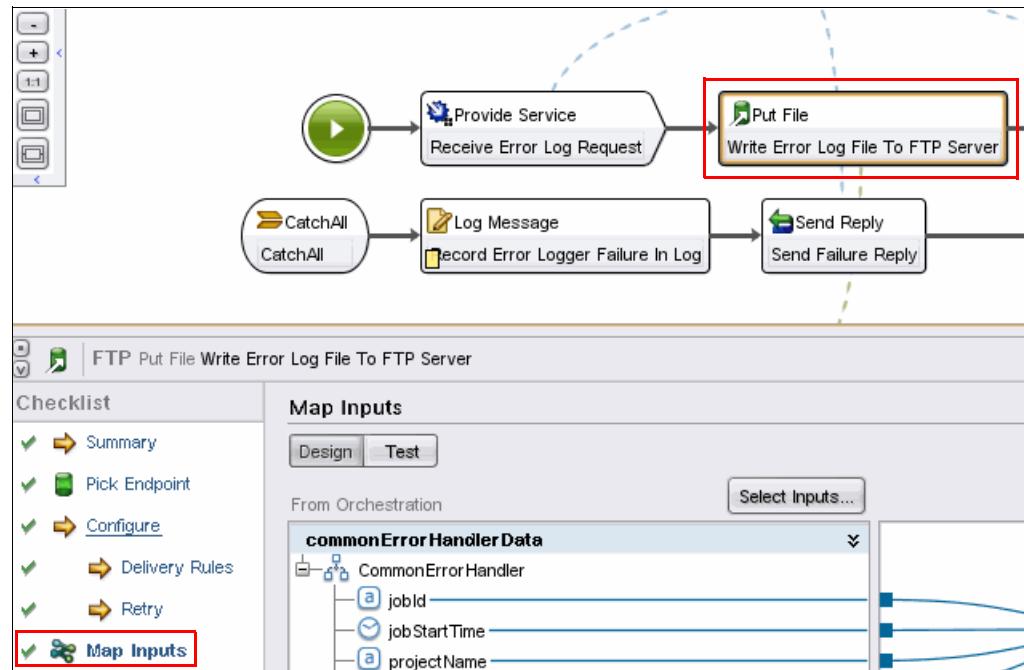


Figure 7-14 Select Map Inputs of the activity to drag to the steps

- Rename the step by typing Map Error Data fields in the Name field.
- Enter the instructions in the Description field, as shown in Figure 7-15.

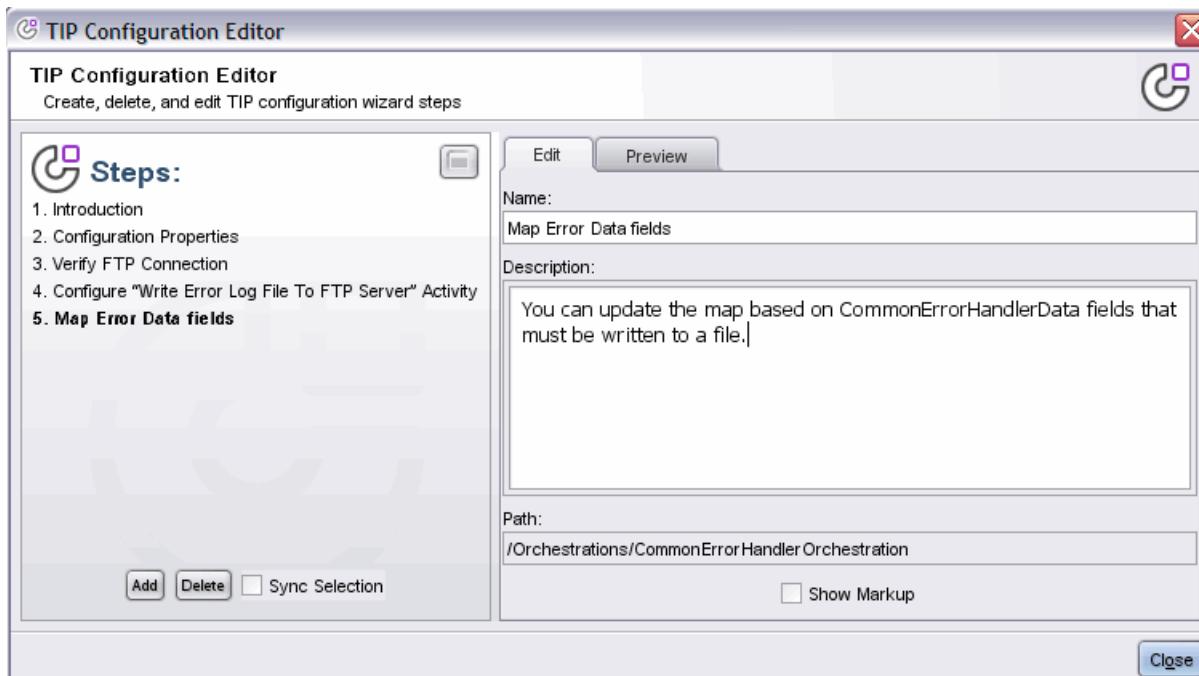


Figure 7-15 Map Error Data fields step

- Add a Verify Orchestration Step to verify the integration:
 - In the Steps panel, click **Add**. Select **Verify Orchestration**, and then click **OK**.
 - Rename the step by typing Verify Integration into the Name field.

- c. Select the **Show Markup** field so that you see the HTML markup in the Description area.
- d. Enter the instructions in the Description field, as shown in Figure 7-16.

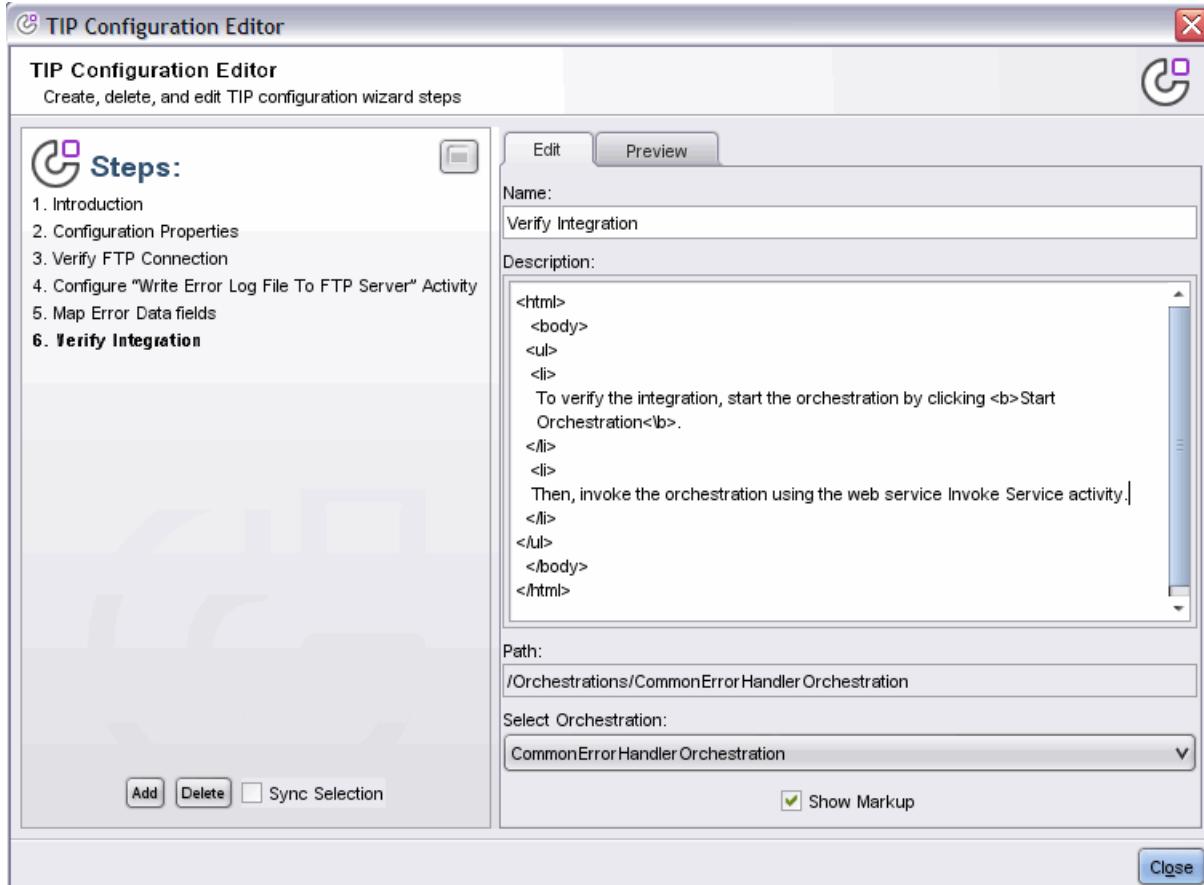


Figure 7-16 Verify Integration Step

9. Click **Close**, and save the project to save the TIP changes.

You can modify this TIP at any time by launching the TIP Configuration Editor again.

Remember:

- ▶ Only the creator of a TIP can modify the TIP.
- ▶ After creating or modifying a TIP, be sure to save the changes before closing the project. If you close the project without saving, your changes to the TIP are not saved.
- ▶ The Finished step is added as the last step by default. This step contains a page that is used to rate the TIP. The step is not visible in the TIP Configuration Editor.

7.4 Uploading TIPs

You can upload TIPs to the Cast Iron solutions repository using a 3-step wizard, which asks you details about the TIP, the scenario and the endpoints that are used. This section illustrates the steps to upload a TIP using the CommonErrorHandler TIP as an example.

To upload TIPs:

1. In Studio, login to Cast Iron community. See “Prerequisites” on page 294.
2. Launch the Upload Project Wizard in the **Studio** menu by selecting **Solutions → Upload Project to Repository**. If you are not logged in, you will not have this option in the menu.
3. The Upload Project Step 1 of 3 window opens, as shown in Figure 7-17.



Figure 7-17 Step 1: Common error handler TIP upload

Enter the summary information listed in Table 7-1, and then click **Next**.

Table 7-1 Summary information

Field	Description
Name	Enter name of the TIP
Path	Specify the path in the Cast Iron solutions repository where Studio will upload the TIP. If the path you enter does not exist, it will be created. For example, /RedBooks/MyTips
Category	Categorize the TIP to any of the following types: <ul style="list-style-type: none">► Usecases► Best_Practices► Utilities► General
Description	Give a brief overview of the TIP

4. The Upload Project Step 2 of 3 window opens, as shown in Figure 7-18 on page 307.

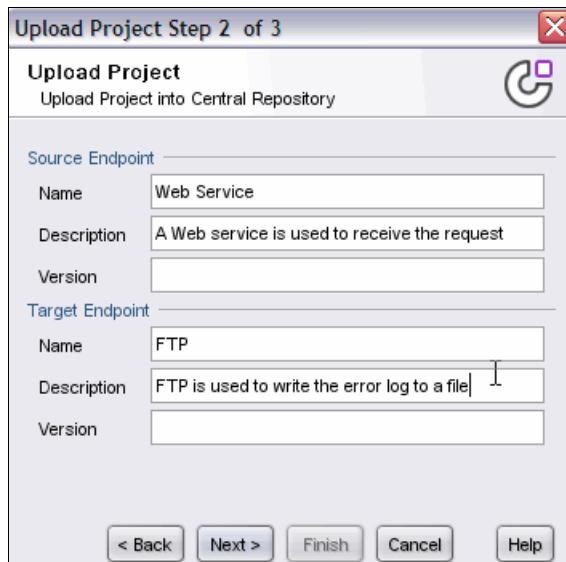


Figure 7-18 Step 2: Common error handler TIP upload

Enter the source and target endpoint information listed in Table 7-2, and then click **Next**.

Table 7-2 Source and target endpoint

Field	Description
Name	Enter the name of the source and target Endpoints respectively
Description	Give a brief description of the source and target endpoints respectively
Version	Enter the version number of source and target endpoints' configuration respectively

5. The Upload Project Step 3 of 3 window opens, as shown in Figure 7-19.

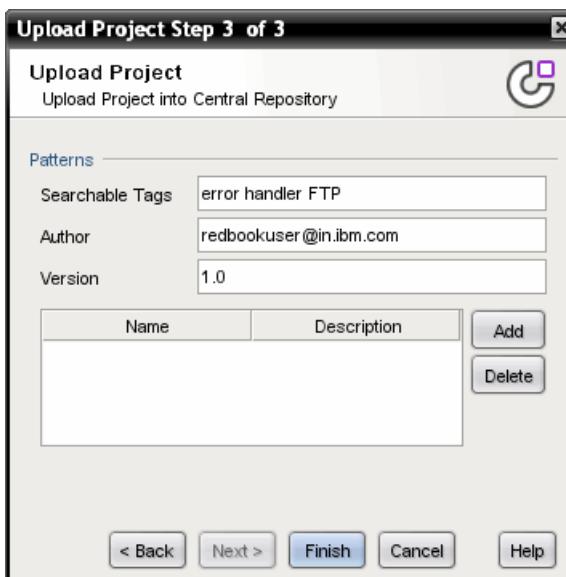


Figure 7-19 Step 3: Common error handler TIP upload

Enter the information about the project listed in Table 7-3.

Table 7-3 TIP pattern details

Field	Description
Searchable Tags	Specify the tags that can be used to search this TIP, for example the name of the endpoints used, the name of the object used, and so forth.
Author	Enter the name of the person or entity who created the project.
Version	Specify the version of the TIP that you are uploading to the Cast Iron solutions repository.
Name/Value	Name/value pairs are used for searching for the TIP in filter mode. <ul style="list-style-type: none">▶ Click Add to add a pattern row where you can enter the value for respective columns.▶ Select the pattern row, and click Delete to delete it.

6. Click **Finish**. You are prompted to accept a license agreement, as shown in Figure 7-20. Read the license agreement, and then select the **I have read and accept this license agreement** option. Click **OK**, and the TIP is uploaded to the Cast Iron solutions repository.

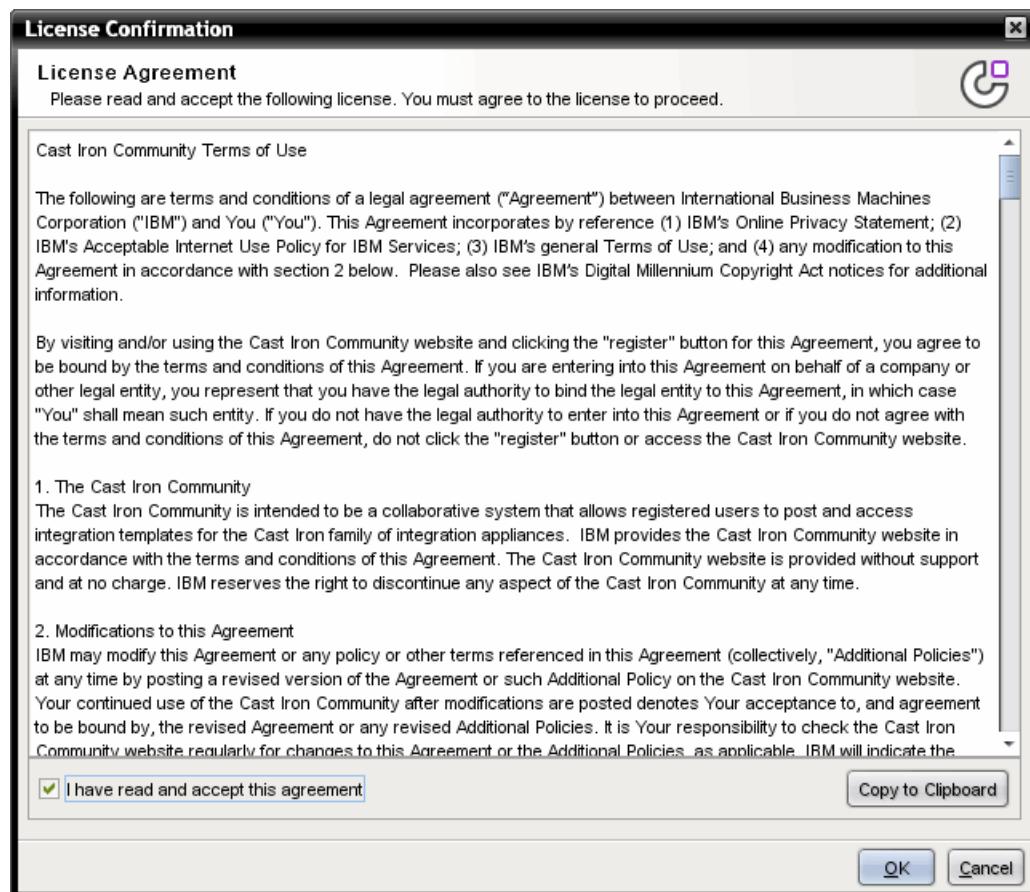


Figure 7-20 License agreement

7.5 Searching and downloading TIPs

You can download and use the TIPs from the Cast Iron solution repository if you have Cast Iron community access. A TIP is downloaded as a new project.

Studio provides an option to search and download TIPs. The following example searches for and downloads the CommonErrorHandler TIP that you uploaded in 7.4, “Uploading TIPs” on page 305:

1. In Studio, select **Solutions** → **Search For TIPs**. The TIP is downloaded as a new project.
2. If you have not already logged in to the Cast Iron community, you are prompted to log in. Enter the proper login credentials, and then click **OK**.
3. In the Search window, shown in Figure 7-21, you can enter the criteria for your search.

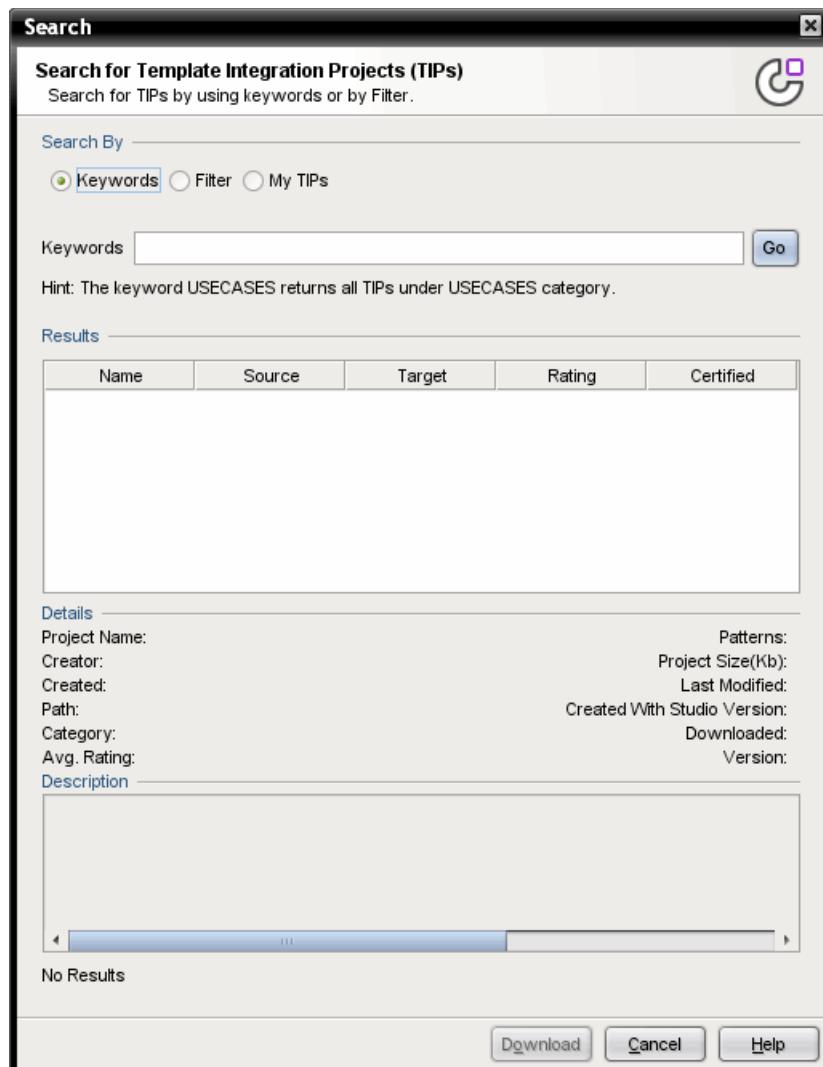


Figure 7-21 Search for TIPs

This window includes the following sections:

– Search By

Specify the condition for searching the TIPs:

- Keywords: Specify any keyword as a search condition.
- Filter: Enter information to be used as a the search condition, as shown in Figure 7-22.



Figure 7-22 Search TIPs by Filter

- My TIPs: List the TIPs that you have created.
- Results: List the TIPs that match the search criteria.
- Details: Provides the metadata for the TIP as shown in Figure 7-21 on page 309.
- Description: Gives a brief overview of the integration scenario covered in the selected TIP.

In the keywords field, enter the text error handler, as shown in Figure 7-23 on page 311, and then click **Go**. Notice that the text is part of the searchable tags entered in Figure 7-19 on page 307. The result is a list of TIPs matching the search condition. You can view the rating of a TIP (entered by previous users), and you can see which TIPs are IBM certified.

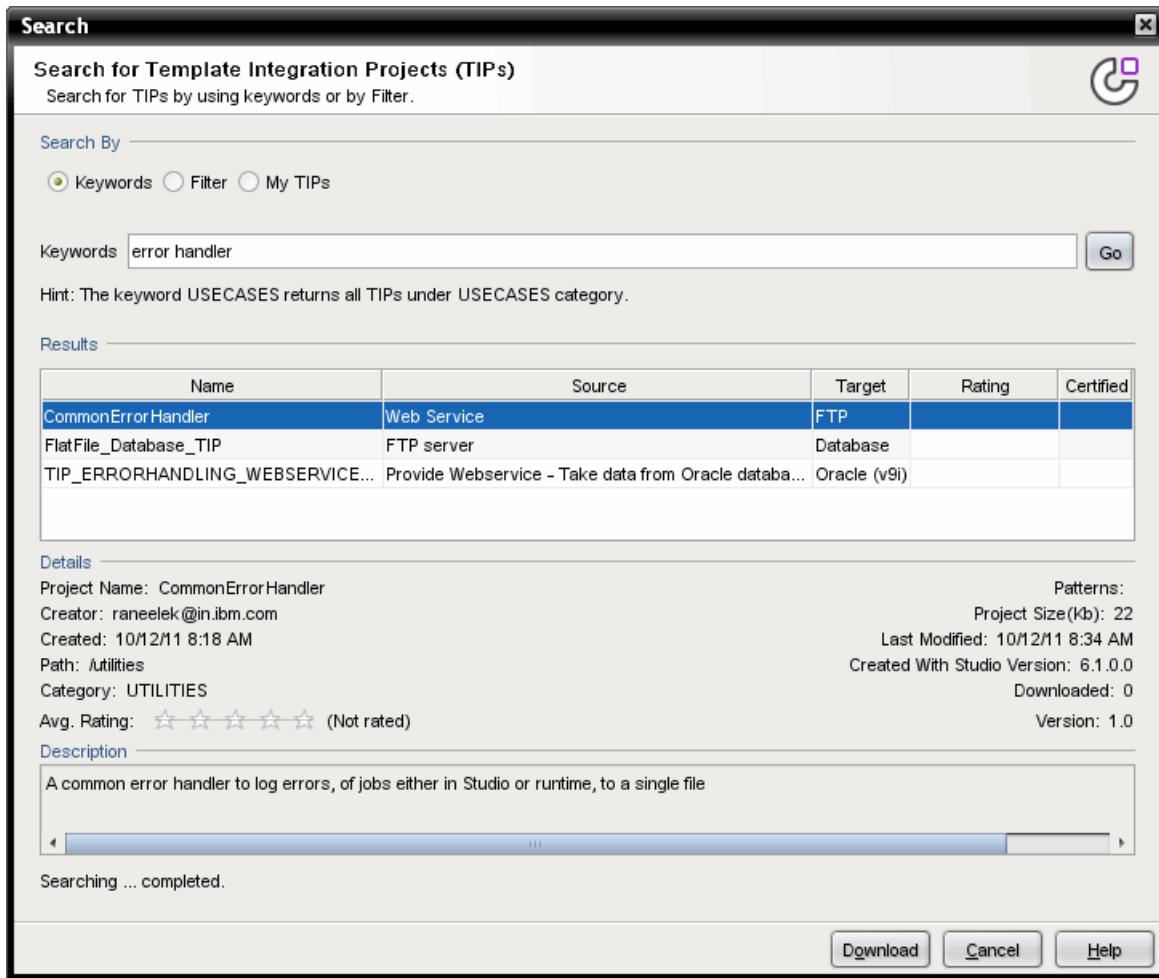


Figure 7-23 TIPs search result

4. Browse through the descriptions of the TIPs in the Results section by selecting each TIP and reading the information in the Details and Description sections. When you find the one that you want to download, select it, and click **Download**.
5. When prompted, read and accept the license agreement, and click **OK**.

6. A dialog box opens for you to select a location to download the TIP, as shown in Figure 7-24. Select the location, and click **Open**. The TIP is downloaded to a specified location and opened in Studio for use.

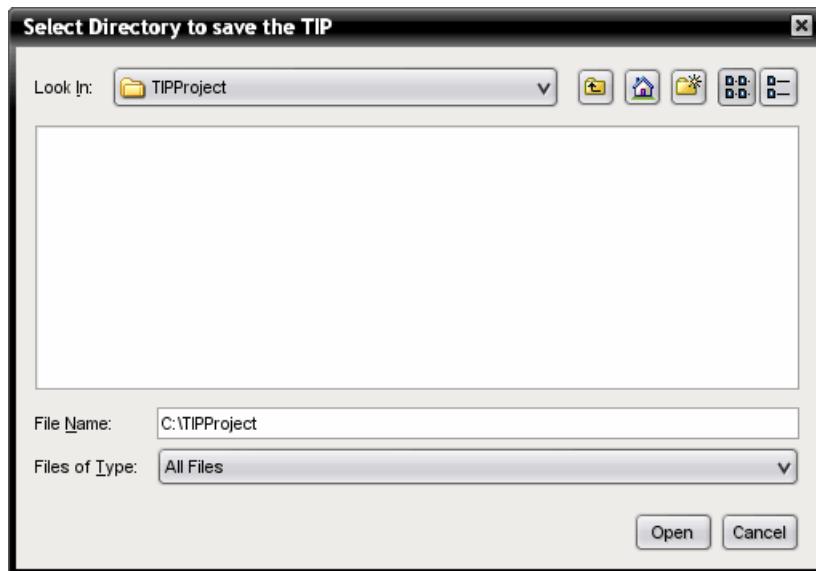


Figure 7-24 Save TIP dialog box

7.6 Verifying the TIP

You can configure and verify the TIP using the TIP Configuration Wizard. The wizard will take you through the steps that were defined in the configuration editor:

1. Launch the Configuration Wizard by selecting **Solutions** → **Start Configuration Wizard**.
2. The Introduction step gives a brief overview of the integration scenario, as shown in Figure 7-25. Click **Next**.

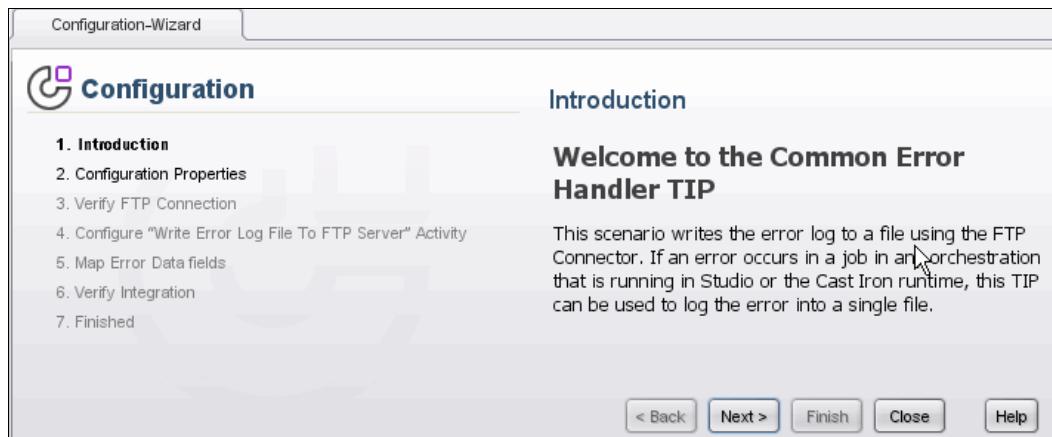


Figure 7-25 Configuration Wizard Introduction step

3. Read the instructions, and enter valid values for the required configuration properties, as shown in Figure 7-26 on page 313. Click **Next**.

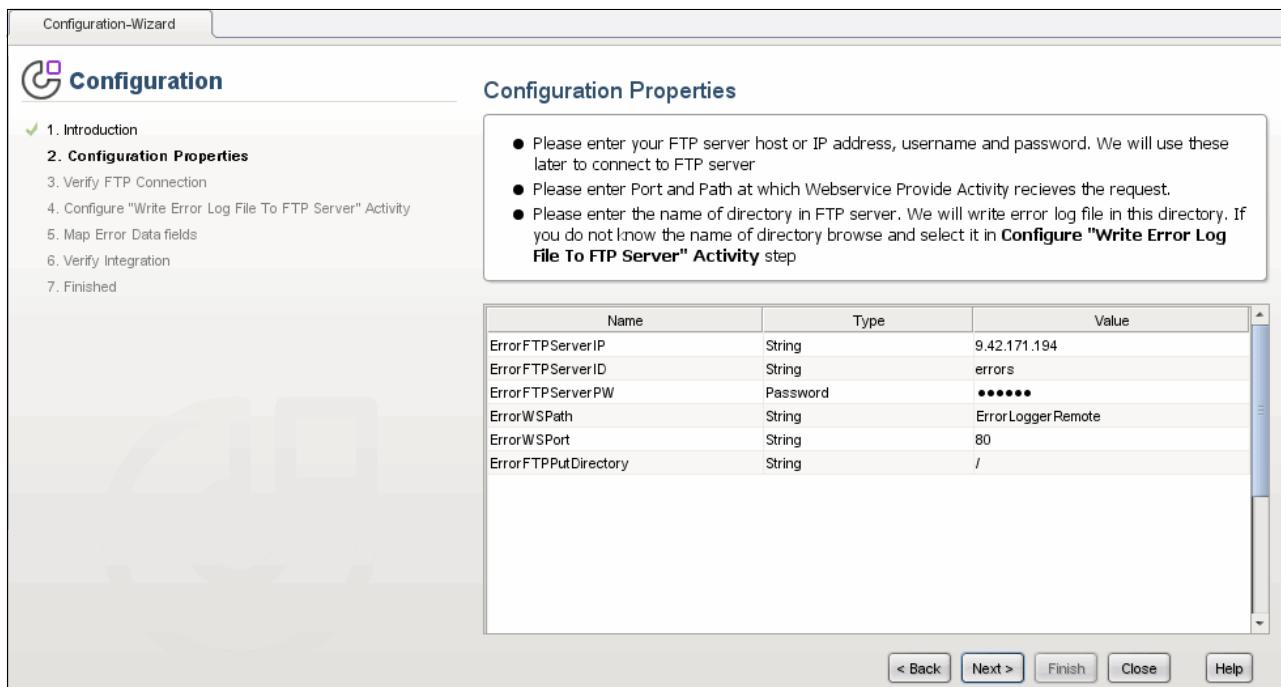


Figure 7-26 Configuration Wizard Configuration Properties step

4. Verify the connection to the FTP server by clicking **Test Connection**. If you can connect successfully to the endpoint, a message displays, as shown in Figure 7-27. Click **OK**, and then click **Next**.

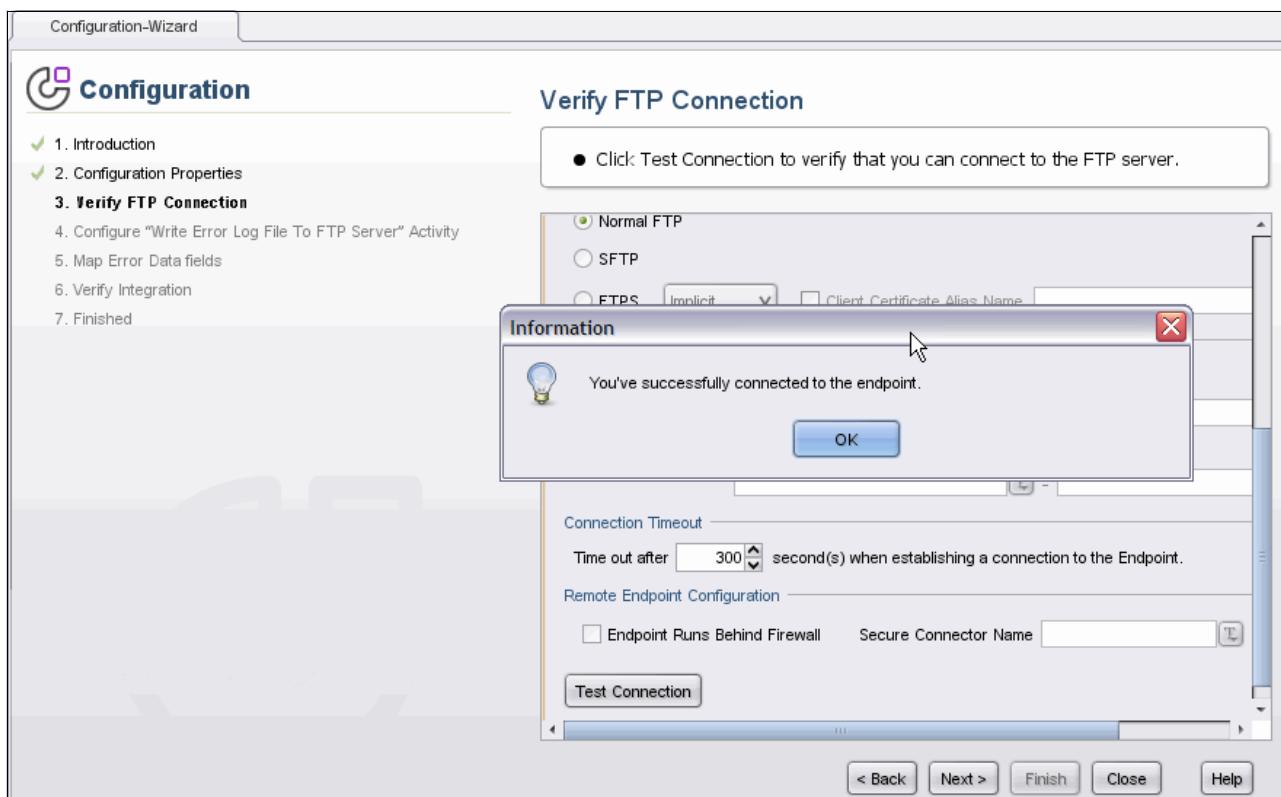


Figure 7-27 Configuration Wizard Verify FTP Connection step

5. Select the directory for FTP to write the file to, and select the type of file, as shown in Figure 7-28. Click **Next**.

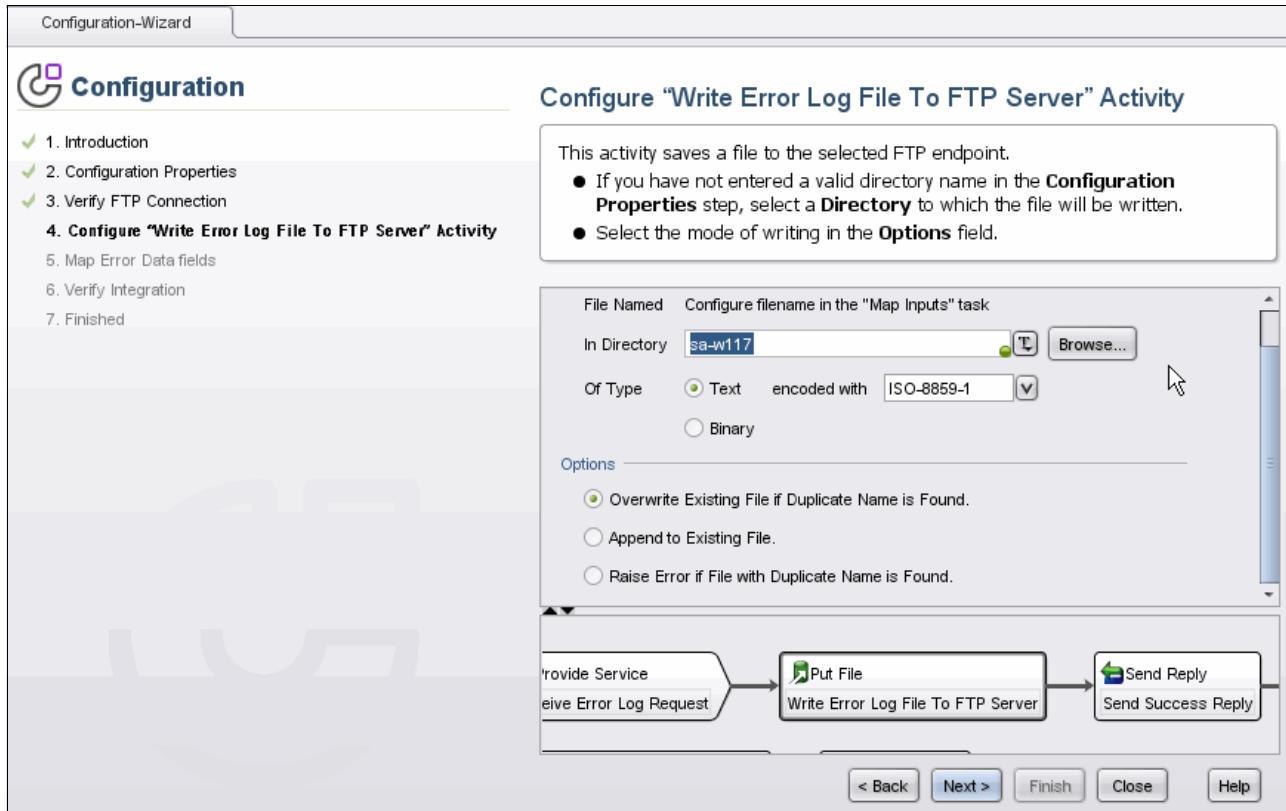


Figure 7-28 Configuration Wizard Configure "Write Error Log File To FTP Server" activity step

6. Optionally, you can change the mapping in the Map Error Data fields step, as shown in Figure 7-29. For our scenario, the current mapping is fine. Click **Next**.

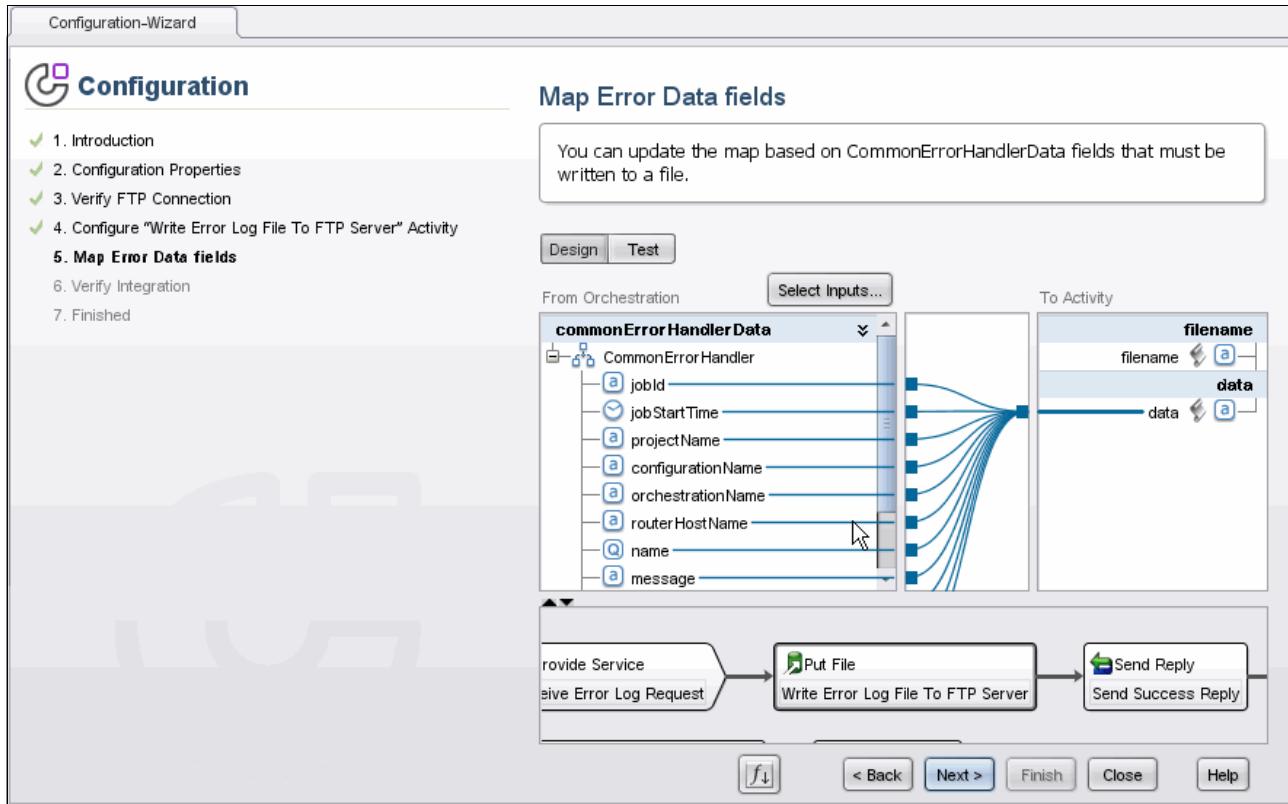


Figure 7-29 Configuration Wizard Map Error Data fields step

7. In this step, you verify that the TIP is working properly, as shown in Figure 7-30 on page 316. We will skip testing for now. Click **Finish**.

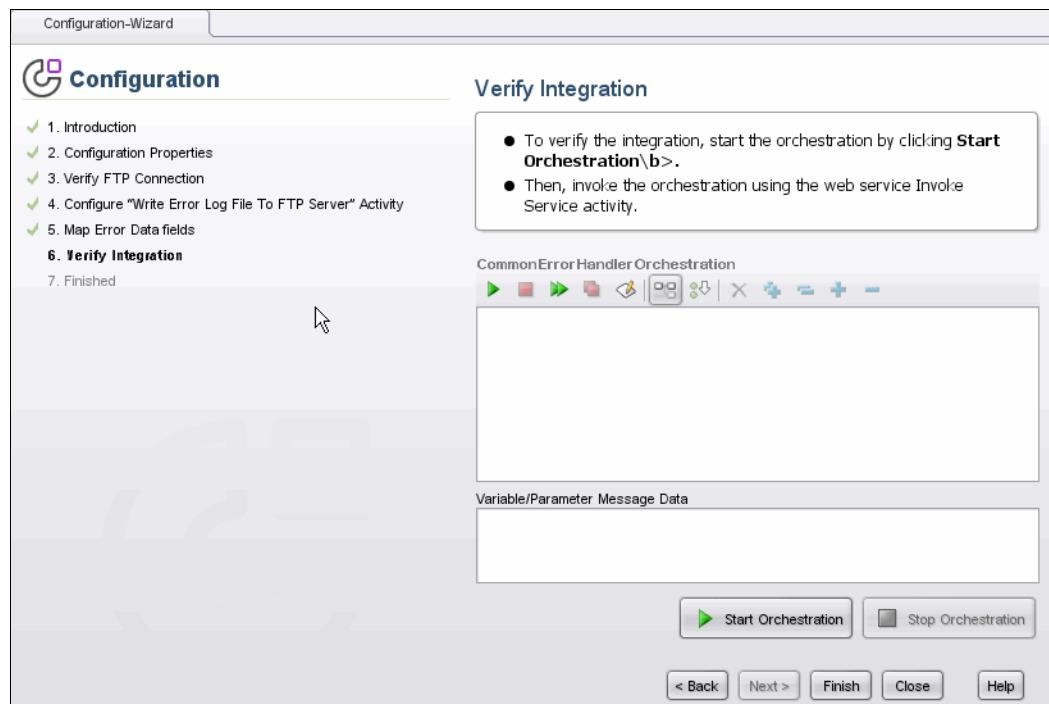


Figure 7-30 Configuration Wizard verification step

8. The final configuration panel looks like Figure 7-31. Click **Close**.

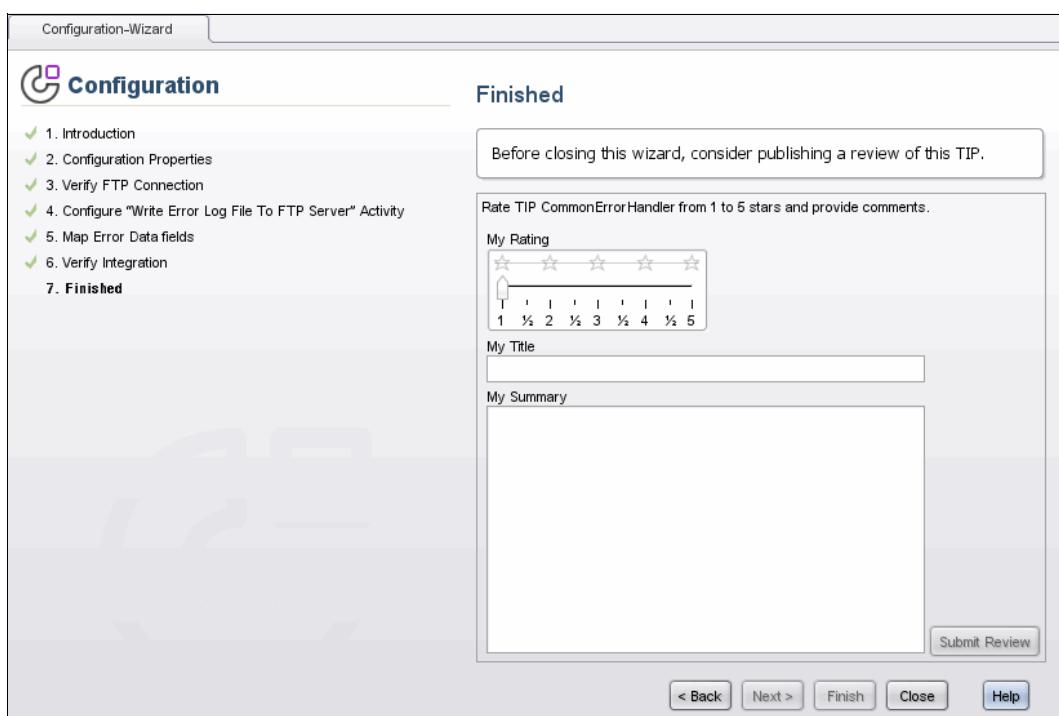


Figure 7-31 Configuration Wizard final panel

9. Test this orchestration using the Invoke Service Tool in the Studio:
 - a. Launch the Invoke Service Tool from the Studio menu by selecting **Orchestrations → Invoke Service**. A window opens with the SOAP envelope.

- b. Enter test values in the fields, as shown in Figure 7-32, and click **Execute**.

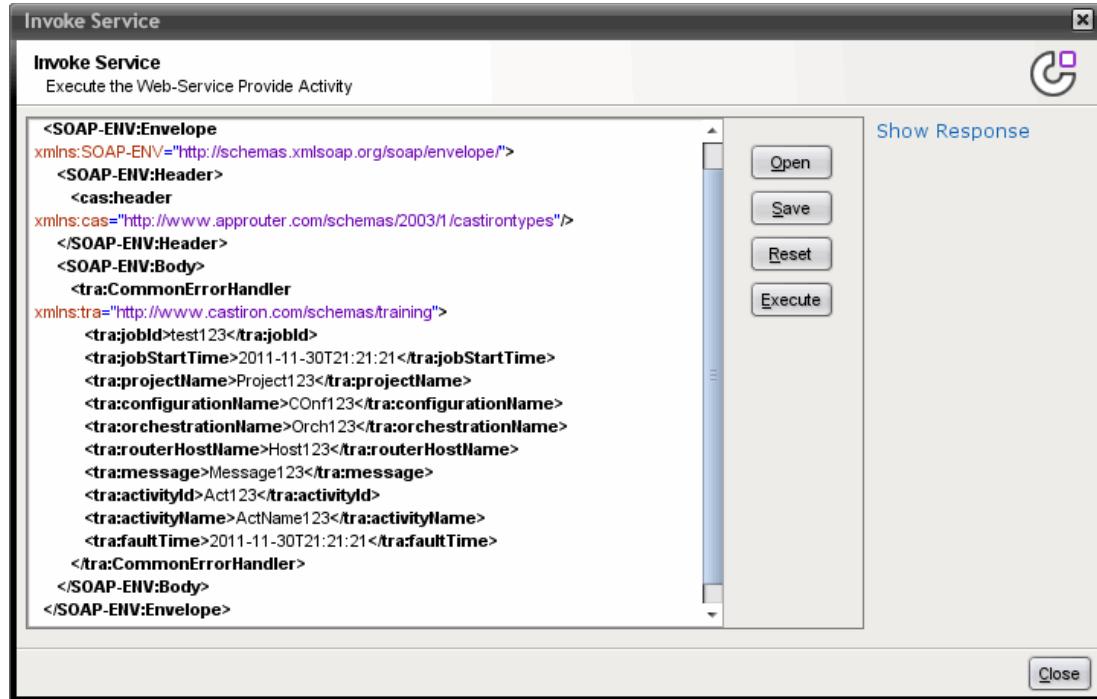


Figure 7-32 *Invoke Service input*

Our test SOAP message is shown in Example 7-1.

Example 7-1 Test input

```

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <cas:header
      xmlns:cas="http://www.approuter.com/schemas/2003/1/castirontypes"/>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
      <tra:CommonErrorHandler
        xmlns:tra="http://www.castiron.com/schemas/training">
          <tra:jobId>test123</tra:jobId>
          <tra:jobStartTime>2011-11-30T21:21:21</tra:jobStartTime>
          <tra:projectName>Project123</tra:projectName>
          <tra:configurationName>COnf123</tra:configurationName>
          <tra:orchestrationName>Orch123</tra:orchestrationName>
          <tra:outerHostName>Host123</tra:outerHostName>
          <tra:message>Message123</tra:message>
          <tra:activityId>Act123</tra:activityId>
          <tra:activityName>ActName123</tra:activityName>
          <tra:faultTime>2011-11-30T21:21:21</tra:faultTime>
        </tra:CommonErrorHandler>
      </SOAP-ENV:Body>
    </SOAP-ENV:Envelope>
  
```

- c. View the results by clicking **Show Response**, as shown in Figure 7-33. Validate that the result is as expected.

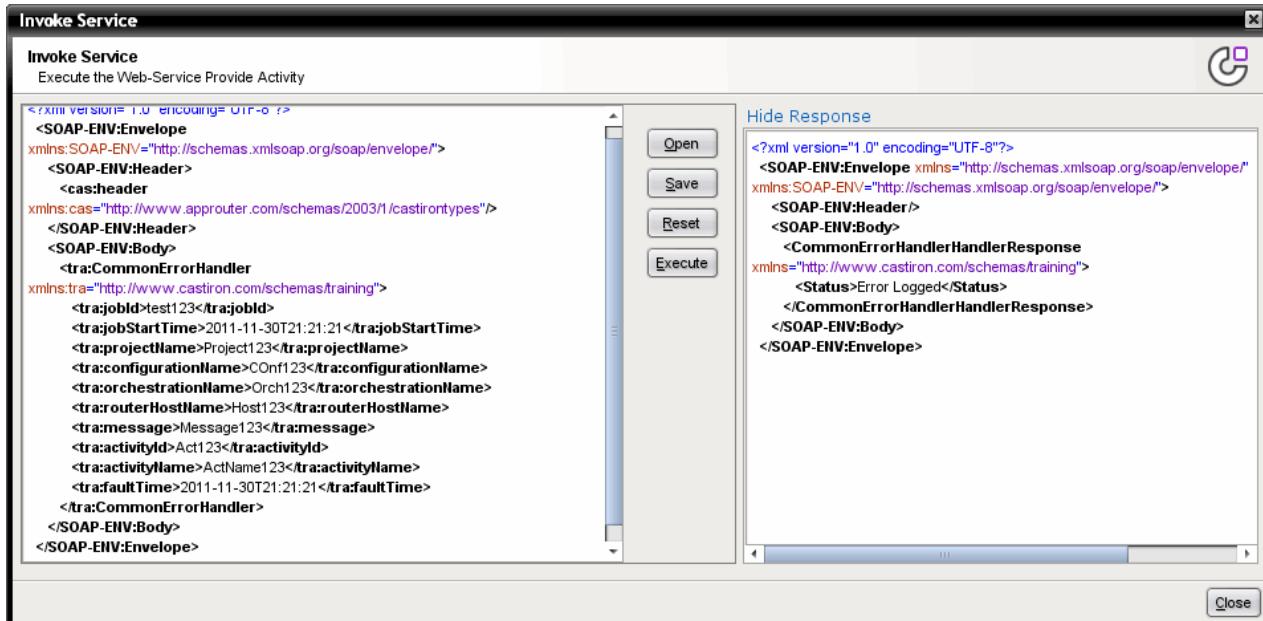


Figure 7-33 *Invoke Service* result

10. The TIP is now configured and ready to use. Click **Finish**.
11. You can also rate the TIP, as discussed in 7.7, “Rating and reviewing TIPs” on page 318.
12. Click **Close**.

7.7 Rating and reviewing TIPs

A TIP can be rated and given review comments by anyone who uses it. The rating scale is from one to five, where one is the lowest and five is the highest. You can view ratings in the Details section of the Search of TIPs window.

In the Studio, when a user submits a rating and review comments, the rating is uploaded to the Cast Iron solutions repository. That user's rating is averaged with other ratings submitted by other users. That average is displayed in the Details section for a particular TIP.

You can rate and review a TIP using either of the following methods:

- ▶ Rate and review in the TIP Configuration Wizard
- ▶ Rate and review using Publish Review

7.7.1 Rating and reviewing in the TIP configuration

To use the TIP Configuration Wizard to review a tip:

1. Launch the TIP Configuration Wizard from the **Studio** menu. Select **Solution** → **Start Configuration Wizard**.
2. Complete the wizard steps, and then click **Finish**. The rating and review page displays, as shown in Figure 7-34 on page 319.

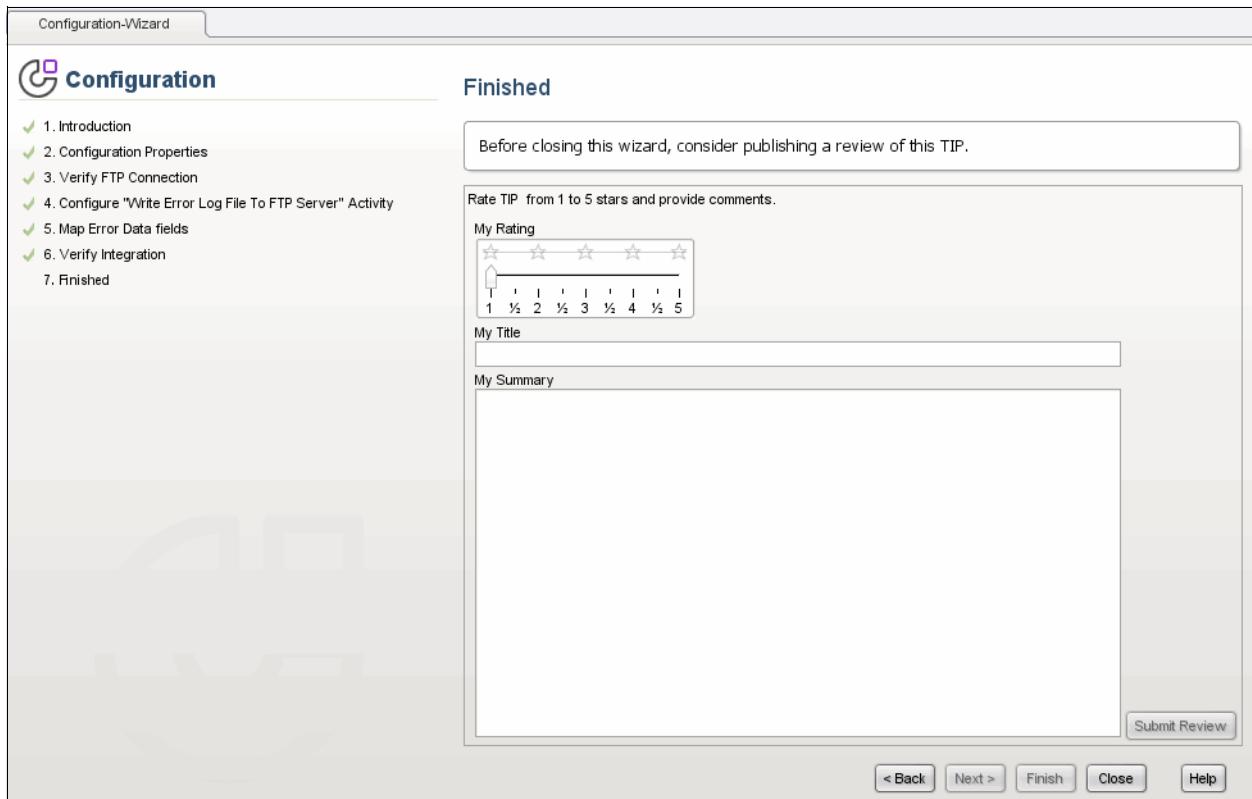


Figure 7-34 Rating and Review in TIP Configuration Wizard

3. In the rating and review page shown in Figure 7-34:
 - a. In the My Rating section, move the slider until the appropriate number of stars are highlighted.
 - b. In the My Title field, give a brief description.
 - c. In the My Summary field, add more review comments for the project.
 - d. Click **Submit Review**. The Studio uploads the rating and review comments to the Cast Iron solutions repository.
4. Click **Close**.

7.7.2 Rating and reviewing using Publish Review

You can also rate and review downloaded TIPs:

1. Login to Cast Iron community, see “Prerequisites” on page 294.
2. Select **Solutions** → **Create Review** in the Studio menu to launch the Publish Review window shown in Figure 7-35 on page 320:
 - a. In the My Rating section, move the slider until appropriate number of stars are highlighted.
 - b. In the My Title field, give a brief description.
 - c. In the My Summary field, add more review comments for the project.

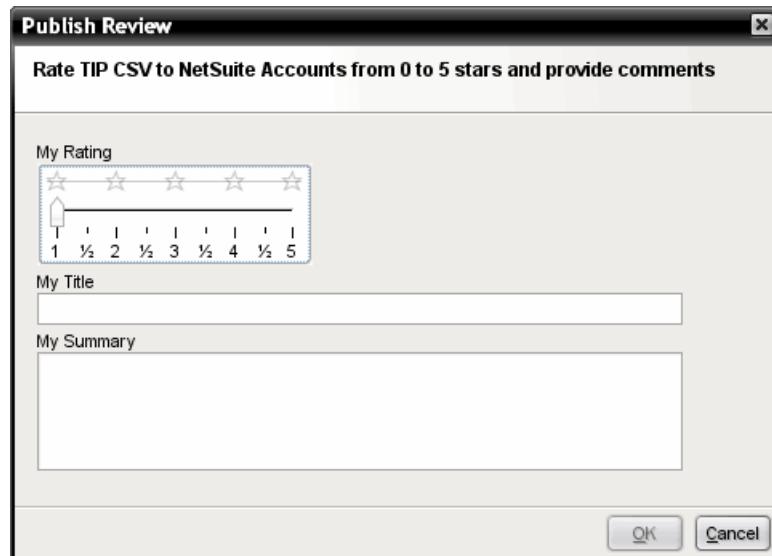


Figure 7-35 Rate and review in Publish Review window

3. Click **OK**. Studio uploads the rating and review comments to the Cast Iron solutions repository.



Building custom plugin connectors

The Connector Development Kit (CDK) provides a method for building connectors to endpoints that are not already included in the Cast Iron product, reducing the need to manually build these connections over and over. This chapter provides an overview of the IBM Cast Iron Connector Development Kit (CDK) and its components. This chapter also illustrates how to publish the CDK connector to a local repository.

This chapter includes the following topics:

- ▶ Introduction to the CDK
- ▶ The CDK components
- ▶ Developing a CDK Connector
- ▶ Publishing the CDK Connector to a local repository
- ▶ Testing and debugging the CDK Connector
- ▶ Sharing the CDK Connector

Disclaimer: The intent of this tutorial is to demonstrate the use of IBM WebSphere Cast Iron Cloud Integration to integrate and utilize third-party applications and does not claim any endorsement or affiliation with the listed products. All product and company names are trademarks or registered trademarks of their respective holders, and IBM disclaims any ownership in such third-party marks. Use of such third-party marks does not imply any affiliation with or endorsement by or for IBM or IBM WebSphere Cast Iron Cloud Integration. The use of any third-party trademarks, logos, or brand names is for informational and instructional purposes only and does not imply that such trademark owner has authorized IBM to promote its products or services.

8.1 Introduction to the CDK

With Cast Iron, you can create connectors to other applications. A *connector* provides a method of extracting data from applications or other data sources. A connector is composed of activities and an endpoint. The *activities* in the orchestration send a request to the endpoint. The *endpoint* receives the request, sends it to the destination application (backend), and then sends a reply back to the activity.

You can use the CDK to build a connector to any entity that exposes either web services or Representational State Transfer (ReST) APIs, such as OracleCRMOnDemand, RightNow, SalesForce, NetSuite, Google, and other applications¹. Connectors that are developed with the CDK are interchangeably known as *Plug-in Connectors*, *Type2 Connectors*, or *CDK Connectors* and require zero coding.

The CDK exposes the following basic APIs. These APIs are used only in Studio for testing connectivity to endpoints and for configuring the CDK connector activity.

- ▶ Test Connection API

When you use a connector activity in an orchestration, you can test the endpoint by entering the connection information. The Test Connection API of the connector allows you to test the connection using the information you enter, as shown in Figure 8-1.



The screenshot shows a dialog box titled "Provide Connection Info". It contains two text input fields: "UserName" and "Password", each with a small "T" icon to its right indicating they are text input fields. Below these fields is a "Test Connection" button.

Figure 8-1 Test Connection

¹ See Disclaimer on page 321.

► List Objects

Use this API to discover and list the objects in the endpoint. Start the API by clicking **Browse** in the Configure panel of the connector activity. A browse window opens, as shown in Figure 8-2, that allows you to search for object types. The List Objects orchestration is invoked internally, and the result is displayed in the Object Type and Object Label columns.

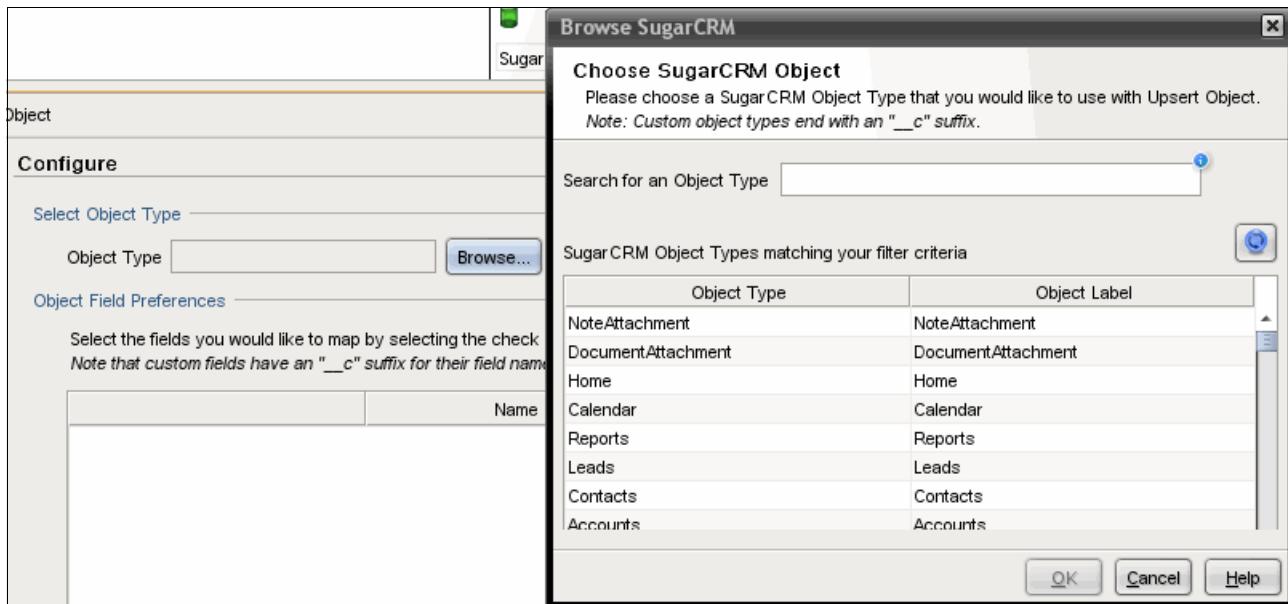


Figure 8-2 List Objects

► Describe Objects

Use this API to list all of the fields and attributes of the selected object. Start it by selecting an object from the list of objects that are displayed in the object selection window, Figure 8-2, and then clicking **OK**. This action opens the list of fields in the object, as shown in Figure 8-3 on page 324.

Configure

Select Object Type

Object Type

Object Field Preferences

Select the fields you would like to map by selecting the check box next to the field name.
Note that custom fields have an *"_c"* suffix for their field name.

Accounts	Name	Label	Type
<input type="checkbox"/>	id	ID	STRING
<input type="checkbox"/>	name	Name:	STRING
<input type="checkbox"/>	date_entered	Date Created:	STRING
<input type="checkbox"/>	date_modified	Date Modified:	STRING
<input type="checkbox"/>	modified_user_id	Modified By	STRING
<input type="checkbox"/>	modified_by_name	Modified By	STRING
<input type="checkbox"/>	created_by	Created By	STRING
<input type="checkbox"/>	created_by_name	Created By	STRING
<input type="checkbox"/>	description	Description:	STRING
<input type="checkbox"/>	deleted	Deleted	BOOLEAN
<input type="checkbox"/>	assigned_user_id	Assigned User:	STRING

Figure 8-3 Describe Objects

8.2 The CDK components

The CDK Connector is based on the following types of files, as illustrated in Figure 8-4:

- ▶ Schema files
- ▶ XML files
- ▶ WSDL files
- ▶ .par files

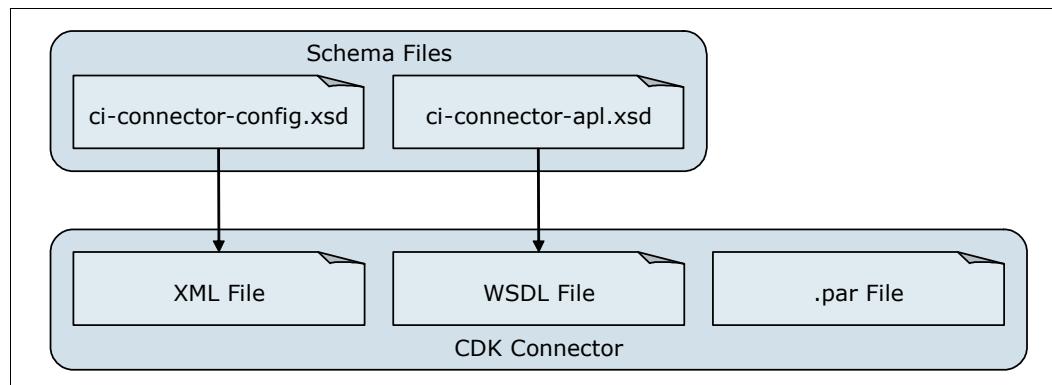


Figure 8-4 Overview of CDK Components

All of these components are interpreted by the CDK platform and render as a connector to Studio and the run time. However, the run time uses only the .par file. This section discusses each of these components in detail.

8.2.1 Schema files

The CDK exposes the following schema files:

- ▶ ci-connector-config.xsd

This schema contains the syntax to define connector configuration details, for example the endpoint field details and activity details, including the configure, input, and output tasks. This file is used by Studio and the deployment engine to read the configuration of the connector.

- ▶ ci-connector-api.xsd

You use this schema to access the endpoint. Studio needs to communicate with the appropriate connector using the connector APIs. This schema file defines the syntax for the request and response of the Test Connection, List Objects, and Describe Object APIs.

Both files are common for all connectors. These schema files provide the syntax and define how elements and attributes are represented in the XML and WSDL file.

For further information about the structure of these files, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.cdk.doc/cdk_schemas.html

8.2.2 XML file

XML files represent and transmit structured data. These files are independent of the software platform, programming language, and hardware. With respect to the CDK connector components, the XML file lists the endpoint fields and activities details and conforms to the syntax given in the ci-connector-config.xsd file. Activity names must match the corresponding web service in the WSDL file, as described in 8.2.3, “WSDL file” on page 327.

Example 8-1 shows the XML file of a connector called *MyConnector*. Notice the templates and examples of the endpoint field and activity details.

Example 8-1 The MyConnector.xml file

```
<con:connectorConfiguration
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation =
"http://www.approuter.com/schemas/cdk/config/ci-connector-config.xsd"
    xmlns:con = "http://www.approuter.com/schemas/cdk/config/"
    category = "CRM"
    name = "MyConnectorPluginConnector"
    label = "MyConnector"
    version = "1.0"
    notes = "First release of MyConnector."
    release-date = "2010-11-30T00:00:00.0Z"
    connection-ns =
"http://www.approuter.com/schemas/module/myconnector/connection/"
    <con:description>IBM WebSphere Cast Iron MyConnector</con:description>
    <con:endpoint name = "MyConnector" endpoint-name = "MyConnector">
        <con:field-group name = "details" label = "Connection Details">

            <!--Template Endpoint field deatils -->
            <con:field name = "InternalFieldName" label = "DisplayFieldName"
configurable = "true">
                <con:type>DataType of field</con:type>
```

```

        <con:value-restriction>
            <con:min-length value = "1"/>
        </con:value-restriction>
    </con:field>

        <!--Example Endpoint field deatils -->
    <con:field name = "userNme" label = "User Name" configurable =
    "true">
        <con:type>string</con:type>
        <con:value-restriction>
            <con:min-length value = "1"/>
        </con:value-restriction>
    </con:field>
    <con:field name = "password" label = "Password" configurable = "true">
        <con:type>string</con:type>
        <con:format>password</con:format>
    </con:field>
    </con:field-group>
    <con:test-connection-action short-description = "Tests the connection to
    the endpoint system"/>
    <con:endpoint-description>MyConnector Endpoint
    description</con:endpoint-description>
    </con:endpoint>
    <con:activity-group>

        <!--Template Activity Details -->
    <con:activity name = "InternalActivityName" label = "DisplayActivityName">
        <con:operation-name>Update</con:operation-name>
        <con:description>Breif description of actiivty</con:description>
        <con:task-list>
            <!--Consists of required tasks for current activity-->
            <con:configure-task>
                <con:type>browse</con:type>
                <con:applies-to>inputs</con:applies-to>
            </con:configure-task>
        </con:task-list>
    </con:activity>

        <!--Example Activity Details -->
    <con:activity name = "InsertObject" label = "Insert Object">
        <con:operation-name>Insert</con:operation-name>
        <con:description>MyConnector Insert Object</con:description>
        <con:task-list>
            <con:configure-task>
                <con:type>browse</con:type>
                <con:applies-to>inputs</con:applies-to>
            </con:configure-task>
        </con:task-list>
    </con:activity>
    </con:activity-group>
    <con:operations default-endpoint-location = "local:///MyConnector/1.0.0.0"/>
</con:connectorConfiguration>

```

8.2.3 WSDL file

A WSDL file is based on the XML format file, and it is used as a standard interface to define and expose web services. With respect to the CDK connector, this WSDL file imports the `ci-connector-api.xsd`. It defines the web service for each activity of the connector. The name of these web services must match with the activity names that are specified in the XML file. In addition, the WSDL file contains connection details, and the names of these connection details must match with the names of the endpoint fields that are specified in the XML file.

Example 8-2 includes a sample WSDL file.

Example 8-2 A sample WSDL file

```
<wsdl:definitions
    name = "MyConnector"
    targetNamespace = "http://www.myconnector.com/connector"
    xmlns:wsdl = "http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns = "http://www.myconnector.com/connector"
    xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
    xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xmlns:conf = "http://www.approuter.com/schemas/cdk/api/">
    <wsdl:types>
        <xsd:schema targetNamespace = "http://www.approuter.com/schemas/cdk/api/">
            xmlns:xsd = "http://www.w3.org/2001/XMLSchema" elementFormDefault =
            "qualified">
                <xsd:element name = "details">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element minOccurs = "1" name = "url" type =
                            "xsd:string"/>
                            <xsd:element minOccurs = "1" name = "userName" type =
                            "xsd:string"/>
                            <xsd:element minOccurs = "1" name = "password" type =
                            "xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:schema>
            <xsd:schema elementFormDefault = "qualified"
                attributeFormDefault = "unqualified" targetNamespace =
                "http://www.myconnector.com/connector">
                <xsd:complexType name = "InsertObject"/>
                <xsd:element name = "Insert">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element maxOccurs = "unbounded" minOccurs = "0" name
                            = "InsertObject" type = "tns:InsertObject"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
                <xsd:element name = "InsertResponse">
                    <xsd:complexType>
                        <xsd:sequence>
```

```

                <xsd:element maxOccurs = "unbounded" minOccurs = "0" name
= "Result" type = "tns:Result"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        <xsd:complexType name = "Result">
            <xsd:sequence>
                <xsd:element minOccurs = "0" name = "Id" type = "xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:schema>
</wsdl:types>
<wsdl:message name = "InsertRequest">
    <wsdl:part element = "tns:Insert" name = "request"/>
</wsdl:message>
<wsdl:message name = "InsertResponse">
    <wsdl:part element = "tns:InsertResponse" name = "response"/>
</wsdl:message>
<wsdl:message name = "details">
    <wsdl:part element = "conf:details" name = "details"/>
</wsdl:message>

<wsdl:portType name = "MyConnectorPort">
    <wsdl:operation name = "Insert">
        <wsdl:input message = "tns:InsertRequest"/>
        <wsdl:output message = "tns:InsertResponse"/>
    </wsdl:operation>
</wsdl:portType>

<wsdl:binding name = "MyConnectorBinding" type = "tns:MyConnectorPort">
    <soap:binding style = "document" transport =
"http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name = "Insert">
        <wsdl:input>
            <soap:header message = "tns:details" part = "details" use =
"literal"/>
            <soap:body use = "literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use = "literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>

<wsdl:service name = "MyConnectorService">
    <wsdl:port binding = "tns:MyConnectorBinding" name = "MyConnectorPort">
        <soap:address location =
"http://{{property:/ApplianceDataIP}}/MyConnector_Connector/1.0.0.0"/>
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

8.2.4 The .par file

The CDK Connectors are projects inside Studio and are represented as .par files. Orchestrations in this project correspond to the implementation of each of the connector activities and the Test Connection, List Objects, and Describe Object APIs, respectively. This file is used by the deployment engine during the deployment of any orchestration containing the CDK Connector activities.

8.3 Developing a CDK Connector

The CDK provides a wizard that you can use to develop the connector. This section describes how to use the CDK Wizard to develop a connector that interacts with the Google Calendar APIs². To use the CDK, a developer needs a working knowledge of XML, XSLT, XSD, and WSDL.

8.3.1 Using the CDK Wizard to create a connector

The CDK Wizard collects the information that is needed to create a connector. Using this information, the wizard generates template orchestrations that you must implement, and the WSDL and XSD files. When you publish a connector to the local repository, the information that you specify in wizard is used to create the XML file.

To create a connector:

1. To launch the CDK Wizard in Studio, select **File → New**. Then, select either **Create New Connector Project** or **Update Connector Project**, depending on the situation.
2. Enter the connector information

You begin the wizard by entering the connector details, as shown in Figure 8-5.

Connector Information

Provide Basic Connector Detail

Name: Connector Name

Version: 1.0.0

Description: Add Description About This Connector.

Add WSDL Or Schema Files

Add WSDL/Schema:

Figure 8-5 Connector Information page

Enter the following information in Figure 8-5:

- a. Enter the name of the connector in the Name field.
- b. Enter the version number of the connector in the Version field, using the format $X.X.X$, where X is an integer. This field is used to specify the proper version number when updating the connector.

² See Disclaimer on page 321.

- c. Enter a brief overview of the connector in the Description field.
- d. Add the WSDL or Schema (XSD) file to be used as part of the connector development. WSDL files are usually provided by a SaaS provider, but the Schema file is created by the developer and contains the I/O types of activities. Click **Browse** to select the file location.

Select the file, and then click **OK**. If the selected file imports other files, a window opens so that you can locate the other files also. Click **Locate** to specify their location. When finished, click **OK**.

The selected file is processed and loaded into Studio.

Click **Next**.

3. Add connection fields, as shown in Figure 8-6.

Connection Field Name	Preview	
UserName		
Password		

Endpoint Connection Fields Details

Name	<input type="text" value="Password"/>	Format	<input type="text" value="PASSWORD"/>
Type	<input type="text" value="STRING"/>		

Field Constraints

<input type="checkbox"/> Enable Constraints			
ValueMinLength	<input type="text" value="0"/>	FieldMinRange	<input type="text" value="0"/>
ValueMaxLength	<input type="text" value="250"/>	FieldMaxRange	<input type="text" value="250"/>

Figure 8-6 Add Connection Fields page

On Figure 8-6, enter the following information:

- a. Add Endpoint Connection Fields

In this section, specify the fields that are used to connect to the endpoint. These fields display in the Endpoint Panel of the connector. The order of the fields in this section is maintained in the Endpoint Panel too. By default, the user name and password fields are pre-configured. The data in the remaining sections corresponds to the selected connection field in this section.

To add a connection field, click the Add icon (). Double-click the added field under the Connection Field Name column to change the name, and then press Enter.

To delete a connection field, select the connection field, and then click the Delete icon ().

Use the Move Up and Move Down icons (▼) and (▲) to change the order of the connection fields.

b. Endpoint Connection Fields Details

In this section, specify the details of the selected field that you added in the Add Endpoint Connection Fields section:

- The Name field displays the name of selected connection field.
- The Format field is used to select the format of data that is entered for the selected connection field. Here, you can set format as either STRING, which is used to display text that is entered as plain text, or PASSWORD, which is used to display a special character to hide the text value that is entered in the field.
- The Type field is used to select the data type of the selected connection field. Available data types are STRING, BASE_64, BOOLEAN, INT, DECIMAL, DATE, DATE_TIME, and ANY_TYPE.

c. Field Constraints

In this section, optionally, you can set constraints to the selected field. Enable field constraints by selecting the **Enable Constraints** option. Set the value and field constraints.

Click **Next**.

4. Add activities

Activities can be either generic or specific:

- Generic activities are executed on multiple objects. Thus, the input and output of this activity must be dynamically configurable because objects have different schemas. For example, in the Salesforce.com connector, you can add a generic activity, such as Update Objects, which can be performed on multiple objects, such as Accounts, Contacts, and others.
- Specific activities perform only a particular job. Thus the input and output for this type of activity must be static. For example, in the Google Calendar connector, you can add a specific activity, such as CreateEntry, that creates an event in Google Calendar.

As shown in Figure 8-7, enter the details of all the activities of the connector.

Add Activities

Activities Added To This Window Are Displayed In Studio Activity Panel Under Folder: **Google_calenar**

Add Connector Activities

Activity Name	
activity0	<input type="button" value="+"/> <input type="button" value="X"/> <input type="button" value="^"/> <input type="button" value="v"/>

Activity Details

Name

Dynamic Input/Output Requirement

Input/Output Parameter Requires Dynamic Discovery

Input is Discovered Dynamically
 Output is Discovered Dynamically
 Both Input/Output are Discovered Dynamically

Select Input/Output Data

Input Type

Output Type

Figure 8-7 Add Activities page

Enter the following information:

a. Add Connector Activities

In this section, enter the names of the activities in the connector. The order of the activities in this section is maintained in the activity toolbox of Studio. By default, this section has one activity entry. You can rename or delete this entry.

Data in the remaining sections corresponds to the selected activity in this section.

To add an activity, click the Add icon (). Double-click the added activity under the Activity Name column to change the name, and then press Enter.

To delete an activity, select the activity, and then click the Delete icon ().

Use the Move Up and Move Down icons () to change the order of activities.

b. Activity Details

This section displays the name of the selected activity and its input/output details, as follows:

- Dynamic Input/Output Requirement

Some activities can be performed on multiple objects; therefore, the input and output of these activities are not static. So, input and output must be discoverable during configuration of the activity. To enable discovery of input or output, select the **Input/Output Requires Dynamic Discovery** option, and then select whether input, output, or both must be discovered dynamically.

- Select Input/Output Data

In this section, select the input and output schemas for the selected activity. The orchestrations created in the connector will have Web Services Provide Service activity, whose input and output is configured to selected input/output schema respectively.

Important: If you select the Input/Output Requires Dynamic Discovery option in the Activity Details panel, select the abstract schema for input/output that must be dynamically discovered.

5. View the final summary of the input.

The CDK Wizard generates the required artifacts. A summary page summarizes the list of artifacts that are generated and the further steps that are needed to complete the development of the connector.

The wizard creates an orchestration for the connector and for each API that is relevant to the connector. If you selected the Input/Output Requires Dynamic Discovery option, you will get an orchestration for each of three APIs: Test Connection, List Objects, and Describe Objects.

8.3.2 Developing a Google Calendar connector using the CDK Wizard

This section explains how to use the CDK Wizard to create a Google Calendar connector. The activity creates an event in a Google Calendar.

To develop a Google Calendar connector:

1. To create a connector, you need a WSDL or schema file to use as input. If you are using web service APIs to connect to the provider, obtain the WSDL from that provider. In this example, however, the ReST API is used to connect to Google Calendar, so the WSDL file is created by the connector developer.

For this scenario, we created a WSDL file, named GcalType.wsdl, with request and response schemas for the CreateEntry activity in its *types*.

Additional material: The WSDL and schema files used to create this connector have been included in the additional materials for this book. For information about obtaining these materials, see Appendix A, “Additional material” on page 517.

2. Open Studio and close any open projects.
3. Create a new connector project. Select **File** → **New** → **Create New Connector Project**.
4. Specify the location to save the connector project by clicking **Browse**. Enter GoogleCalendar as the name of the connector project, and click **OK** to launch the CDK Wizard.
5. Enter the following connector details, as shown in Figure 8-8 on page 334:
 - a. In Name field, enter Google Calendar as the name of the connector.
 - b. Enter a brief description in the Description field.
 - c. Because you are developing the connector now, in the Version field, leave the version number as 1.0.0.
 - d. To add the WSDL file, click **Browse**, and select the GcalType.wsdl file. Click **OK**.

This WSDL file imports two schema files:

- Common.xsd
- GcalResp_Imp2.xsd

To locate the imported files, click **Locate**, and then click **OK** (Figure 8-8).

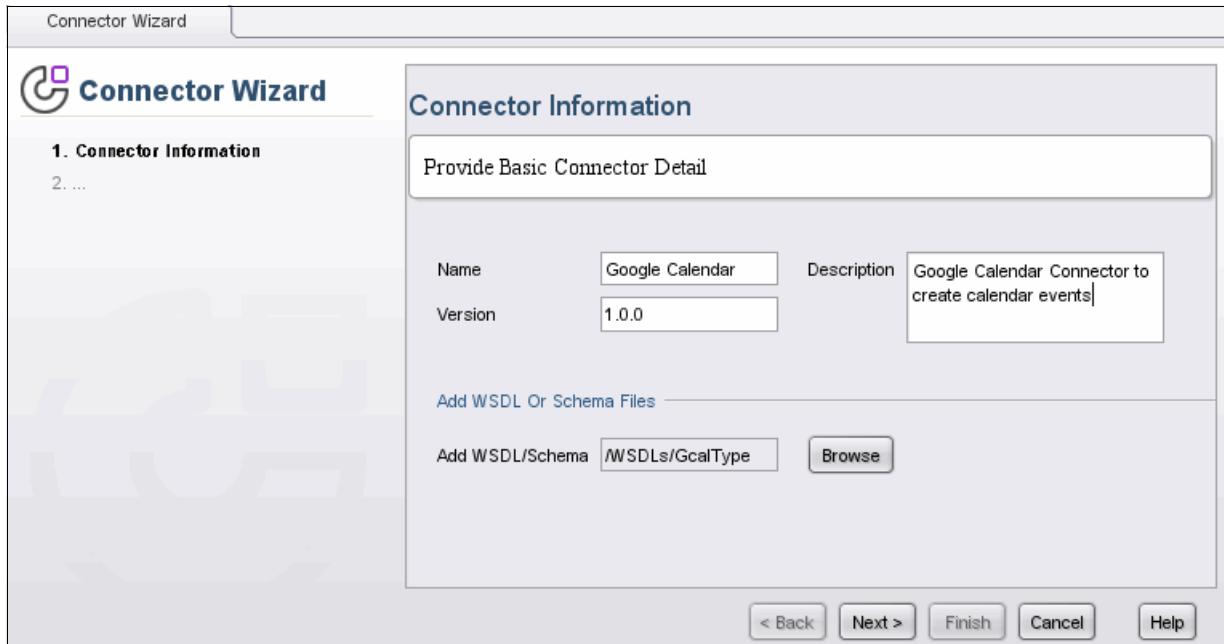


Figure 8-8 Google Calendar connectorInformation page

- e. Click **Next**.
6. The Google login requires only the user name and password fields. By default, these fields are added and pre-configured in the Add Connection Fields page. Click **Next**.
7. Because you will create a Google Calendar event, you must create an activity called CreateEntry. In the Add Activities page, shown in Figure 8-9 on page 335, enter the following information:
 - a. In the Add Connector Activities section, double-click **activity0** and change the name to CreateEntry. Press Enter.
 - b. The input and output schemas of the CreateEntry activity are not dynamically discoverable and must be added manually. In the Select Input/Output section, click **Select Input**, and a window with schemas in the GcalType.wsdl file opens. Select the **entry** schema, and then click **OK**. The Input Type field is populated with the selected schema name.
 - c. Click **Select Output**, and a window with schemas in the GcalType.wsdl file opens. Select the **entry_response** schema, and then click **OK**. The Output Type field is populated with the selected schema name.

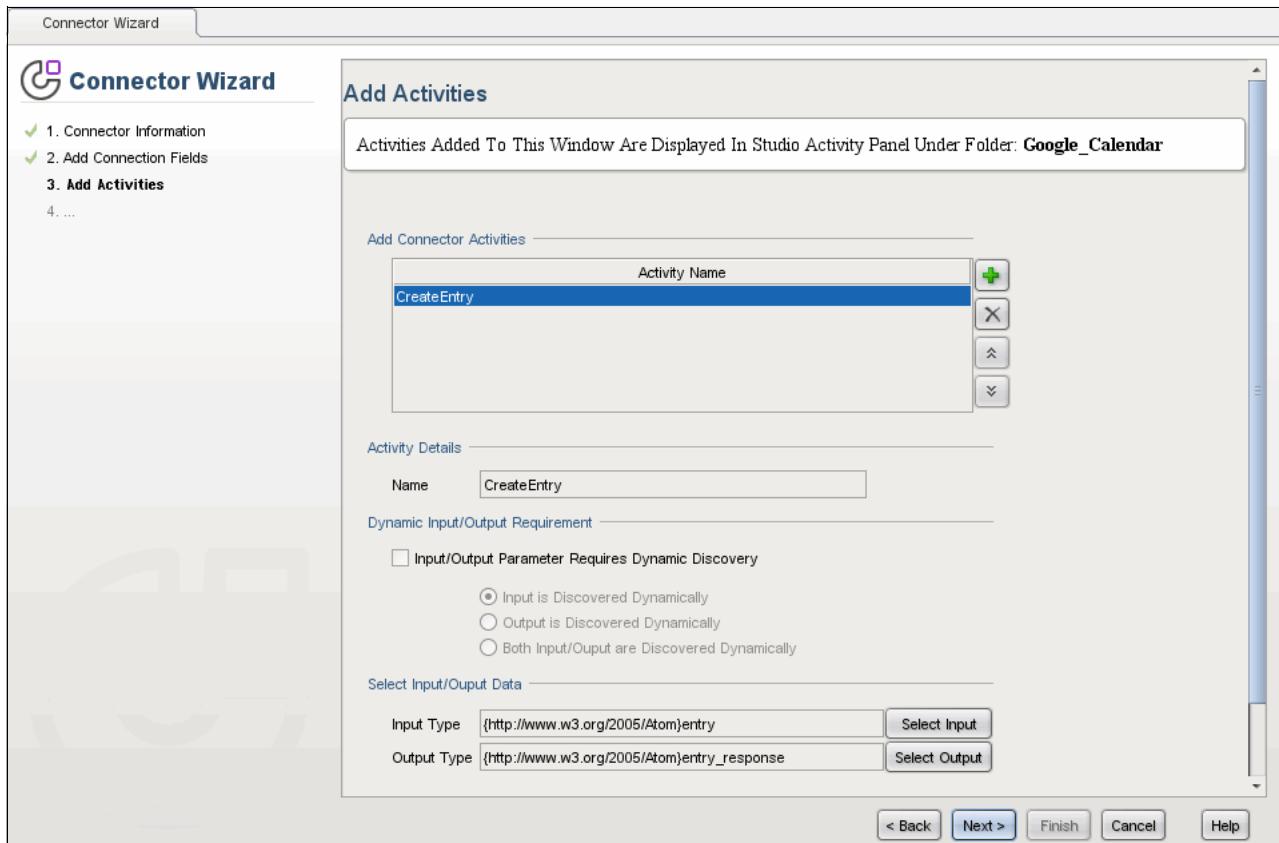


Figure 8-9 Google Calendar connector Add Activities Page

- d. Click **Next**.
8. The CDK Wizard now has the required information to create Studio artifacts and orchestrations. A confirmation window displays. Click **Yes** to generate these artifacts and orchestrations.
9. The Final Summary page gives the summary of the orchestrations that were created and information about the next steps required for the connector development, as shown in Figure 8-10 on page 336.

Two orchestrations were created. The CreateEntry orchestration and the TestConnection orchestration, allowing you to use the Test Connection API. This is the only API of the three that is available for this connector. This is because the CreateEntry activity's input/output is not discoverable. The List Objects and Describe Object APIs are used only for dynamic discovery of schemas in an activity.

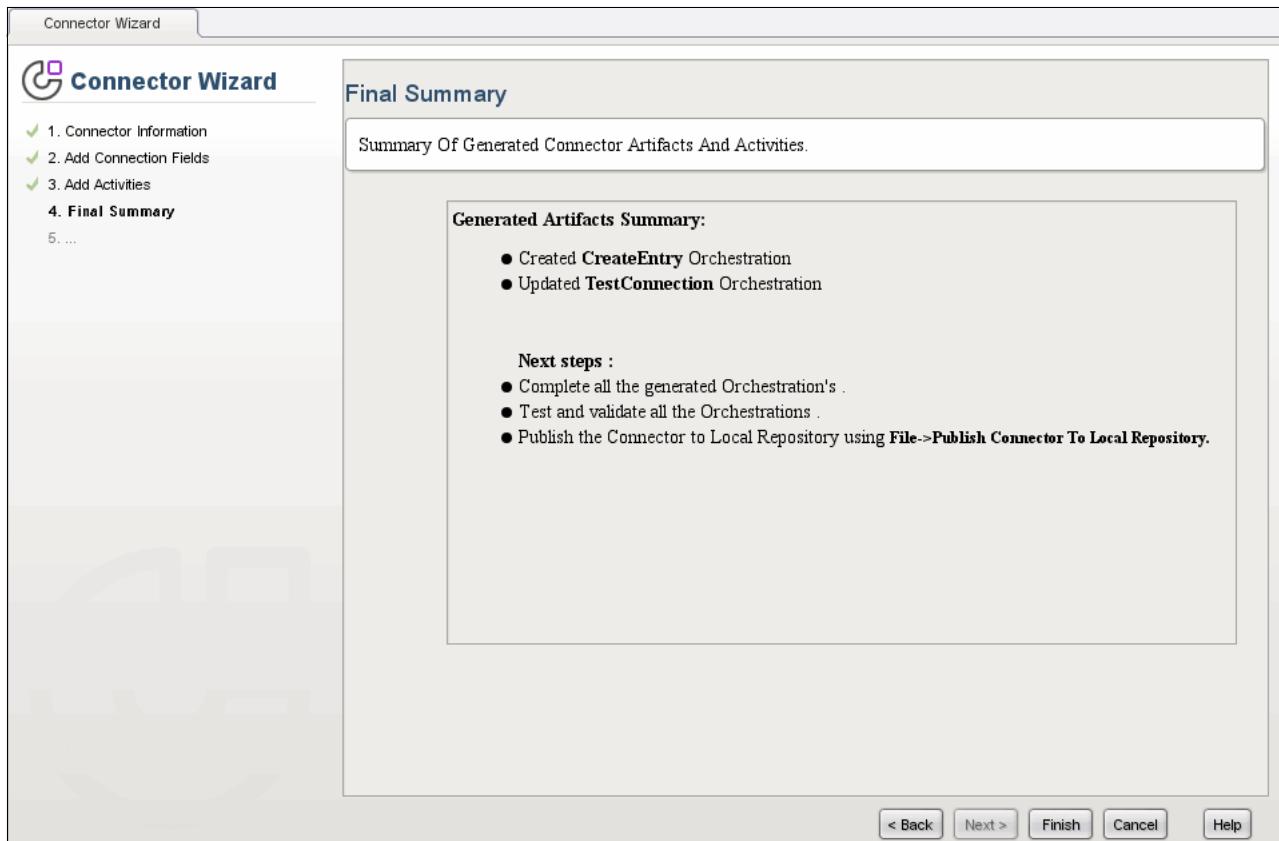


Figure 8-10 Google Calendar connector Final Summary page

10. Click **Finish** to close the CDK Wizard.
11. The new connector project will be open in the Studio.

Completing the TestConnection orchestration

The TestConnection orchestration is designed to test the connection to Google Calendar.

Enter the connection details and click **Test Connection** in the connectors endpoint panel. This starts the orchestration (Test Connection), which receives the values entered and tests the connection to the endpoint.

The orchestrations created by the wizard are basic. Switching to the Project tab and double-clicking the TestConnection orchestration will open the orchestration (Figure 8-11 on page 337).

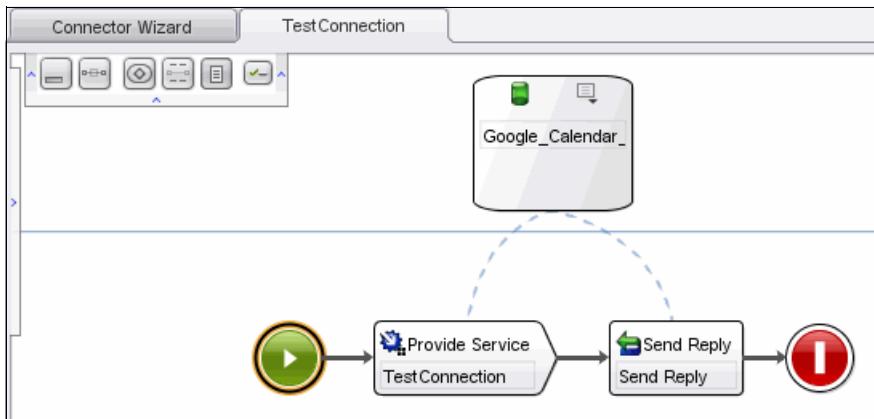


Figure 8-11 TestConnection orchestration

You can extend this orchestration to meet your needs. In this case, we want the Test Connection orchestration to process the **TestConnectionRequest** data received in the Web Service Provide Service activity. This task involves extracting the connection information and using it in an HTTP Post Activity to login to the Google account. It must also handle a login failure. A Send Reply activity returns a proper message and a boolean value in a **TestConnectionResponse**, which is used to display the result of the Test Connection in a dialog box.

In this example, the Test Connection orchestration is completed using the activities in Table 8-1.

Table 8-1 Activities in the Test Connection orchestration

Activity	Activity Name	Activity Use
Web Service Provide Activity	TestConnection	After entering endpoint Details in the Endpoint Panel, when you click the Test Connection button. This orchestration is instantiated where this activity receives entered endpoint details as TestConnectionRequest .
Apply XSLT	Apply XSLT	Extracts user name and password in the TestConnectionRequest .
Map Variables	Decode Password	To check if the extracted password is encoded, we use the Decode Password Configuration function and save the result in String variable.
If	Encoded	If the resultant string variable in the above activity is not empty or "", we can say that the extracted password is encoded.
If-> Map Variables	Map Variables	If the password is encoded, replaced the extracted password value with the decoded version of it using the Decode Password Configuration.
Map Variables	Init Variables	After we have connection details, initialize other variables used for login, such as GoogleSource, GoogleService, and accountType.
Try	Login	Add try catch block to handle gracefully if any error happens during the execution HTTP Post activity used for authenticating to Google.
CatchAll	CatchAll	If an error happens during execution, in this block execution flows continue from here.
Try-> HTTP Post Utility	Login	Makes a login request to login.
Try -> If	Success	Checks if login is successful.

Activity	Activity Name	Activity Use
Try ->If ->Map variables and Try->Else -> Map Variables	Map Variables	Initializes the respective response to the TestConnectionResponse variable.
Catch -> Map Variables	Map Variables	An error during execution of the activity in try block, initializes proper error response to the TestConnectionResponse variable.
Send Reply	Send Reply	Sends the initialized TestConnectionResponse as a Connector activity response.

Figure 8-12 shows the first half of the orchestration.

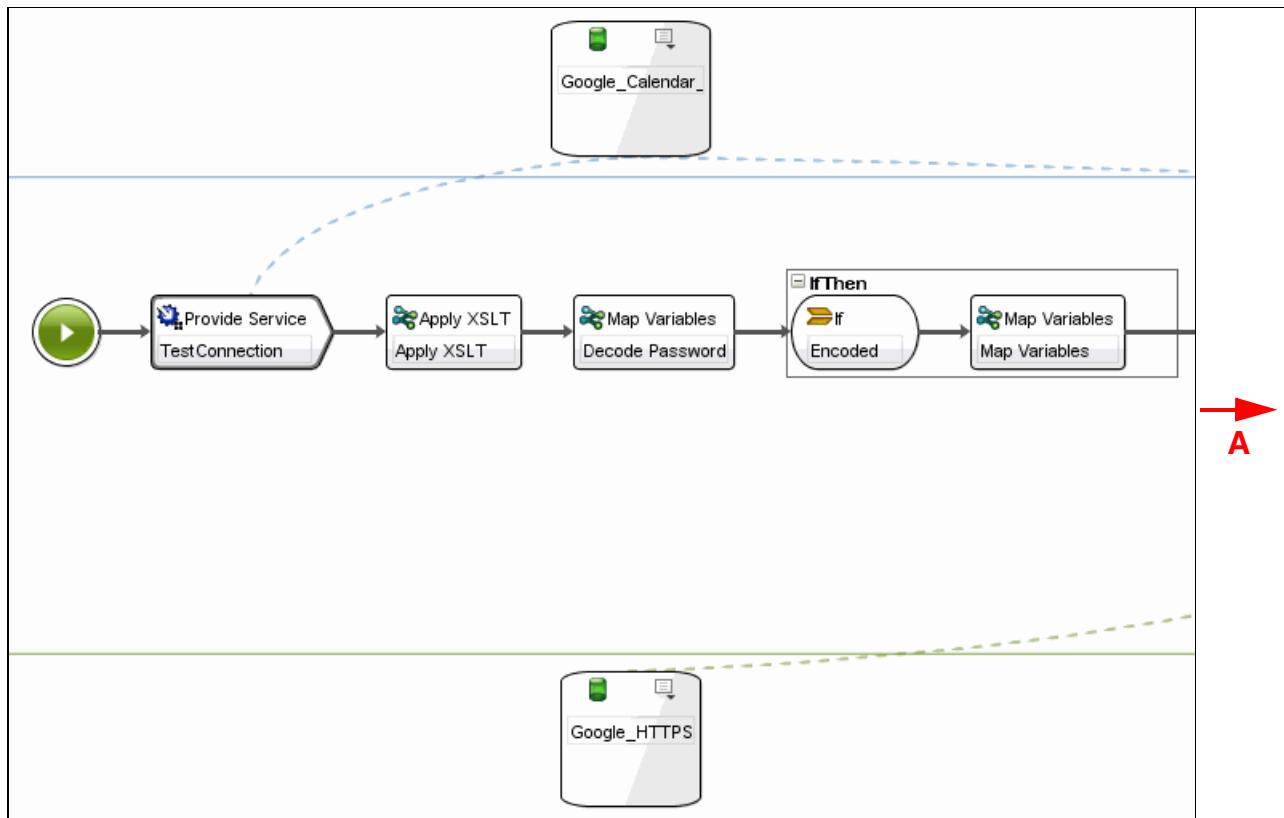


Figure 8-12 First half of the Test Connection orchestration

Figure 8-13 on page 339 shows the second half of the orchestration.

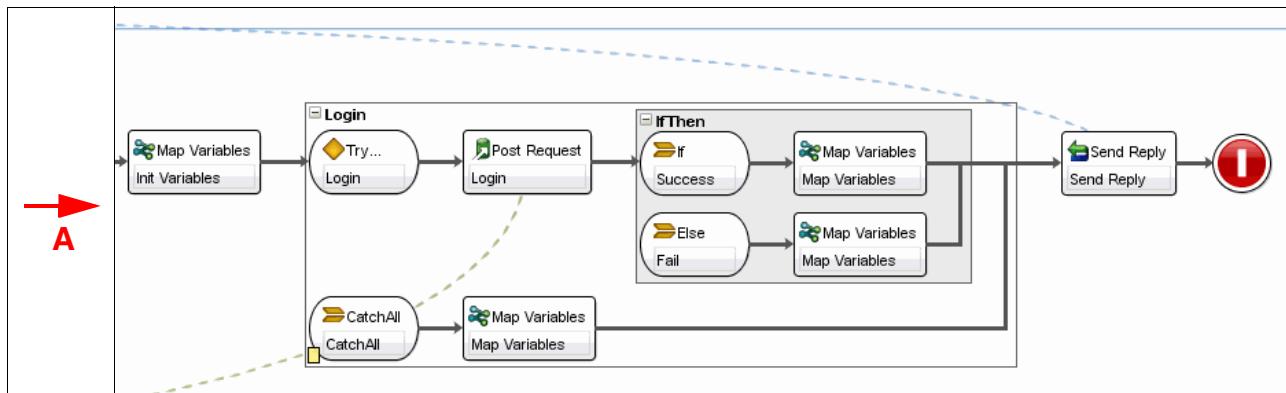


Figure 8-13 Second half of the Test Connection orchestration

Completing the CreateEntry orchestration

The CreateEntry orchestration created by the CDK wizard, Figure 8-14, is also basic.

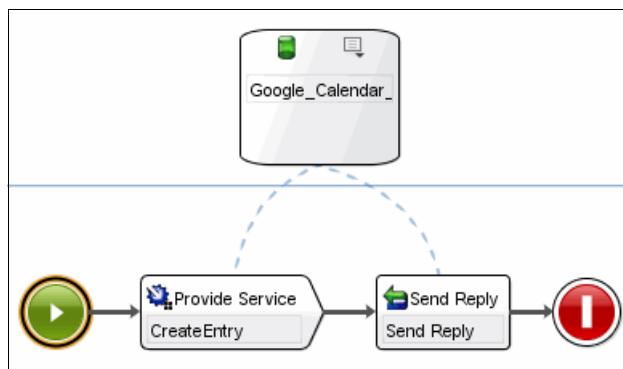


Figure 8-14 Initial CreateEntry orchestration

In this example, the orchestration is extended. The orchestration must log in to a Google account. It must then extract auth code from the login result and use it in a request to a Google account. The Google ReST API creates an entry in Google Calendar. The orchestration must be able to handle errors. It also must handle the case where on the first request to create the calendar entry a response is received with a redirect URL and sends the same request to the new URL.

The following steps outline the flow of the new orchestration:

1. The Web Service Provide Service starter activity receives the request to create an entry in Google Calendar.
2. The next series of steps are the same as those in the Test Connection orchestration. They log in to Google Calendar.
3. Extract the Auth code from the login response and save it in string variable.
4. Use the extracted Auth value in an HTTP Post Activity (CreateEntry) and the data received in the Provide Service starter activity to send a request to the ReST API to create an entry in Google Calendar.
 - a. If the request is redirected (HTTP response 302):
 - i. Extract the redirect URL from response.
 - ii. Resend the calendar entry request to the extracted URL.

- iii. If the entry is created, send the response to the Connector Activity.
 - b. If the calendar entry is created (no redirect URL), send a response to the Connector Activity.
 - c. If the calendar entry creation fails, send a response with a meaningful error message to the Connector Activity.
5. If any errors occur during the previous steps, send a meaningful error message to the Connector Activity.

Figure 8-15 shows the first steps of the orchestration.

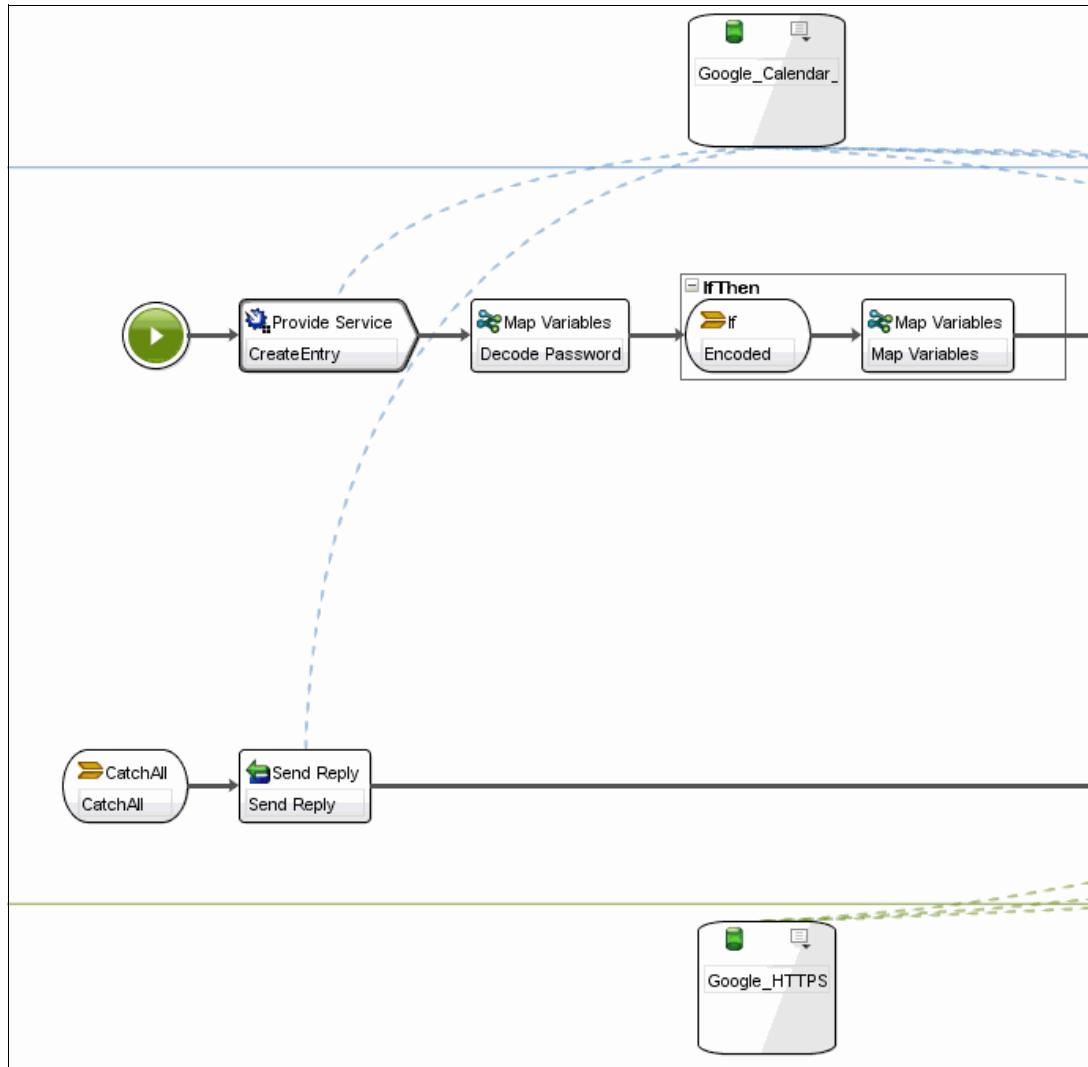


Figure 8-15 First steps in the orchestration

Figure 8-16 on page 341 shows the next steps.

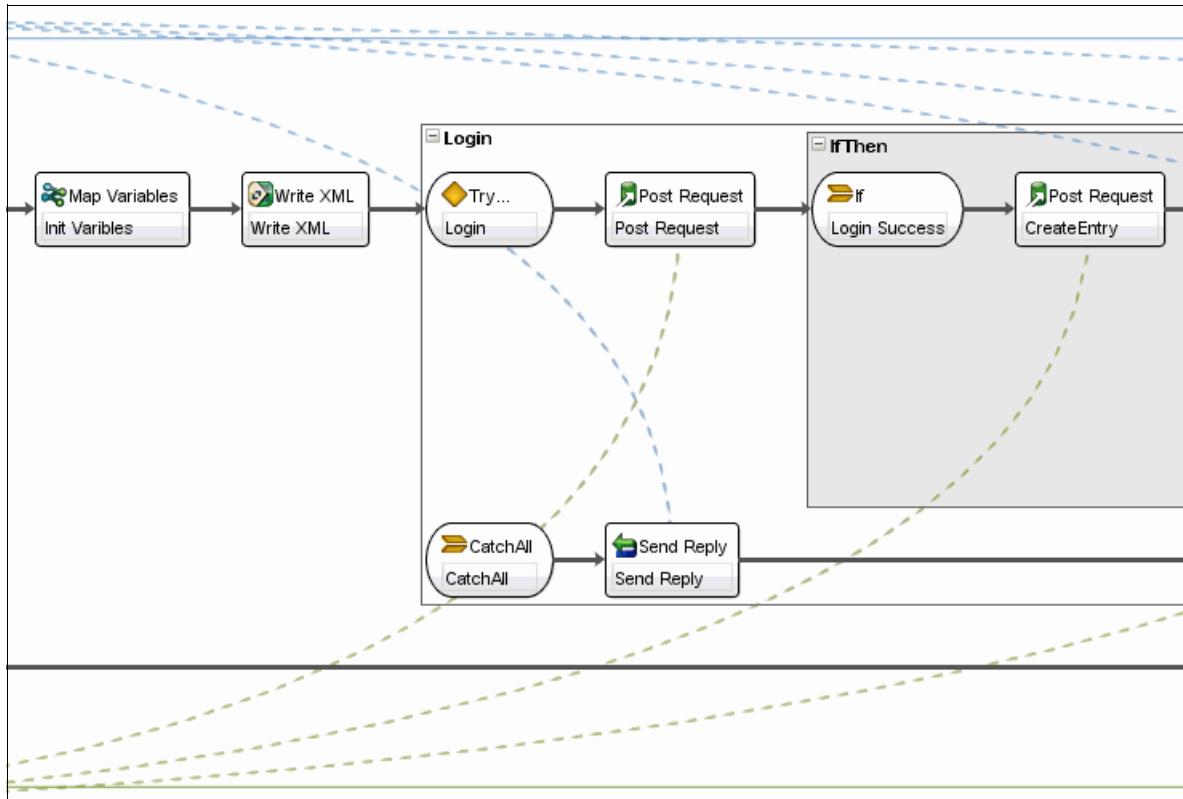


Figure 8-16 Next steps in the orchestration

Figure 8-17 shows the final steps in the orchestration.

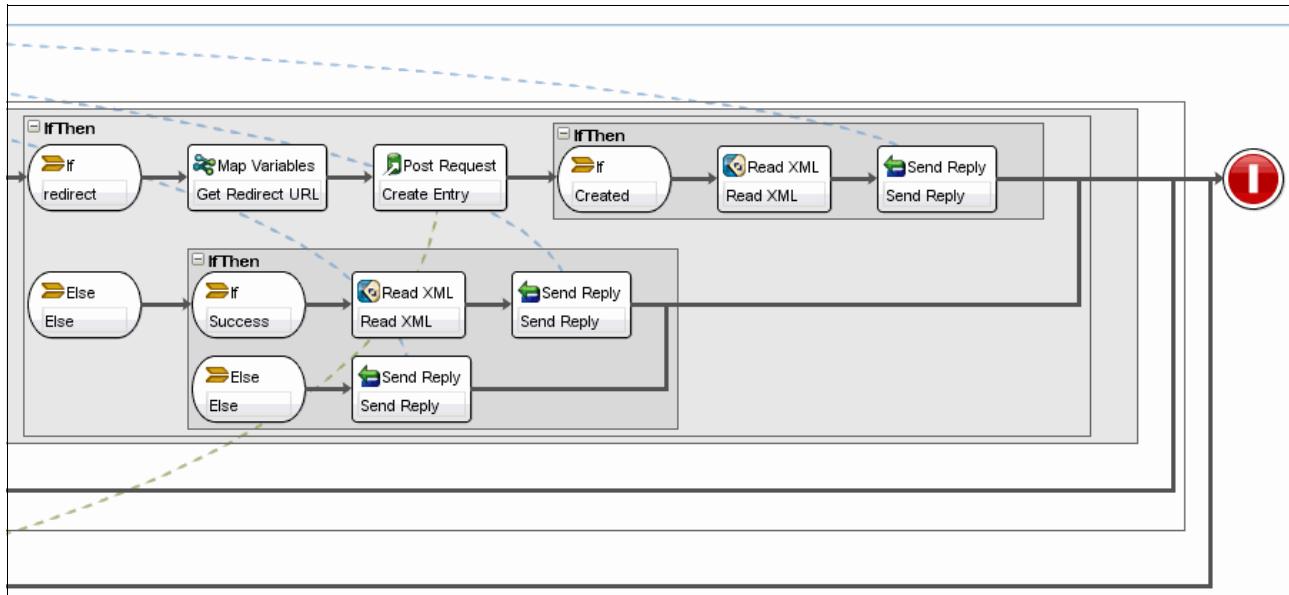


Figure 8-17 Final steps in the orchestration

8.4 Publishing the CDK Connector to a local repository

After you implement the orchestrations generated by the CDK Wizard, you are finished with the development of the connector. Now, you can publish the connector project to the local repository and the connector is ready to use.

The published connector is located in the /<home>/ .castiron/connector-repository. It includes the CDK connector components that were created using data that you entered in the CDK Wizard and the connector project.

Publish a connector project by clicking the Publish Connector icon () , or in Studio, select **File → Publish Connector to Local Repository**.

8.5 Testing and debugging the CDK Connector

You can test the connector at different stages of development. Testing is the only way to validate that the activities and APIs of the connector are functioning as expected. Testing also helps you to determine and debug configuration and data errors.

The CDK offers the following types of testing:

- ▶ Unit testing

Unit testing is typically done for each orchestration in the connector when the development is completed. Each orchestration is an implementation of either an activity of the connector or any of the three APIs. Use the Invoke Service Tool for unit testing, as described in 3.12.9, “Invoke Service tool” on page 169.

- ▶ Integration testing

After the development and unit tests are complete, the connector is published to the local repository and is ready to use. Integration testing is used for functional verification of the connector activities. Optionally, you can display the steps inside an activity, in this case an orchestration, in the Verify tab of Studio to keep track of the internal data flow. This display of information is useful for debugging when the unit test passes but the integration test fails.

Enable the display of all processes of the activity by selecting **Show CDK Connector Activity Logs** in the Orchestrations tab of Studio preferences, as shown in Figure 8-18. To open the preferences select **Edit → Preferences → Orchestration**.

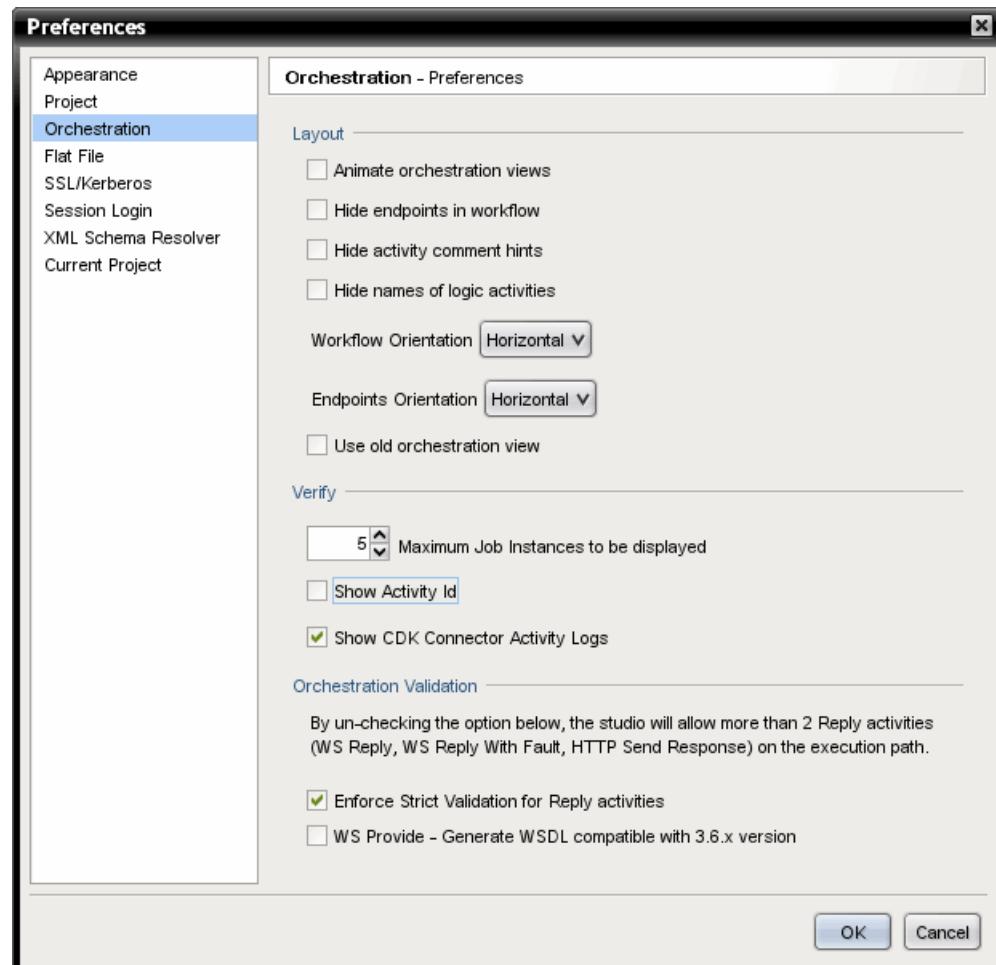


Figure 8-18 Enable the CDK Activity execution log

You can perform an integration test on the CreateEntry activity of the Google Calendar connector, which you developed in 8.3.2, “Developing a Google Calendar connector using the CDK Wizard” on page 333, with the following steps:

1. Create a new project in Studio. In the Studio Menu, select **File → New → New Project**.
2. Create a new orchestration and select it. The orchestration is displayed in Studio Workspace.
3. Drag the CreateEntry activity of the Google Calendar connector from the Activity toolbox to the Orchestration, as shown in Figure 8-19.



Figure 8-19 CreateEntry Activity added to the orchestration

4. In the Studio workspace, click the activity to highlight it.
5. Select **Pick Endpoint** in the checklist panel to open the Pick Endpoint panel. Click **New**.
6. Enter the connection details, as shown in Figure 8-20, and verify the connection by clicking **Test Connection**.

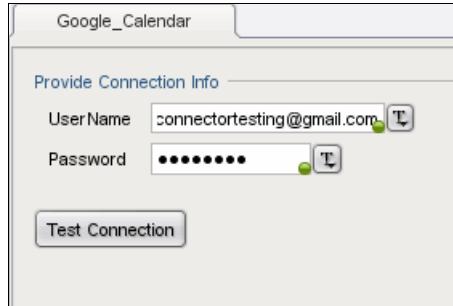


Figure 8-20 Google Calendar Endpoint panel

7. Complete the input mapping by entering test data as the default values to the required fields.
8. Enable the CDK Activity execution log, as shown in Figure 8-18 on page 343.
9. Right-click the activity, and select **Verify Activity**. Figure 8-21 shows internal data flow of the activity as displayed in the Verify tab of Studio.

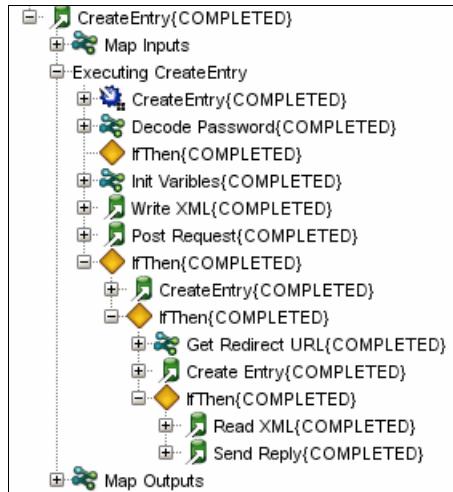


Figure 8-21 CreateEntry Activity Execution Flow

8.6 Sharing the CDK Connector

After the development, testing, and publishing of the connector is complete, you can export the connector to an external location for others to use. Similarly, you can import other connectors into Studio. IBM-developed CDK Connectors are hosted in a server that you can download connectors from into Studio, provided that you have Cast Iron community login credentials.

The next section explains how to import, export, and download a connector.

8.6.1 Exporting a CDK Connector

After publishing a connector, you can use the connector. However, for others to use your connector, you must export it to a location where it can be imported from by others. You can also export IBM-developed connectors that are downloaded from the Cast Iron Community (select **Solutions** → **Plugin Connectors**).

To export the Google Calendar connector:

1. Open Studio, and launch the Export CDK Connector window by clicking the Export icon ( or in Studio, by selecting **File** → **Export Connector**).
2. From the drop-down list, select the Connector to export.
3. Select Google Calendar, and click **OK**, as shown in Figure 8-22.



Figure 8-22 Export CDK Connector selection

4. Select the location, and click **Save**.
5. In the confirmation window, click **OK**.

8.6.2 Importing a CDK Connector

You can use a connector that was developed by others or by IBM by importing it into Studio.

To import the Google Calendar connector:

1. Open Studio and launch the Import Connector window by clicking the Import icon ( or in Studio by selecting **File** → **Import Connector**).
2. Select the archive file of the Google Calendar connector, and then click **Select Connector**.
3. In the confirmation window, click **OK**. Now the Google Calendar connector is ready to use.

8.6.3 Downloading a CDK Connector

You can install and update IBM-developed CDK Connectors in Studio using the Plugin Connectors window:

1. Make sure that you have a community ID. If you do not have a community ID, you can request one at:
<http://community.castiron.com>
2. Log in to the community by clicking **Login**, as indicated in Figure 8-23 on page 346.

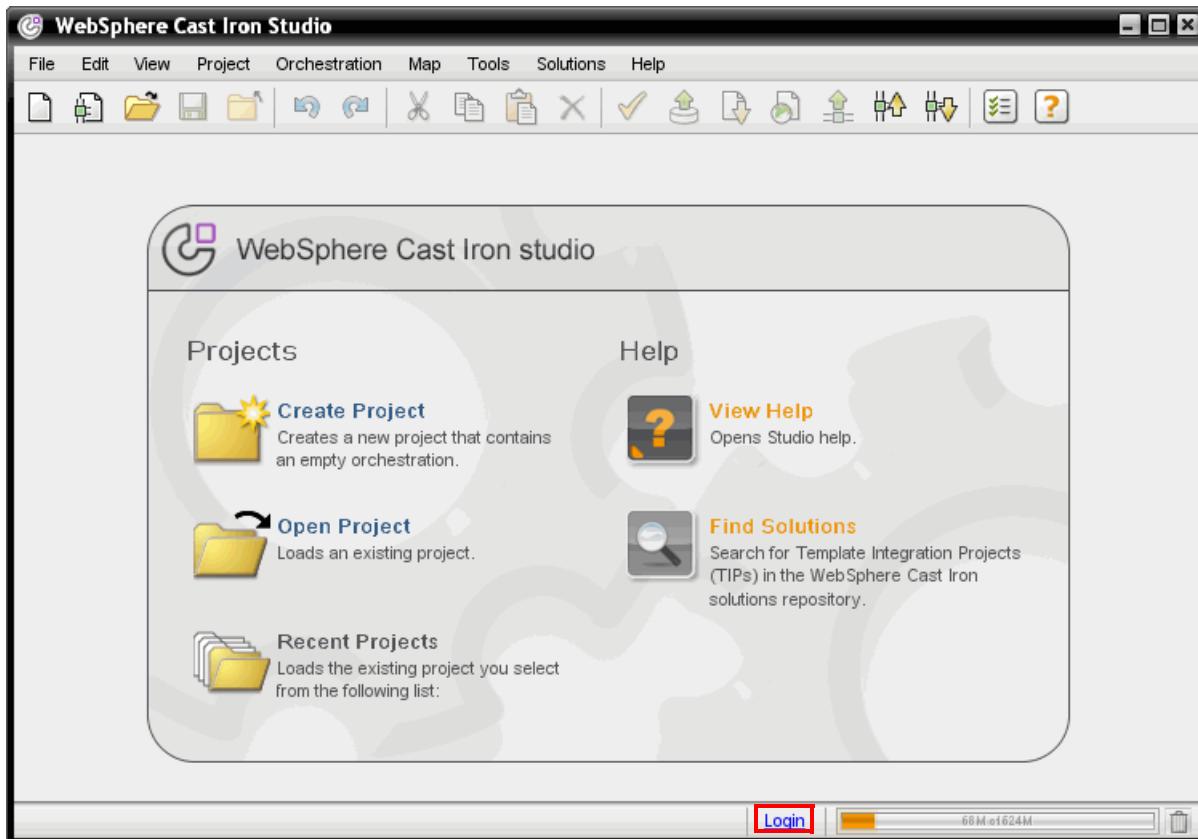


Figure 8-23 Login option

3. Enter the community user name and password and click **OK**.

4. Launch the Plugin Connectors window from Studio Menu by selecting **Solutions** → **Plugin Connectors**. Figure 8-24 shows the window that opens.

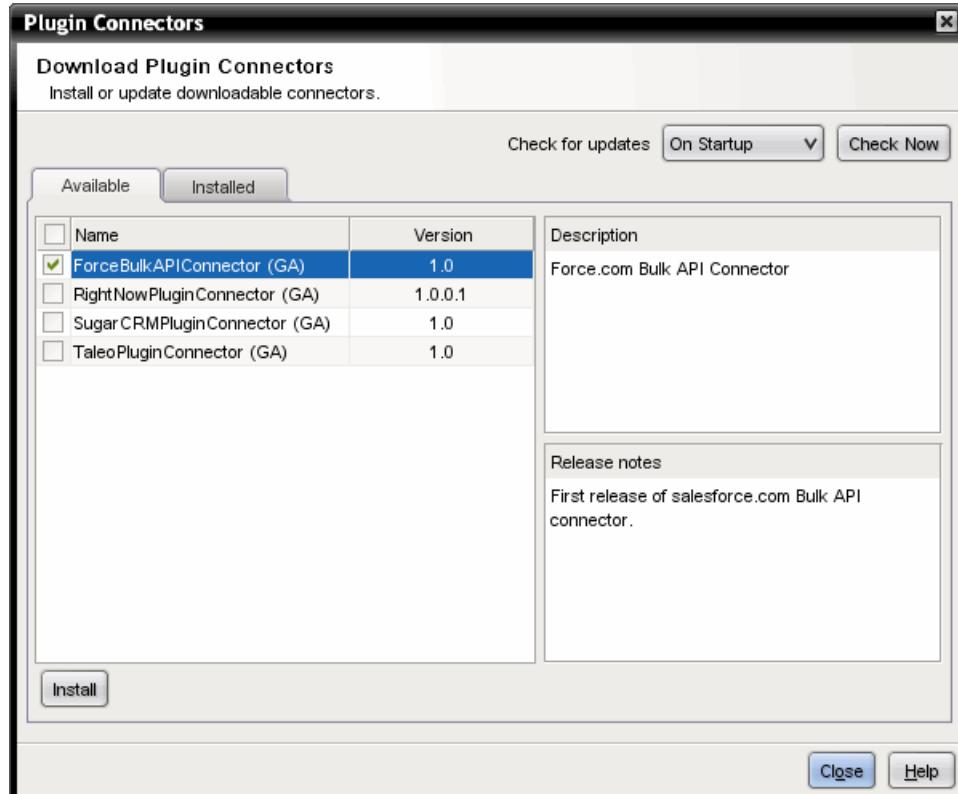


Figure 8-24 Plugin Connectors window

Use the following sections in this window to manage CDK Connectors:

- The Available tab, shown in Figure 8-25, lists all of the CDK Connectors that are available to install. To install a connector or connectors, select the required connector or connectors, and then click **Install**.

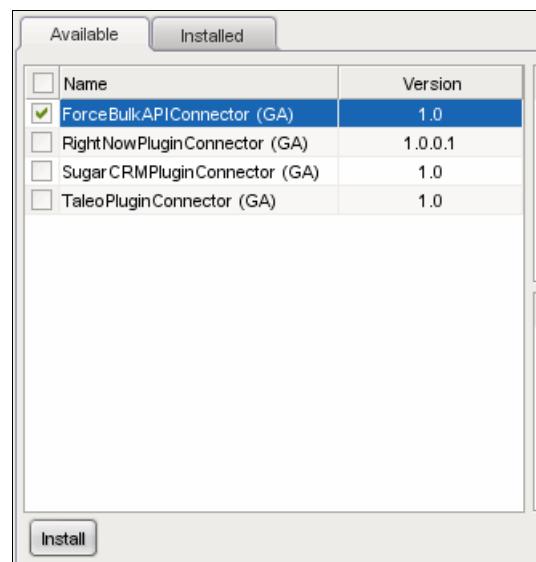


Figure 8-25 Available tab

- The Installed tab, shown in Figure 8-26, lists all of the connectors that are installed in Studio, including the connectors that are published to the local repository. You can update or uninstall a connector or connectors. Select the connector or connectors, and then click either **Update** or **Uninstall** as appropriate.

Name	Version
ForceBulkAPIConnector (GA)	1.0
Google_Calendar	1.0.0

Figure 8-26 Installed tab

Using the Installed tab: Note the following information about the Installed tab:

- The Update button is enabled only if the selected connector has an update.
- Newly created connectors, such as the Google Calendar connector, are not hosted in the server. However, these connectors are listed in the Installed tab because they are published to the local repository.
- The Description field includes a brief overview of the selected connector.
- The Release notes field specifies the changes that were part of the current version of the selected connector.
- The Check for updates field allows you to query updates for connectors based on a time line, which is configurable. By default, this field is set to On Startup. You can change the value to Daily, Weekly, Alternate Weeks, Monthly, or Never by selecting the connector in the drop-down menu of the field. You can check the updates instantly by clicking **Check Now**.



Common error handlers

This chapter describes how to use an orchestration for handling errors in the same manner across multiple orchestrations and how a common error handler orchestration can be used by the other orchestrations.

It also gives details of an example common error handler project. We created this project for this book and uploaded it to the Cast Iron Solutions Repository.

This chapter discusses the following topics:

- ▶ The principles of a common error handler
- ▶ The common error handler Template Integration Project
- ▶ Altering the common error handler TIP
- ▶ Using the common error handler TIP

9.1 The principles of a common error handler

Error handling is important for most orchestrations. An error handler can be created to fix specific errors that occur only in one orchestration; however, it is often useful to create an error handler that can be called by more than one orchestration. This common error handler can contain any error functionality that is the same in every orchestration that uses it. A common error handler might not be suitable for every orchestration to use, but it can simplify and standardize error handling where it can be used.

Common error handlers are created as separate orchestrations and published to the same Cast Iron runtime as the orchestrations that use them. They are usually created in their own project so that changes to the error handler can be made without having to undeploy and redeploy the orchestrations that use the error handler. Figure 9-1 shows four orchestrations that are all running on the same Cast Iron runtime, calling a common error handler (using web services).

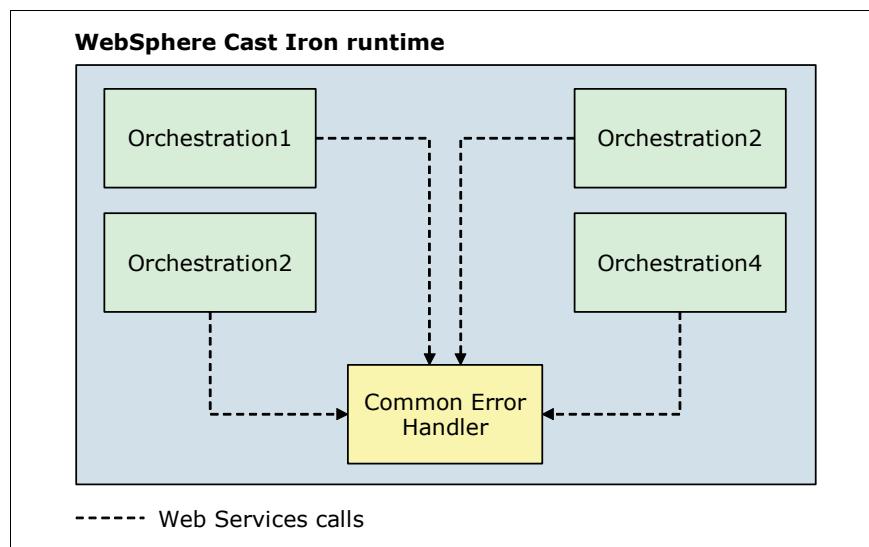


Figure 9-1 Calling a common error handler

9.2 The common error handler Template Integration Project

An example of a common error handler was created for this book. The common error handler was uploaded as a Template Integration Project (TIP) to the Cast Iron Solutions Repository. TIPs and the Cast Iron Solutions Repository are described in Chapter 7, “Reusability with Template Integration Projects” on page 293. To find the common error handler TIP, search in the Solutions Repository for the keyword `commonerrorhandler`. The remainder of this chapter gives details of the functionality that is provided by this common error handler TIP and how to use it.

The project inside the common error handler TIP contains an orchestration called `CommonErrorHandlerOrchestration`. This orchestration is called using web services and receives the `JobInfo` and `faultInfo` data for the error that has occurred from the orchestration in which the original error occurred. The error handler writes this information to a file on an FTP server. The file is created with a name that includes a time stamp using the following form:

`ErrorLogyyyMMddHHmmssSS.txt`

Figure 9-2 shows the common error handler orchestration. The endpoint WebService_In has a Type of Provide and a Transport of HTTP. The Port and Path fields use the following configuration properties:

ErrorWSPort	The port on which the web service listens
ErrorWSPath	The URI that is used to call the web service, set to <i>ErrorLoggerRemote</i> by default in the TIP

Tip: If the common error handler will be deployed to Cast Iron Live, the Type of this endpoint must be changed to Cast Iron Cloud. The port is then automatically set.

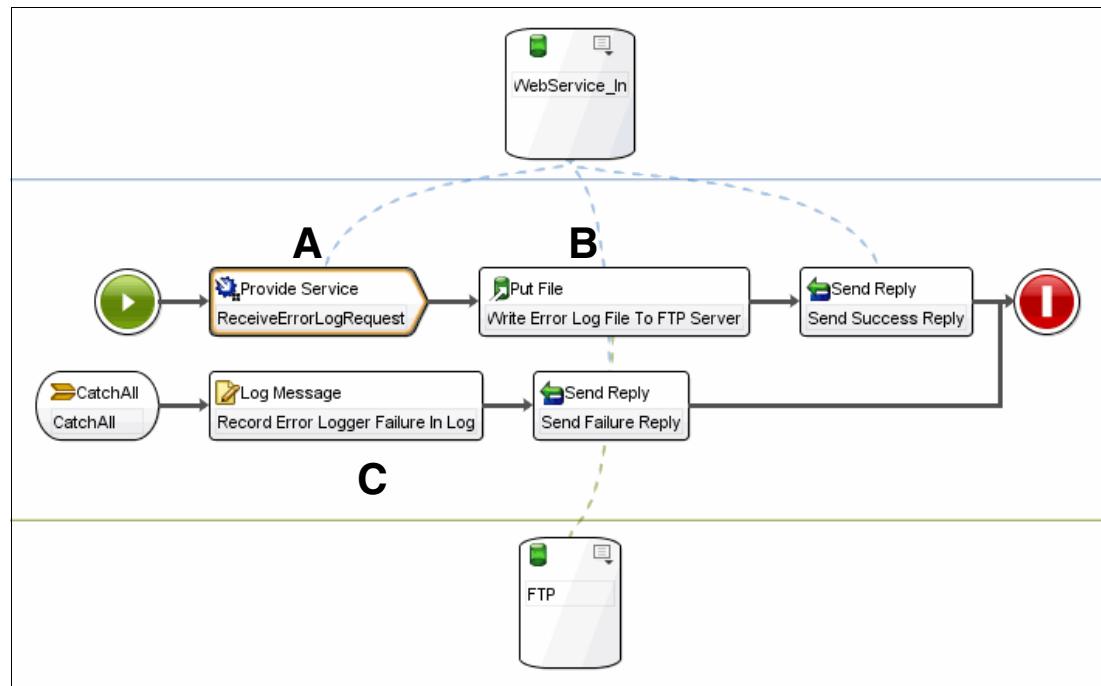


Figure 9-2 The common error handler orchestration

The project also contains an FTP endpoint that is used by an FTP Put File activity in the orchestration. The FTP endpoint holds the connection properties to the FTP server where the error log files are written. The endpoint uses defaults for all settings apart from the Host Name, User Name, and Password, which use the following configuration properties:

ErrorFTPServerIP	The IP address of the FTP server that will be used
ErrorFTPServerID	The user ID to connect to this FTP server
ErrorFTPServerPW	The password associated with this user ID

In the orchestration in Figure 9-2, the Provide Service activity (A) copies its input data to an output variable called *commonErrorHandlerData*. The activity is configured to require a response. The two associated Send Reply activities send a success or failure string message back to the calling orchestration. The activity has an Activity Name of *ReceiveErrorLogRequest*.

Tip: The Activity Name of the Provide Service activity is used to set the operation name for this orchestration when it is called using web services.

The Put File activity (B) sets the Save In Directory field to a configuration property called ErrorFTPPutDirectory. The activity sets the file name of the file to be created using the following built-in functions, one after the other:

1. Get Current Date and Time
2. Format Date String
3. Concatenate

The Put File activity also uses the custom function shown in Example 9-1 to set the data that is written to this file.

Example 9-1 Custom function to create the data written to the error file

```
function formatErrorData(jobId, jobStartTime, projectName, configurationName,
orchestrationName, routerHostName, errorName, errorMessage, activityId,
activityName, faultTime) {
    return 'Job ID: ' + jobId
    + ' - Job Start Time: ' + jobStartTime
    + ' - Project Name: ' + projectName
    + ' - Configuration Name: ' + configurationName
    + ' - Orchestration Name: ' + orchestrationName
    + ' - Router Host Name: ' + routerHostName
    + ' - Error Name: ' + errorName
    + ' - Error Message: ' + errorMessage
    + ' - Activity ID: ' + activityId
    + ' - Activity Name: ' + activityName
    + ' - Fault Time: ' + faultTime;
}
```

The orchestration also includes a Global Exception Handler, as described in 3.11.2, “Global exception handler” on page 158. The Log Message activity (C) in the Global Exception Handler writes a Critical log message to the log with the following message:

The Common Error Handler has failed.

9.3 Altering the common error handler TIP

The following examples shows how you can alter the common error handler TIP for different functional requirements:

- ▶ You can replace the FTP endpoint and activity to allow the errors to be recorded in any system that can be accessed through the Cast Iron runtime, for example a database or a separate logging system with a web services interface.
- ▶ You can include specific functionality to isolate particular error types, endpoints, or source orchestrations by adding in an If..Then activity with specific activities after each condition.
- ▶ Because the common error handler is running as a separate orchestration it will have its own job keys. Thus, you can add a Create Job Key activity to allow particular information about the error handled to be shown in the Web Management Console.

9.4 Using the common error handler TIP

The following steps describe how to use the common error handler in an existing orchestration:

1. Publish the common error handler project to a Cast Iron runtime.
2. Log in to the Web Management Console for the Cast Iron runtime where you deployed the common error handler. Update the configuration properties to provide appropriate values for the FTP server that will be used.

Also update the configuration properties for the web services port and paths so that they do not conflict with anything else that is running on this Cast Iron runtime.

3. To use the common error handler, the calling orchestration needs a WSDL file that describes the common error handler web service. You can download this WSDL file from the Web Management Console. To download this file:
 - a. Select the Configuration Details pane for the common error handler configuration. Click **WebService_In** in the Assets section, as shown in Figure 9-3, to open the Web Services Assets window.



Figure 9-3 The Assets section of the common error handler configuration details

- b. Click **ReceiveErrorLogRequest** to show the generated WSDL that is associated with the common error handler web service.
 - c. Click **Download** at the bottom of the window to download the WSDL file to a local machine. The WSDL is given a default name of WebServicesWSDL.wsdl1. Change this default name to make it clear that this WSDL file is the WSDL file for the common error handler, for example CommonErrorHandler.wsdl1.
 - d. Close the Web Services Assets window.
4. Start the common error handler configuration from the Home or Repository panes of the Web Management Console.
5. Open the existing orchestration that will call the common error handler in the Studio, and add the WSDL file that you downloaded to the project:
 - a. Select the Project tab.
 - b. Right-click **WSDLs**, and then select **Add Document**.
 - c. Browse to find the WSDL file and click **OK**.
6. Create a Web Services endpoint in the project that contains the calling orchestration.
 - a. In the Project toolbox tab, right-click **Endpoints**, and select **Create Endpoint → WebService**.
 - b. Select **Remote** for the Invoke Location drop-down menu field.
 - c. Browse to find the WSDL file that is associated with the common error handler. Selecting the WSDL file sets all other properties.
 - d. Create the Location text field property as a configuration property, for example CommonErrorHandlerLocation as highlighted in Figure 9-4 on page 354.

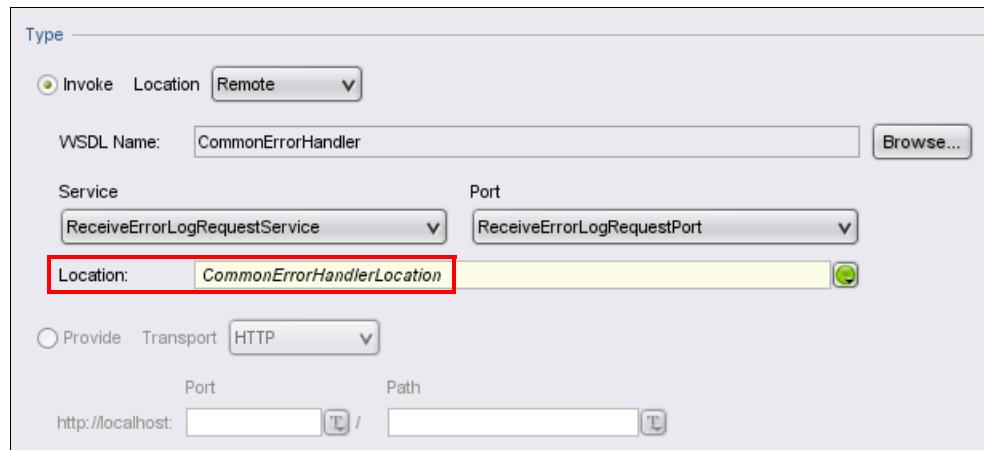


Figure 9-4 The calling orchestration web services endpoint settings

- e. Close the Web Service settings.
- 7. Set the value of the configuration property that contains the web service Location to the location of the remote common error handler orchestration. In Figure 9-4, the configuration property is called CommonErrorHandlerLocation. Click **Project** → **Configuration Properties**. Use the following format for the Location:

`http://CastIronVM:CastIronPort/RemoteURI`

Where:

CastIronVM	The IP address of the Cast Iron runtime where the common error handler was deployed. This must be the data IP address for the Cast Iron runtime and not the management IP address.
CastIronPort	The Port set for the common error handler remote orchestration, defined in the ErrorWSPort configuration property of the common error handler TIP.
RemoteURI	The URI set for the common error handler remote orchestration, defined in the ErrorWSPPath configuration property of the common error handler TIP.

Tip: If the common error handler was deployed to Cast Iron Live, the Location must be set to the correct URL format for an HTTP request that is sent to Cast Iron Live. The Cast Iron Live HTTP request URL format is described in 3.6.8, “Using HTTP and web service starter activities with Cast Iron Live” on page 116.

The URL will be of the form:

`https://<Hostname>:443/env/<Environment>/<RequestURI>`

For example, a complete URL can be:

`https://provide.castiron.com:443/env/Development/ErrorLoggerRemote`

- 8. Use this web services endpoint to create an Invoke Service activity within the orchestration at the point where the common error handler is called. The inputs for this activity are the JobInfo variable and a variable that includes the faultInfo data. Map these two variables to the corresponding input variables of the activity.

9. Test the orchestration in the Studio. If the Studio has network access to the Cast Iron runtime where the common error handler is running, the orchestration cannot invoke the common error handler in case of error.
10. Publish the project that contains the orchestration to the same Cast Iron runtime on which the common error handler orchestration is running. The orchestration will now use the CommonErrorHandlerOrchestration for its common error handling.

Examples of using the common error handler are included in the scenarios in this book. The common error handler is used to provide error handling in the following scenarios:

- ▶ Chapter 10, “Scenario: Bidirectional account synchronization” on page 357
- ▶ Chapter 11, “Scenario: CRM to cloud calendar services” on page 435
- ▶ Chapter 12, “Scenario: Data enrichment and aggregation” on page 465



Scenario: Bidirectional account synchronization

This chapter provides an example integration to synchronize account records between an enterprise resource planning (ERP) system (in this example, SAP) and a customer relationship management (CRM) system (in this example, Salesforce.com). It includes scenarios that demonstrate this synchronization in both directions.

This chapter includes the following topics:

- ▶ Cast Iron concepts demonstrated in this scenario
- ▶ Scenario overview
- ▶ Prerequisites
- ▶ Overview of the use cases for this scenario
- ▶ Use case 1: Synchronizing from SAP to Salesforce.com
- ▶ Use Case 2: Synchronizing from Salesforce.com to SAP

Disclaimer: The intent of this tutorial is to demonstrate the use of IBM WebSphere Cast Iron Cloud Integration to integrate and utilize third-party applications and does not claim any endorsement or affiliation with the listed products. All product and company names are trademarks or registered trademarks of their respective holders, and IBM disclaims any ownership in such third-party marks. Use of such third-party marks does not imply any affiliation with or endorsement by or for IBM or IBM WebSphere Cast Iron Cloud Integration. The use of any third-party trademarks, logos, or brand names is for informational and instructional purposes only, and does not imply that such trademark owner has authorized IBM to promote its products or services.

10.1 Cast Iron concepts demonstrated in this scenario

This chapter demonstrates the use of the following Cast Iron concepts:

- ▶ Accessing and using a Template Integration Project (TIP)
- ▶ Using configuration properties
- ▶ Using the connectors for HTTP, SAP, and Salesforce.com
- ▶ Using a Pick activity to allow multiple starter activities
- ▶ Using Filters to extract data and work on subsets of data
- ▶ Handling errors using Try-Catch with Continue, If-Then, and global catch
- ▶ Logging errors using Log Message and the Common Error Handler
- ▶ Testing parts of the an orchestration using terminate
- ▶ Avoiding feedback loops

10.2 Scenario overview

A company has an SAP¹ ERP system and has just bought Salesforce.com CRM as their CRM system in the cloud. The CRM system is intended to be used by the call center and the sales teams to communicate with their customers. To work with the CRM system, account data is needed. This account data is available in SAP but has to be transferred into Salesforce.com¹.

The company made the following decisions regarding data management:

- ▶ New customer accounts will be created in SAP.
- ▶ Users in Salesforce.com will be allowed to modify customer data such as addresses.

IBM WebSphere Cast Iron Cloud Integration is used to keep the data of the two systems synchronized.

This scenario implements the following scenarios:

- ▶ Synchronizing new and updated customer data from SAP into Salesforce.com
- ▶ Synchronizing updated account data from Salesforce.com into SAP

Figure 10-1 illustrates the company's need to synchronize data to and from SAP and Salesforce.com.

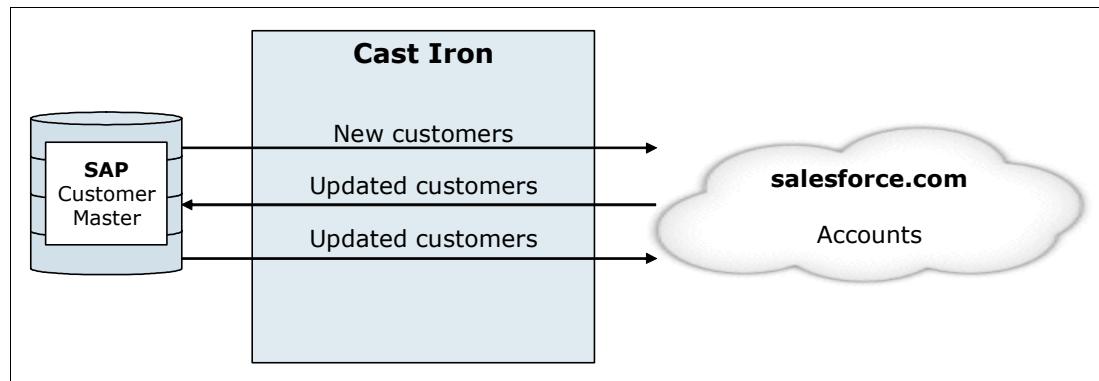


Figure 10-1 Use case scenario

¹ See Disclaimer on page 357.

Tip: This scenario refers to companies as both customers and accounts as follows:

- ▶ In SAP, the companies are called *customers*.
- ▶ In Salesforce.com, the customers are called *accounts*.

10.3 Prerequisites

To build and test the orchestrations in this scenario, the following requirements must be met:

- ▶ WebSphere Cast Iron Studio

Cast Iron Studio Version 6.1.0.1 is already installed. See Chapter 2, “Installing and setting up WebSphere Cast Iron integration” on page 27 for more information.

- ▶ WebSphere Cast Iron appliance

The scenario includes the Common Error Handler, as discussed in Chapter 9, “Common error handlers” on page 349. You must have access to an Integration Appliance that is running the error handler.

- ▶ Cast Iron Template Repository

The scenario uses a TIP from the Cast Iron template repository. You must have a Cast Iron community user to access the repository.

- ▶ SAP System and User

You must have access to an SAP system with valid customers.

You must have an SAP account. The user for the account must be able to read the customer data from SAP and be allowed to update customer data in SAP.

- ▶ SAP Libraries

- You must have the SAP JCO libraries installed in Cast Iron Studio.

- If you want to test your orchestration on the appliance, the SAP libraries must be installed there too.

For information about installing these libraries, see Chapter 2, “Installing and setting up WebSphere Cast Iron integration” on page 27.

- ▶ A Salesforce.com account

You must have a Salesforce.com account. The user ID must be able to read account data from Salesforce.com and to create and update account data in Salesforce.com.

In Salesforce.com, you must define the following fields for the Account object:

- An external ID field of type text and an external ID named *ExtAccountID* or similar. The external ID contains the SAP customer number in a format of SAP_xxx later on.
 - An external field of type text that is named *SalesOrganization* or something similar.

To test feedback loops, you must have another Salesforce.com user in the same Salesforce.com environment with the same rights. In that scenario, one user represents the sales people or call center assistant working with Salesforce.com, and the other user is for integration purpose.

Tip: The Salesforce.com password that this scenario uses for Cast Iron is a combination of the login password and a security token. Before you work through the steps listed in 10.6, “Use Case 2: Synchronizing from Salesforce.com to SAP” on page 407, request a security token from the Salesforce.com portal.

10.4 Overview of the use cases for this scenario

This chapter explains how to build two orchestrations to synchronize accounts between SAP and Salesforce.com:

- ▶ Use case 1: Synchronizing from SAP to Salesforce.com

This use case provides an initial load of the existing customers and updates from SAP into Salesforce.com. It uses an existing orchestration that is available as a Template Integration Project (TIP) and that is accessible in the Cast Iron Template Repository. The repository is available for every Cast Iron customer.

Use case 1 adjusts the orchestration slightly to fit to the SAP and Salesforce.com systems.

- ▶ Use Case 2: Synchronizing from Salesforce.com to SAP

This use case is built from scratch but reuses the endpoints that were created for scenario 1.

Each use case starts by building simple orchestrations and then enhances the orchestrations with additional functionality for improved performance, for error handling, and to avoid feedback loops. Both use Cast Iron Studio for modelling and testing the orchestrations. The Cast Iron Hypervisor Edition is the runtime for these orchestrations.

You can use the physical appliance or Cast Iron Live as the runtime environment instead. If using Cast Iron Live, you must define a secure cloud connector and configure the SAP endpoint to use it.

10.5 Use case 1: Synchronizing from SAP to Salesforce.com

This first scenario implements an integration between SAP and Salesforce.com.

Triggered by an HTTP request, several customer records are read from SAP and stored in Salesforce.com. Account synchronization between SAP and Salesforce.com is a typical customer scenario; therefore, templates are available for this type of scenario in the template repository. Instead of creating a project from scratch, this scenario downloads an existing TIP and customizes it accordingly.

10.5.1 Downloading the TIP

Begin by downloading the TIP:

1. Start WebSphere Cast Iron Studio.
2. Click **Solutions** → **Search for TIPs**.

If not already logged in, you are prompted to log in to the Cast Iron community. After a successful login, the template repository panel opens.

3. In the Search window select **Filter**. Select **SAP** as the source endpoint and **Salesforce.com** as the target endpoint. Click **Query**, as shown in Figure 10-2 on page 361. A list of available templates displays in the Results section.

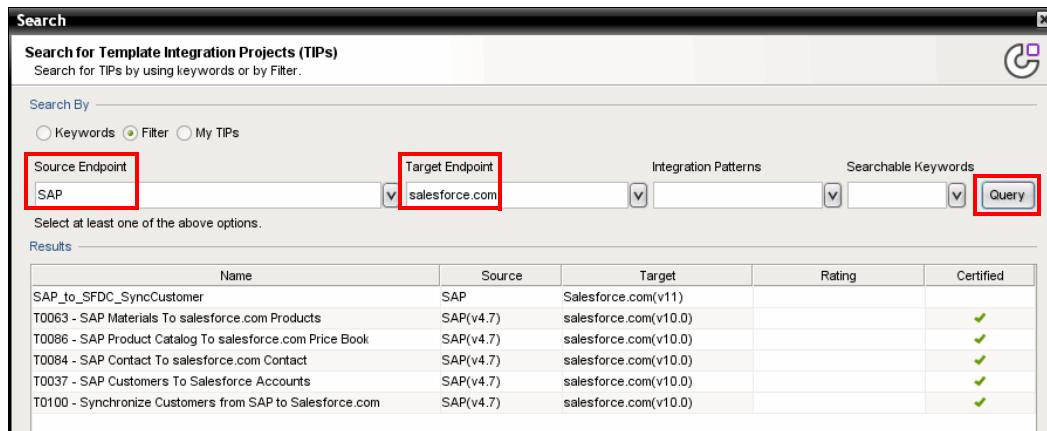


Figure 10-2 Selecting the source and target endpoints

4. Several TIPs are available for SAP to Salesforce.com integration. Select **T0037**, and read the description, as shown in Figure 10-3.

T0037 - SAP Customers To Salesforce Accounts	SAP(v4.7)	salesforce.com(v10.0)			✓
T0100 - Synchronize Customers from SAP to Salesforce.com	SAP(v4.7)	salesforce.com(v10.0)			✓

Details

Project Name: T0037_SAPCustomerToSFDCAccounts
 Creator: Cast Iron Systems
 Created: 9/10/08 6:57 PM
 Path: /tips
 Category: USECASES
 Avg. Rating: ☆☆☆☆☆ (Not rated)

Patterns:
 Project Size(Kb): 31
 Last Modified: 10/16/08 4:20 PM
 Created With Studio Version: 4.0.1
 Downloaded: 195
 Version: 1.0

Description

This TIP searches for customers in SAP and upserts the data in salesforce.com.

Figure 10-3 TIP T0037 description

5. Click **Download**, and save the TIP to a place of your choice. A warning displays, as shown in Figure 10-4. Click **Yes** to open the TIP anyway.

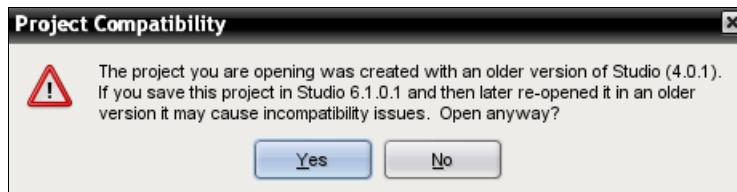


Figure 10-4 TIP version warning message

10.5.2 Using the TIP configuration wizard

Now that you downloaded the TIP, you can use the configuration wizard to customize the TIP for this scenario. The configuration wizard opens with an introduction panel, as shown in Figure 10-5 on page 362.

The steps that you must perform to use the TIP display on the left side of the window. Text that describes the template's actions display on the right side of the window. Click **Next**.

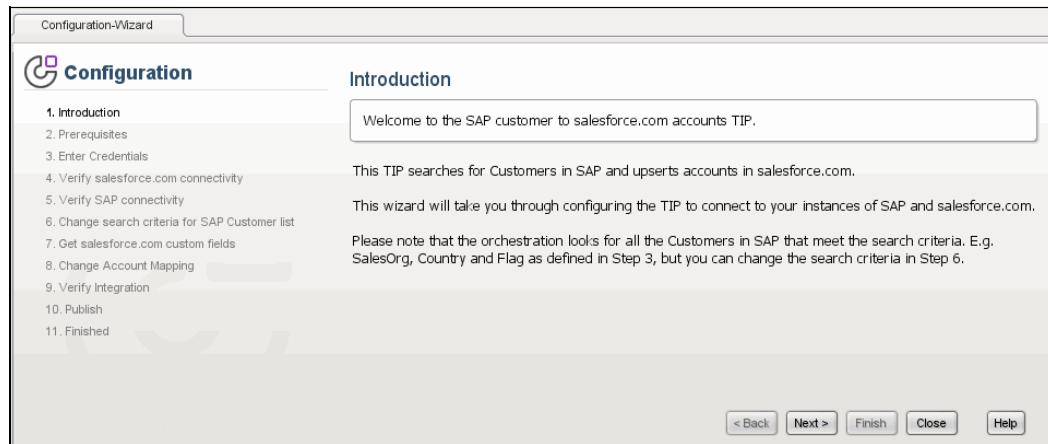


Figure 10-5 TIP wizard step 1: Introduction

In step 2, review the TIP requirements. The first two prerequisites listed in Figure 10-6 are already met. This scenario does not use the ExtSource field, so you can ignore the third requirement. Click **Next** to continue to step 3.

Prerequisites

Install the SAP client libraries and create custom fields in salesforce.com to hold SAP information.

Before going further, you need to install the SAP client libraries and create two custom fields in Salesforce to hold the SAP data:

1. Install the SAP client libraries in Cast Iron Studio and on your appliance (if you plan on running the TIP there). Obtain the SAP client libraries from your SAP administrator.
2. ExtAccountID__c - Text(25) (External ID) - This is the external Id to store the SAP customer number in salesforce.com. The orchestration upserts accounts salesforce.com using this id.
3. ExtSource__c - Text(15) - This field holds the default "SAP" value indicating that this data came from an external source system.

Figure 10-6 TIP wizard step 2: Prerequisites

The following sections describe the remaining steps of the configuration wizard.

Entering the credentials

In step 3, the panel displays several configuration properties for the endpoints, such as credentials and search criteria, as shown in Figure 10-7 on page 363.

Enter Credentials

SFDCPassword: Salesforce Password
SFDCUsername: Salesforce Username
SAP Host: SAP Server Hostname or IP
SAP Sys: SAP System#
SAP Client: SAP Client#
SAP User: SAP Username
SAP Password: SAP Password
SAP SalesOrg: SAP Sales Organisation ID (Used as search criterion, **Example:** 3020)
SAP Country: Country (Used as search criterion, **Example:** US)
SAP Search Flag: Flag (Used as search criterion, **Example:** 2)

Name	Type	Value
SFDCPassword	Password	●●●●●●●●
SFDCUsername	String	user@name.com
SAPHost	String	SAPServer
SAPSys	String	00
SAPClient	String	000
SAPUser	String	SAPUser
SAPPassword	Password	●●●●●
SAPSalesOrg	String	0000
SAPCountry	String	US
SAPSearchFlag	String	2

Import

Figure 10-7 TIP wizard step 3: Enter Credentials

Enter the following information:

1. Enter your credentials for SAP and Salesforce.com. The credentials for Salesforce.com are the password concatenated with the security token. If you have two users for Salesforce.com (one for sales and one for integration), use the integration user.
2. Adjust the other settings, for example the SAP Country and the SAP Sales Organization, to fit your environment.
3. After you enter this information, click **Next** to move to the next step in the wizard.

Importing global configuration properties: If you created global configuration properties for some of these values, click **Import** to open the Import Properties panel, shown in Figure 10-8. Select the source and target fields for a value, and then click **Value** to import those configuration. When you are finished, click **OK**.

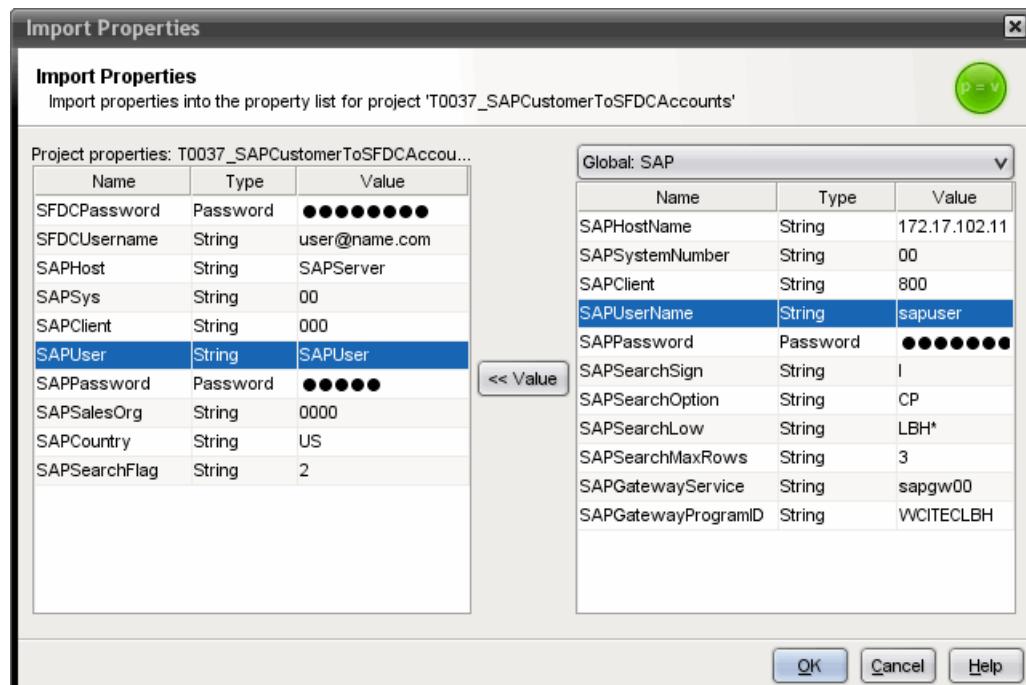


Figure 10-8 Import properties

Verifying Salesforce.com connectivity

In step 4, you verify the connection to Salesforce.com. Figure 10-9 shows the endpoint configuration for Salesforce.com.

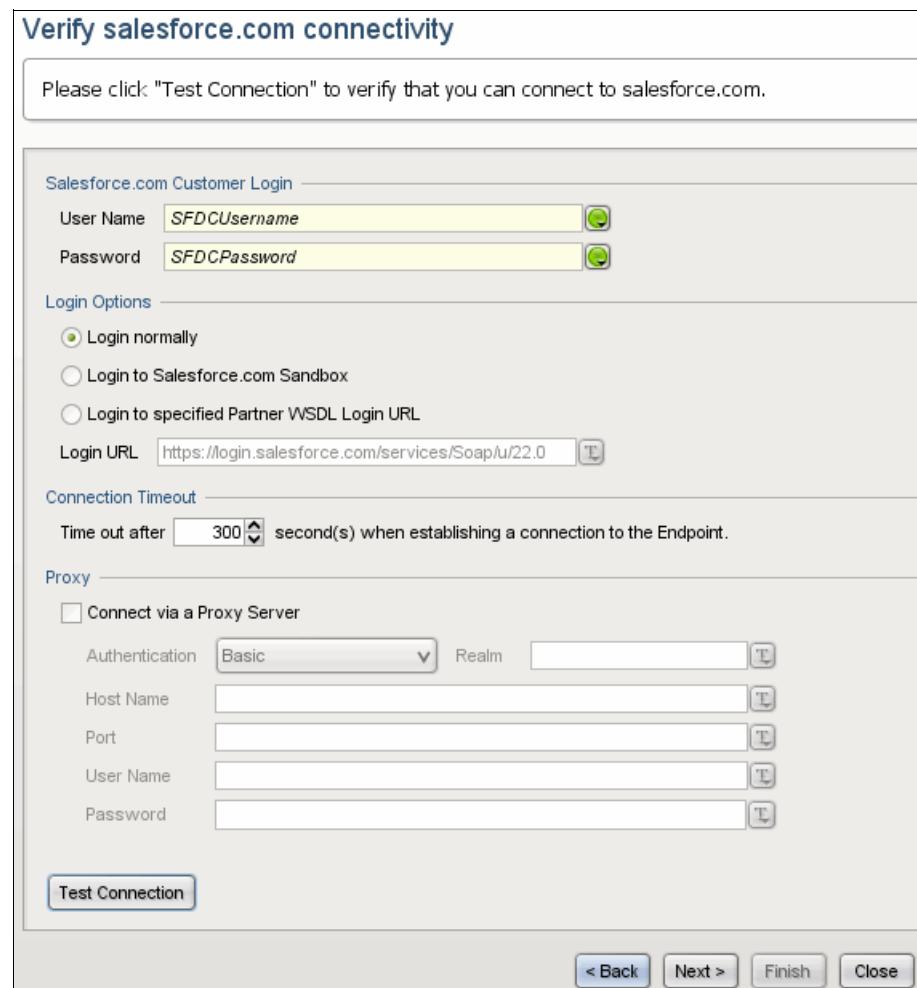


Figure 10-9 TIP wizard step 4: Verify Salesforce.com connectivity

If required, you can now change additional connectivity settings for Salesforce.com, such as providing proxy information. Then you can verify the connection:

1. Click **Test Connection**. Studio attempts to log in to Salesforce.com. If the login is successful, a message opens that displays your organization's name, as shown in Figure 10-10.



Figure 10-10 Successful connection to Salesforce.com panel

2. Click **OK**. If there is an issue with the connection, review the Salesforce.com connection settings.

3. After the connection to Salesforce.com is successful, click **Next** to go to the next step in the wizard.

Verifying SAP connectivity

In step 5, you can adjust the connection settings for SAP on the Verify SAP connectivity panel, shown in Figure 10-11.

Verify SAP connectivity

Please click "Test Connection" to verify that you can connect to SAP.

SAP Application Server

Host Name	SAPHost	
System Number	SAPSys	

Client Information

SAP Client	SAPClient	
User Name	SAPUser	
Password	SAPPASSWORD	
Language	EN	

Connection Pool Options

Maximum Connections	5
Maximum Idle Time	50 minute(s)

Inbound Gateway

Enable inbound gateway - Required only for the Receive IDOC activity

Host	<input type="text"/>	
Server Instances	<input type="text"/>	
Program ID	<input type="text"/>	
Service	<input type="text"/>	
Retry Interval	<input type="text"/>	second(s)

Remote Endpoint Configuration

Endpoint Runs Behind Firewall Secure Connector Name

Test Connection

Buttons: < Back, Next >, Finish, Close

Figure 10-11 TIP wizard step 5: Verify SAP connectivity

To verify the SAP connectivity:

1. Click **Test Connection**. If the connection is successful, a message similar to that shown in Figure 10-12 opens.



Figure 10-12 Successful connection to SAP panel

2. Click **OK**. If there is an issue with the connection, review the SAP connection settings.
3. After the connectivity to SAP is successful, click **Next** to go to the next step in the wizard.

Changing search criteria

In step 6, you can specify how to search for accounts within SAP. Figure 10-13 shows the mapping.

Change search criteria for SAP Customer list

You can change the search criteria for fetching SAP Customers here. Right-click in the "To Activity" panel and use the find menu item to search for "COUNTRNISO" and modify the default value. Similarly, you can find "PiSalesorg" or "PiSearchFlag" and modify those.

Design Test

From Orchestration Select Inputs... To Activity

The 'To Activity' panel shows the following structure:

- request
 - Customer.Search
 - PiAddress
 - FORM_OF_AD
 - FIRST_NAME
 - NAME
 - NAME_3
 - NAME_4
 - DATE_BIRTH
 - STREET
 - POSTL_CODE
 - CITY
 - REGION
 - COUNTRY
 - COUNTRNISO
 - COUNTRAIISO
 - INTERNET
 - FAX_NUMBER
 - TELEPHONE
 - TELEPHONE2
 - LANGU
 - LANGU_ISO
 - CURRENCY
 - CURRENCY_ISO
 - COUNTRYISO
 - ONLY_CHANGE_COMADDRESS
 - PiSalesorg
 - PiSearchFlag

Below the panels is a process flow diagram:

```
graph LR; Start(( )) --> RR[Receive Request  
Start Account Sync]; RR --> IBL[Invoke BAPI  
Get Cust List]; IBL --> FEEC[For Each Customer  
For Each Customer]; FEEC --> IBD[Invoke BAPI  
Get Cust Detail]; IBD --> ICU[If Customers to Upsert];
```

Buttons at the bottom: f↓, < Back, Next >, Finish, Close, Help

Figure 10-13 TIP wizard step 6: Change search criteria

Displayed at the right of this panel, Figure 10-3 on page 361, are three elements that have the letter *D* to the right of the name. This letter indicates that a default value is defined for these values. The elements hold country and sales organization information and a search flag. These elements are used to build the search criteria for the accounts. The default values refer to the configuration properties listed in “Entering the credentials” on page 362. You can add additional mappings or change the existing ones.

When you are finished, click **Next** to go to the next step in the wizard.

Obtaining Salesforce.com custom fields

In step 7, you can select the primary ID that is used to identify whether an update or insert occurs in Salesforce.com. In addition, you can select the fields that you want to send to Salesforce.com.

Tip: Custom fields in Salesforce.com end with _c.

Notice in Figure 10-14 on page 370 that the fields displayed are not identical to the fields of your Salesforce.com account. Specifically, the SalesOrganization field is missing.

Get salesforce.com custom fields

Please click on the refresh button to get your custom fields for the account object from salesforce.com.

Salesforce.com Object Type

Object Type

External ID

Object Field Preferences

Select the fields you would like to map by selecting the check box next to the field name.
Note that custom fields have an *"_c"* suffix for their field name.

<input type="checkbox"/>	Name	Label	Type
<input checked="" type="checkbox"/>	Name	Account Name	Name
<input checked="" type="checkbox"/>	Type	Account Type	Picklist
<input checked="" type="checkbox"/>	ParentId	Parent Account ID	Lookup(Account)
<input checked="" type="checkbox"/>	BillingStreet	Billing Street	Text Area(255)
<input checked="" type="checkbox"/>	BillingCity	Billing City	Text(40)
<input checked="" type="checkbox"/>	BillingState	Billing State/Province	Text(20)
<input checked="" type="checkbox"/>	BillingPostalCode	Billing Zip/Postal Code	Text(20)
<input checked="" type="checkbox"/>	BillingCountry	Billing Country	Text(40)
<input checked="" type="checkbox"/>	ShippingStreet	Shipping Street	Text Area(255)
<input checked="" type="checkbox"/>	ShippingCity	Shipping City	Text(40)
<input checked="" type="checkbox"/>	ShippingState	Shipping State/Province	Text(20)
<input checked="" type="checkbox"/>	ShippingPostalCode	Shipping Zip/Postal Code	Text(20)
<input checked="" type="checkbox"/>	ShippingCountry	Shipping Country	Text(40)
<input checked="" type="checkbox"/>	Phone	Account Phone	Phone
<input checked="" type="checkbox"/>	Fax	Account Fax	Phone
<input checked="" type="checkbox"/>	AccountNumber	Account Number	Text(40)
<input checked="" type="checkbox"/>	Website	Website	URL(255)
<input checked="" type="checkbox"/>	Sic	SIC Code	Text(20)
<input checked="" type="checkbox"/>	Industry	Industry	Picklist
<input checked="" type="checkbox"/>	AnnualRevenue	Annual Revenue	Currency(18, 0)
<input checked="" type="checkbox"/>	NumberOfEmployees	Employees	Number(8, 0)
<input checked="" type="checkbox"/>	Ownership	Ownership	Picklist
<input checked="" type="checkbox"/>	TickerSymbol	Ticker Symbol	Text(20)
<input checked="" type="checkbox"/>	Description	Account Description	Long Text Area(32000)
<input checked="" type="checkbox"/>	Rating	Account Rating	Picklist
<input checked="" type="checkbox"/>	Site	Account Site	Text(80)
<input checked="" type="checkbox"/>	OwnerId	Owner ID	Lookup(User)
<input type="checkbox"/>	CustomerPriority__c	Customer Priority	Picklist
<input type="checkbox"/>	SLA__c	SLA	Picklist
<input type="checkbox"/>	Active__c	Active	Picklist
<input type="checkbox"/>	NumberofLocations__c	Number of Locations	Number(3, 0)
<input type="checkbox"/>	UpsellOpportunity__c	Upsell Opportunity	Picklist
<input type="checkbox"/>	SLASerialNumber__c	SLA Serial Number	Text(10)
<input type="checkbox"/>	SLAEExpirationDate__c	SLA Expiration Date	Date
<input checked="" type="checkbox"/>	ExtAccountId_c	Ext Account ID	Text(25) (External ID)
<input checked="" type="checkbox"/>	ExtSource__c	External Source	Text(15)

Figure 10-14 TIP wizard step 7: Get Salesforce.com custom fields

To retrieve the list of fields that are available for your Salesforce.com environment:

1. Click the blue circle. Studio connects to Salesforce.com to retrieve a current list of standard and customer fields, also called *metadata*, for your Account object. While Studio connects, the message shown in Figure 10-15 displays.



Figure 10-15 TIP wizard retrieving metadata from Salesforce.com

2. Verify that the list is updated with the ExtAccountID__c and SalesOrganization__c fields and that they are selected, as shown in Figure 10-16.

<input checked="" type="checkbox"/> ExtAccountID__c	ExtAccountID	Text(15) (External ID) (UNIQUE)
<input checked="" type="checkbox"/> SalesOrganization__c	SalesOrganization	Text(10)

Figure 10-16 TIP wizard new fields for account object

3. Select **ExtAccountID__c** in the list of custom fields, and verify that ExtAccountID__c is selected in the External ID field (shown at the top of the window), as shown in Figure 10-17.

Figure 10-17 TIP wizard Salesforce.com external ID

4. Clear fields that you do not need, for example the fields for shipment details, such as ShippingCity.
5. Click **Next** to go to the next step in the wizard.

Changing account mapping

In Step 8 of the wizard you can define the mapping from the SAP elements to the Salesforce.com elements. Some mappings are already done, but some are missing, as shown in Figure 10-18.

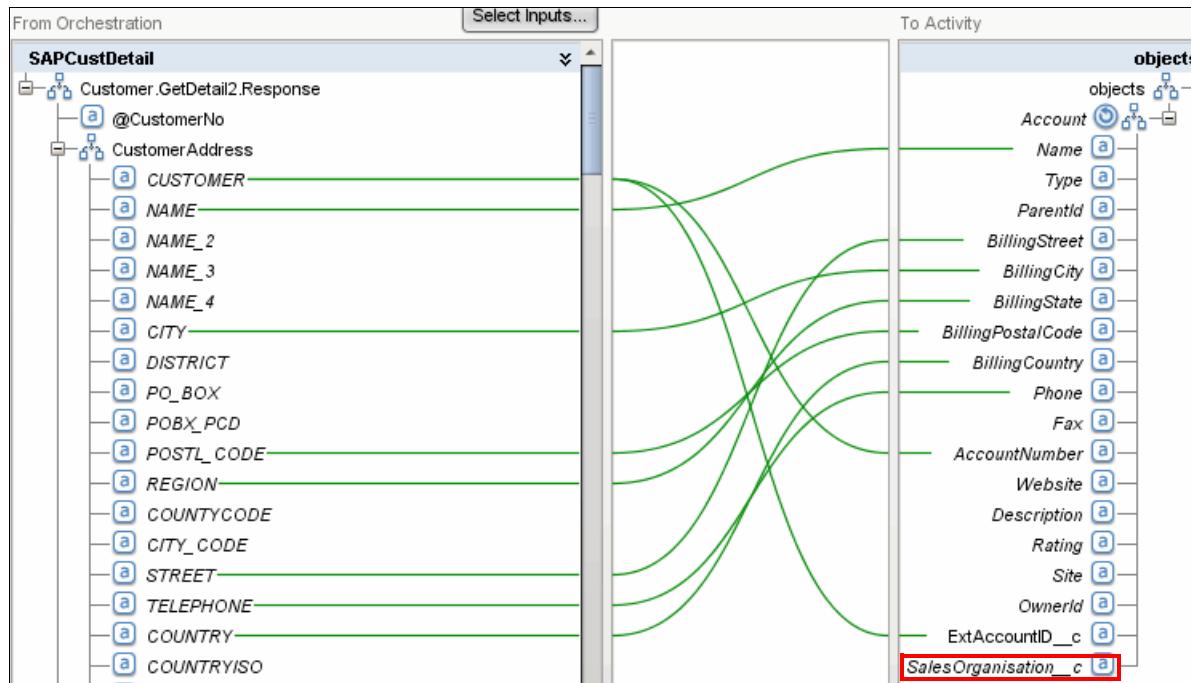


Figure 10-18 TIP wizard mapping SAP to Salesforce.com

As indicated in the figure, the field for the sales organization on the right side is not mapped. To store the sales organization into Salesforce.com, you must put the information into the field:

1. Right-click the **SalesOrganisation_c** field, and select **Define Default Value**. Click the symbol on the right side of the entry field as indicated in Figure 10-19, and select the configuration property **SAPSalesOrg**. Click **OK**.

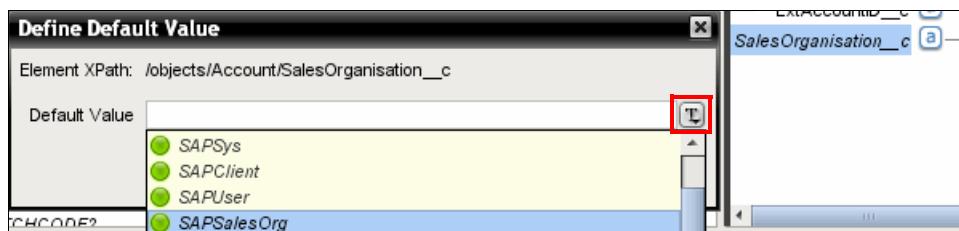


Figure 10-19 Use case SAP to SFDC TIP wizard define mapping for SAP sales organization

2. Follow the link from the SAP field CUSTOMER on the left of the Salesforce.com field. ExtAccountID is on the right. This is a direct mapping, meaning that the content is copied from the source to the target.

However, for this scenario, you do not want to store the SAP customer number as is into Salesforce.com. Instead, you need the SAP customer number to use the following format:

SAP <SAPcustomernumber>

Therefore, you must modify the mapping, and enhance it with a concatenate function.

3. Click the function symbol (**f**) to the left of the Back button, located at the bottom of the panel, as indicated in Figure 10-20.



Figure 10-20 TIP wizard function button

4. Grab the **Concatenate** function, and drop it on the link between CUSTOMER and ExtAccountID_c, as shown in Figure 10-21. Double-click the created Concatenate (&) symbol to open the concatenate function.

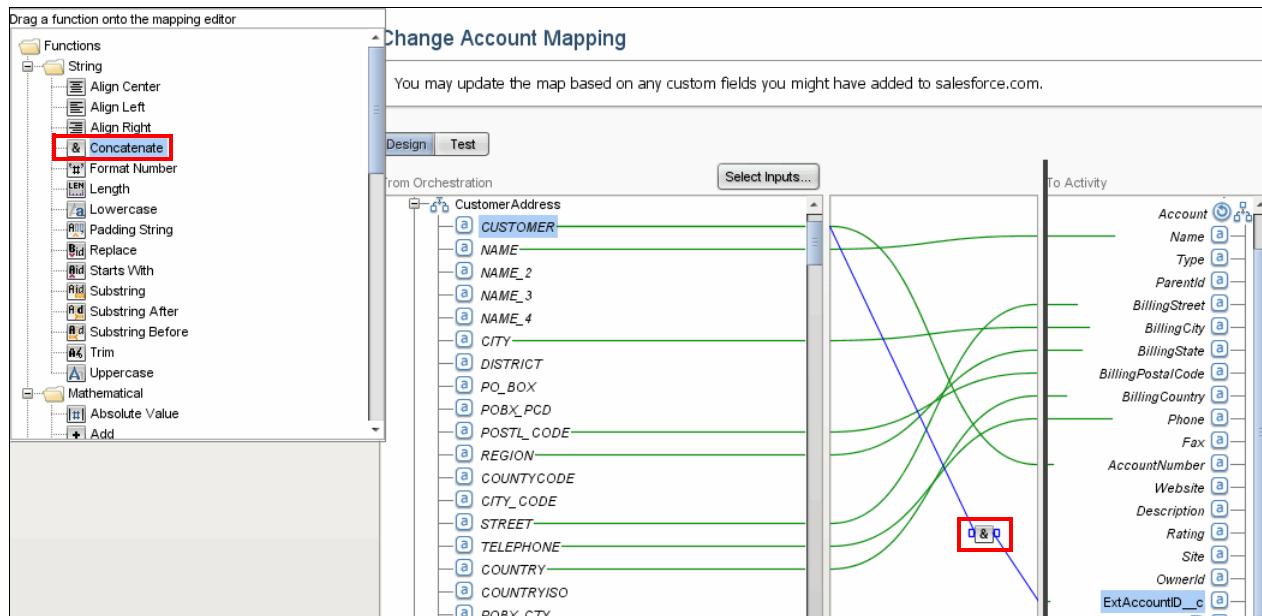


Figure 10-21 TIP wizard enhance mapping for ExtAccountID

5. Enter SAP_ for the second input string, and click **Up**. The function now looks as shown in Figure 10-22.

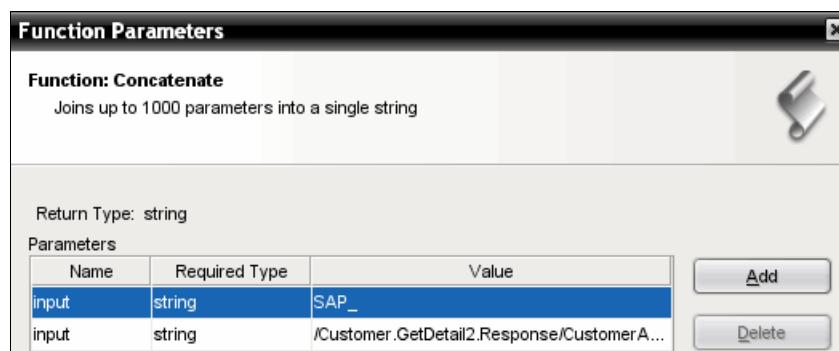


Figure 10-22 TIP wizard concatenate function parameters

- Click **OK**, and then right-click the Concatenate function symbol (**&**), and select **Apply Function Graph**. The mapping now looks as shown in Figure 10-23.

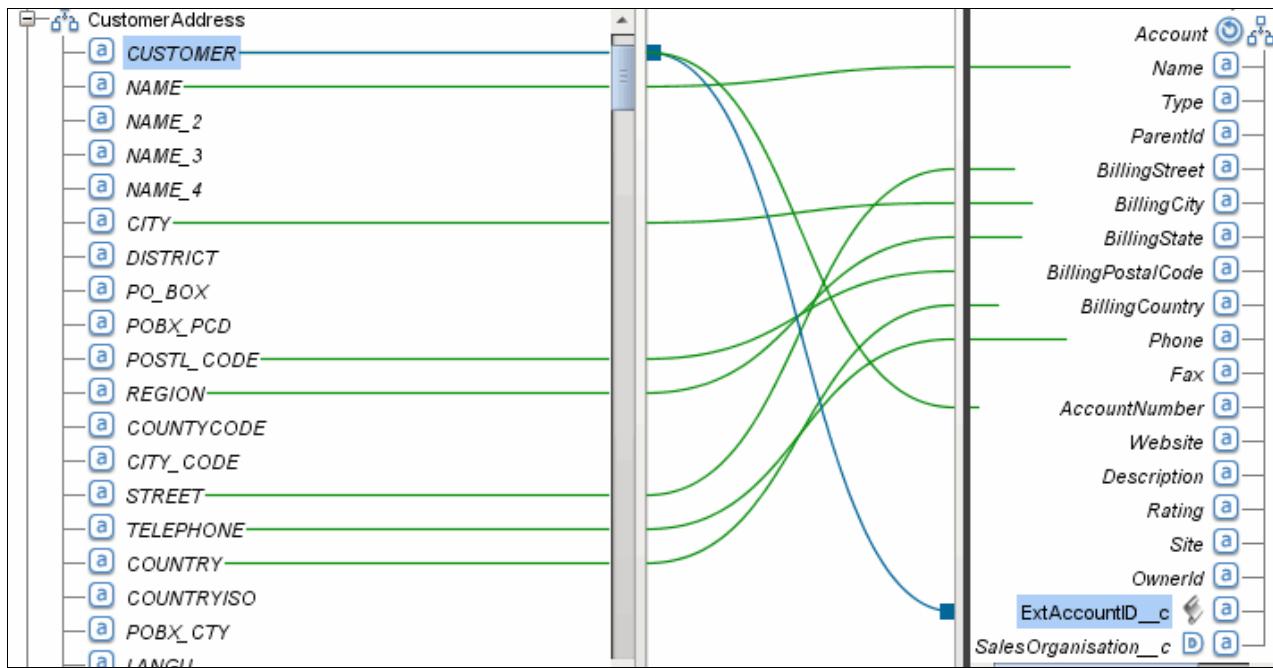


Figure 10-23 TIP wizard final SAP to Salesforce.com mapping

- The mapping is now finished from SAP to Salesforce.com. Click **Next** to go to the next step in the wizard.

Verifying the integration

In step 9, you can now start and verify the orchestration, as shown in Figure 10-24.

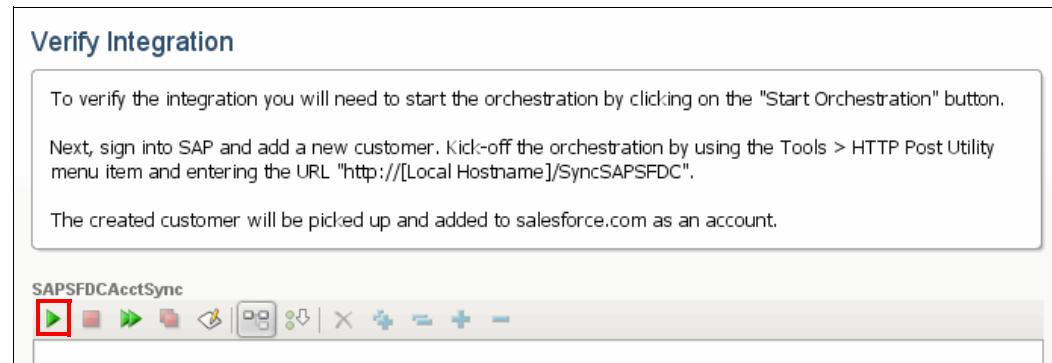


Figure 10-24 TIP wizard verify integration

Before you verify the orchestration: Before you complete this step, verify with your SAP administrator that the SAP query will not return hundreds of records, which can cause problems in Salesforce.com. If you are worried about testing the orchestration at this time, click **Next**, and skip to step 10.5.3, “Customizing the orchestration” on page 376.

To verify and start the orchestration:

1. Click **Start Orchestration**. The orchestration is triggered by an HTTP request. So, start the HTTP post utility using **Tools** → **HTTP Post Utility**. Use the following URL to send the request, as shown in Figure 10-25:

`http://<hostname>/SyncSAPSFDC`

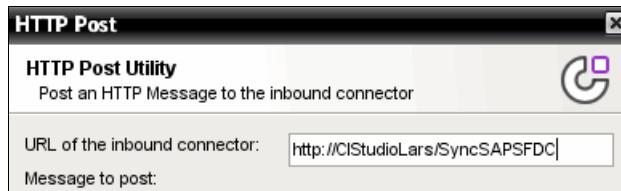


Figure 10-25 Orchestration test using HTTP Post Utility

2. Studio shows that the orchestration is triggered and completes quickly, unless you have a lot of customer accounts that are processed or a connection problem. Expand the tree, Figure 10-26, and verify that each account was processed successfully.

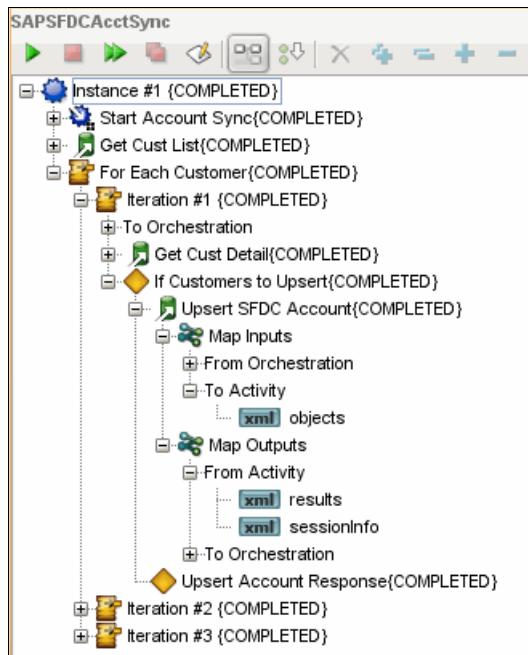


Figure 10-26 Orchestration test to verify the integration results

3. Log into Salesforce.com, and verify that the accounts were created.
4. Return to Cast Iron Studio, and click **Next** to continue to the next step in the wizard.

Publishing and finishing

At this point, you can publish the orchestration to an Integration Appliance. However, for this scenario, skip this step at the moment. Click **Finish**.

In the final panel of the TIP configuration wizard, you can rate and comment on this TIP. To support the community, rate the TIP after you finish testing. Click **Close** to close the wizard.

Tip: Do not forget to save your work from time to time using **File** → **Save**.

10.5.3 Customizing the orchestration

This section provides a closer look at the orchestration. After closing the configuration wizard, the orchestration displays, as shown in Figure 10-27.

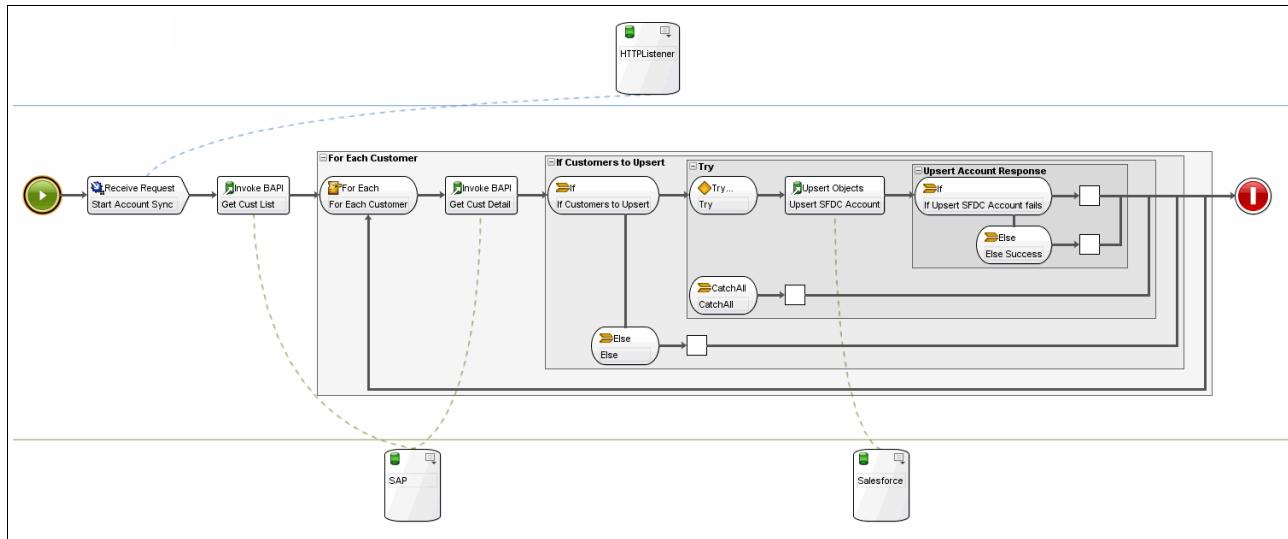


Figure 10-27 Use case SAP to SFDC Orchestration overview

To see the configuration properties for this configuration, go to **Project → Configuration Properties**. The properties display, as shown in Figure 10-28. Notice that there are credentials for Salesforce.com and for SAP. In addition, there are search parameters for SAP.

Figure 10-28 Project configuration properties

The next sections walk through the process for an incoming request.

Retrieving and validating data

The orchestration starts with an HTTP Receive Request activity, as shown in Figure 10-29 on page 377.

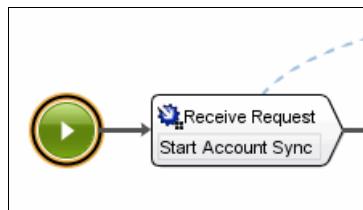


Figure 10-29 Use case SAP to SFDC receive request

The activity listens on port 80 for the /SyncSAPSFDC URL and does not return any data.

The second activity is the SAP Get Cust List activity. It takes the country, the sales organization, and a search flag as input (see Figure 10-30) and returns in the SAPSearchResult variable a list of SAP customer numbers for these criteria. The BAPI function that is used is Customer Search.

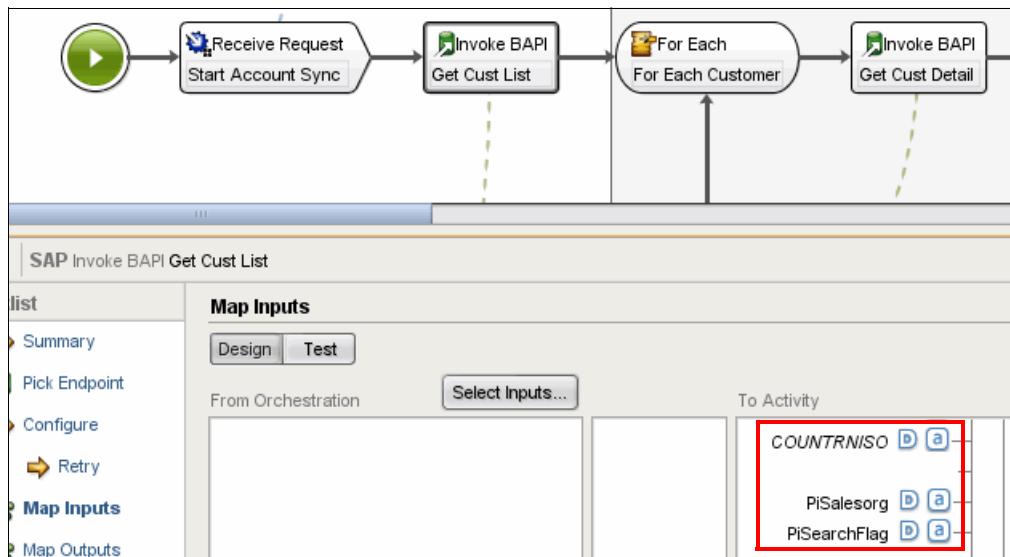


Figure 10-30 Get Cust List

The next activity is the For Each loop, which loops through all the customer numbers in the list. Notice in the details that this activity takes the SAPSearchResult as input and creates a loop variable named SAPCustomer that contains just one customer number. Click into the field, and change the variable to SAP_OneCustomerNumber. The activity now looks as shown in Figure 10-31 on page 378.

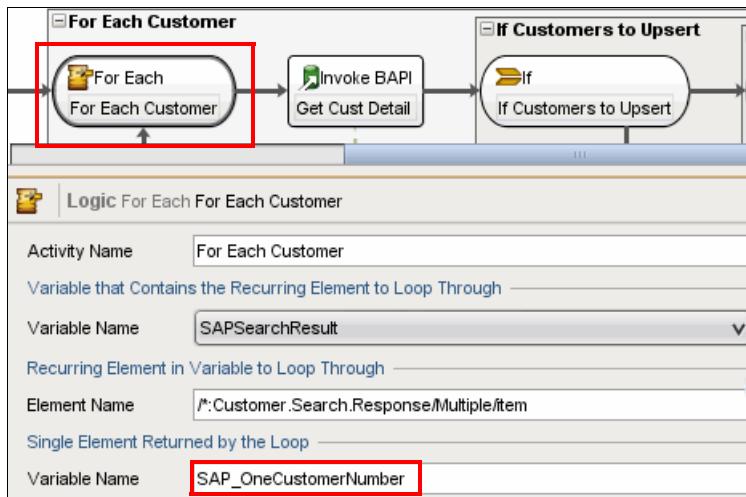


Figure 10-31 For Each loop

The loop uses the SAP BAPI with the Customer object and the GetDetail2 method to get the customer details for the SAP customer number, as shown in Figure 10-32.

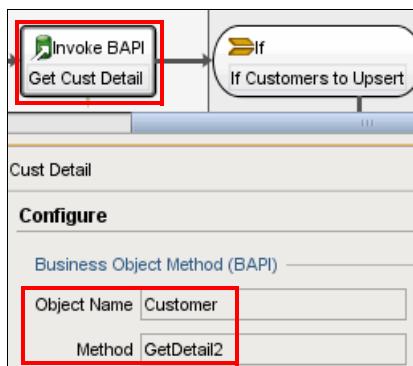


Figure 10-32 SAP customer GetDetail2

This search returns an element named SAPCustDetail with the SAP customer information, as shown in Figure 10-33.

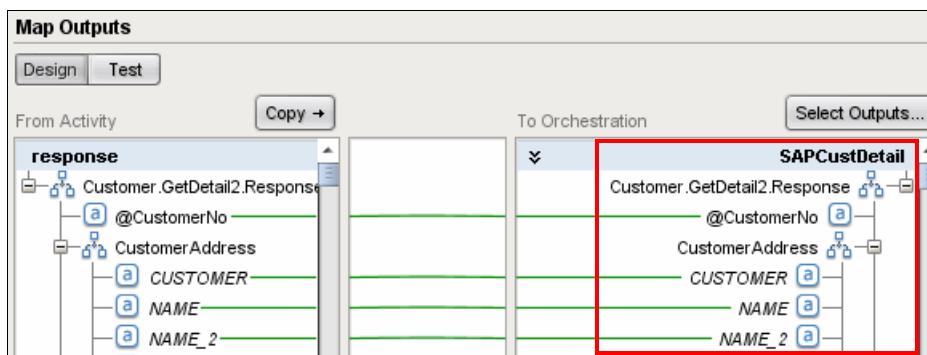


Figure 10-33 SAP customer GetDetail2 result

The returned SAP customer record is validated in the If-Then activity, which in this case performs a simple validation when the number of CUSTOMER records returned is greater than 0, as shown in Figure 10-34 on page 379.

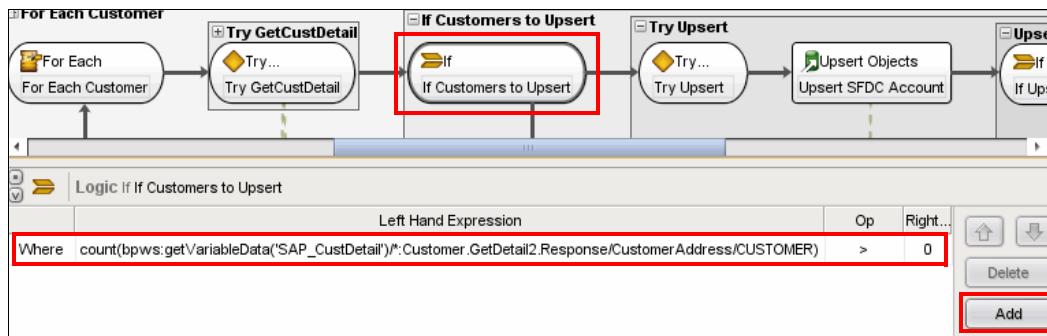


Figure 10-34 Validate SAP customer using If-Then activity

You can adjust the validation, for example to avoid a customer without a name, as follows:

1. Click the If Customers to Upsert activity, and click **Add** in the properties pane to add another expression.
2. For the left expression, select the **SAPCustDetail** variable and the **CustomerAddress → NAME** element. Studio creates the related XPATH expression automatically.
3. For the operation, use the **!=** option for *not equal*.
4. For the right expression, insert quotation marks (""), which indicates an empty string.

The expression now looks as shown in Figure 10-35.

	Left Hand Expression	Op	Right Hand Expression
Where	count(bpws:getVariableData('SAPCustDetail'))>Customer.GetDetail2.Response/CustomerAddress/CUSTOMER	>	0
and	bpws:getVariableData('SAPCustDetail'))>Customer.GetDetail2.Response/CustomerAddress/NAME	!=	""

Figure 10-35 Enhanced validation of SAP customer

Adding a log message

Now, the else-branch of the if-activity is empty. This configuration is not recommended. At minimum, you need a log message. You can add a log message as follows:

1. Add a Log Message activity to the else-branch, and name it Log Invalid SAP Customer.
2. In the Checklist click **Map Inputs**:
 - a. Right-click **level**, select **Define Default Value**, and then select **WARNING**. Click **OK**.
 - b. Click **Select Inputs**, and select **SAPCustDetail**. Click **OK**.
 - c. Click the Functions tab, and drop the Concatenate function on the middle part of the canvas.
 - d. Link the fields CUSTOMER and NAME to the concatenate function, and the concatenate function to the field message, as shown in Figure 10-36 on page 380.

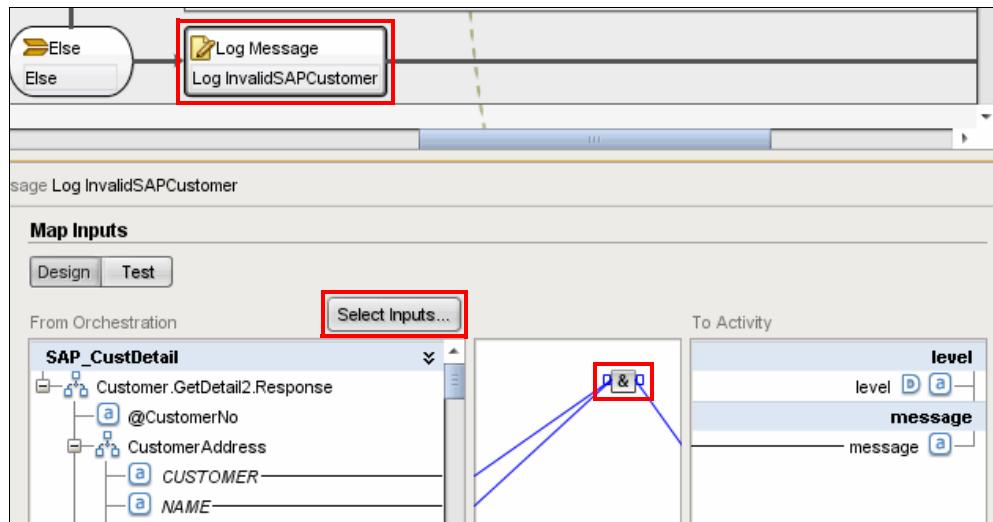


Figure 10-36 Create log message

- e. Double-click the **Concatenate** function, and build a meaningful message from the input fields by adding strings with text blocks and moving them to the right place. The definition shown in Figure 10-37 creates the following message:

Invalid customer with customer number xxx and customer name yyy

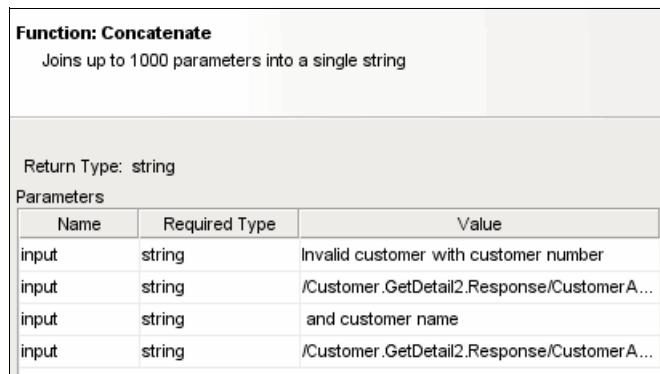


Figure 10-37 The else branch log message concatenate

- f. Click **OK** to close the concatenate function.
g. Right-click the **Concatenate** function in the map, and select **Apply Function Graph**.

Inserting or updating Salesforce.com

You are in the last part of the orchestration. You have retrieved a list of SAP customer numbers, selected one-by-one within the For Each loop, retrieved the SAP customer details, and verified those details. Now, you must insert or update Salesforce.com. Figure 10-38 on page 381 shows an overview of this process.

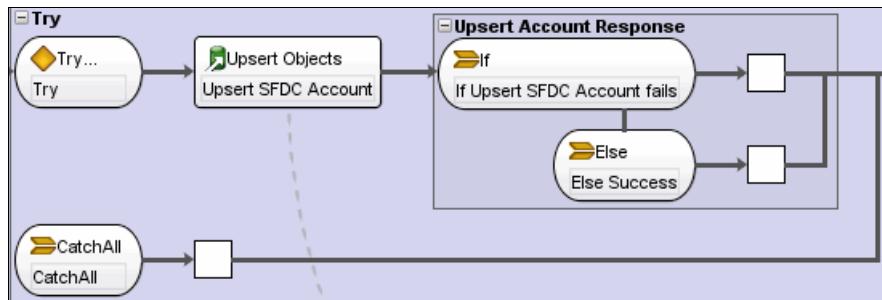


Figure 10-38 Upstart Salesforce.com

Try-Catch block

First, a Try-Catch block handles failures that might occur during an update. Without this block, for example, if one account cannot be updated due to an error, the entire orchestration stops and no other account is updated within the synchronization.

As shown in Figure 10-38, the Catch All branch is empty, which is not recommended. You will add a Log Message activity and, later, the Common Error Handler. To make the orchestration more readable and easier to understand:

1. Rename the Try activity to *Try Upstart*. Click the name or right-click, and select **Rename** to change the name.
2. Rename the CatchAll Block to *CatchUpstart*.
3. Click the **CatchUpstart** block, and rename the variables to **SFDC_UpsertFaultName**, **SFDC_UpsertFaultData**, and **SFDC_UpsertFaultInfo** so that you can identify them easily later.

The CatchUpstart now looks as shown in Figure 10-39.

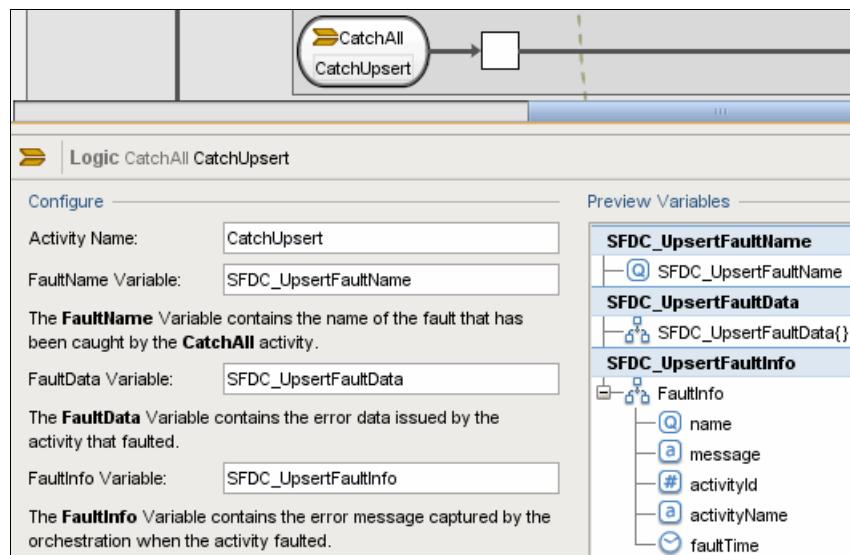


Figure 10-39 Catch Upstart

4. Drag a Log Message activity to the catch branch, and name it *Log Error during Upstart*.
5. In the Map Inputs section, define the error message as follows:
 - a. Set the level to **ERROR**.

- b. As input, select the **SFDC_UpsertFaultInfo** variable, which contains details about the fault.
- c. Add the concatenate function. Map the input element's name, message, activityName, and faultTime to the concatenate function. Map the output element message to the function. Create a message for the concatenate, such as:
Fault xxx, error yyy in activty zzz during Upsert at TTT
For details, refer to the procedure beginning with step a on page 379.

Upsert Objects activity

The next activity is the Upsert Objects activity. A function called *upsert* is provided by Salesforce.com. This function does an insert if the object is a new account. Otherwise, it does an update. The decision of whether or not an account is new is made based on a primary key, the external ID.

The process for the upsert is:

1. As shown in Figure 10-40, the external ID is defined as **ExtAccountId_c**. This ID was set in the template configuration wizard in “Obtaining Salesforce.com custom fields” on page 369.

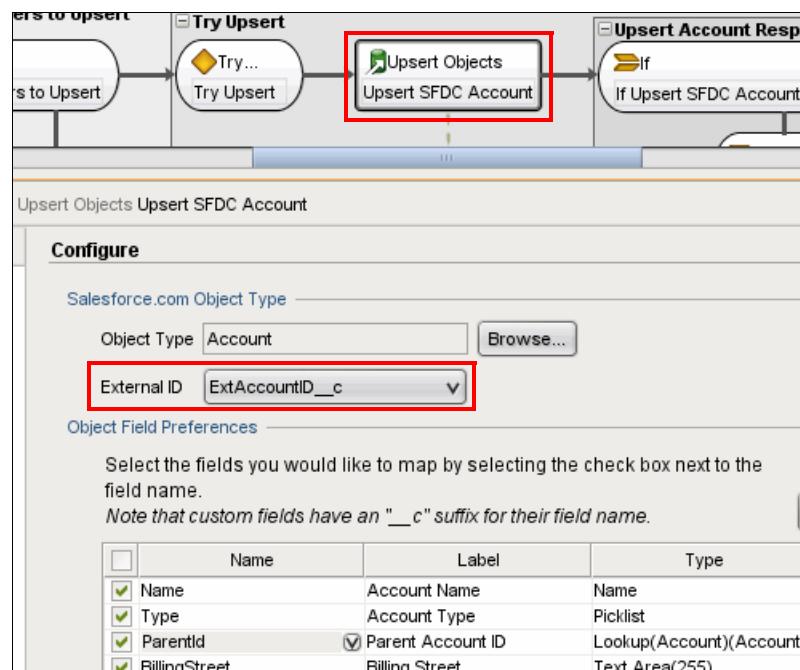


Figure 10-40 Upinsert configuration

2. Click **Map Inputs** for the input mapping. In the configuration wizard (step 2 on page 372), you configured the **ExtAccountId_c** field to be filled with a the **SAP_xxxx** string, where **xxxx** is the SAP customer number.
3. Click **Map Outputs** to view the results that are stored in the **SFDCUpserResponse** variable.
4. The If Upsert SFDC Account Fails If-Then activity, validates that the upsert was successful by looking at the success element of the **SFDCUpserResponse** variable (refer to Figure 10-41 on page 383).

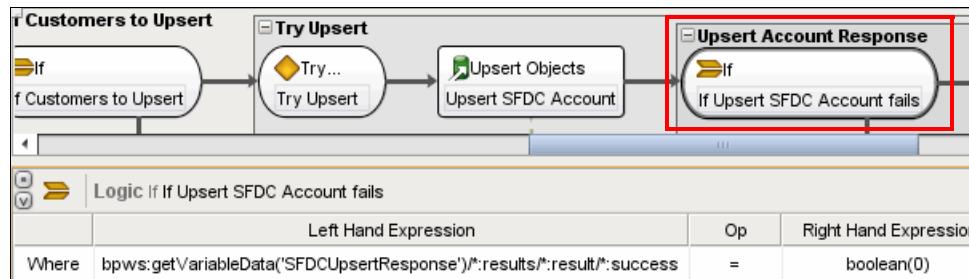


Figure 10-41 Validate that the upsert successful

The If-Then activity

The If-Then activity has two branches, but both branches are empty. You can add a Log Message activity to both branches:

1. For a failed upsert, set the message level to WARNING or ERROR. Change the content of the message to a message that contains both the customer number from the SAPCustomer variable and details from the SFDCUpsertResponse statusCode. For example:

Upsert of customer xxx failed, error yyy

Tip: The statusCode element in SFDCUpsertResponse is part of a recurring structure. Thus, you cannot map it directly to the Concatenate function. If you do, you receive the following error:

Mapping is not allowed. You cannot map from a recurring node to a non-recurring node

Instead, select the first element as the one to map by right-clicking **statusCode**, and then clicking **Select One Occurrence**. Select the result[1]/errors[1]/statusCode element shown in Figure 10-42. Click **OK**, and create a link from statusCode to the Concatenate function.

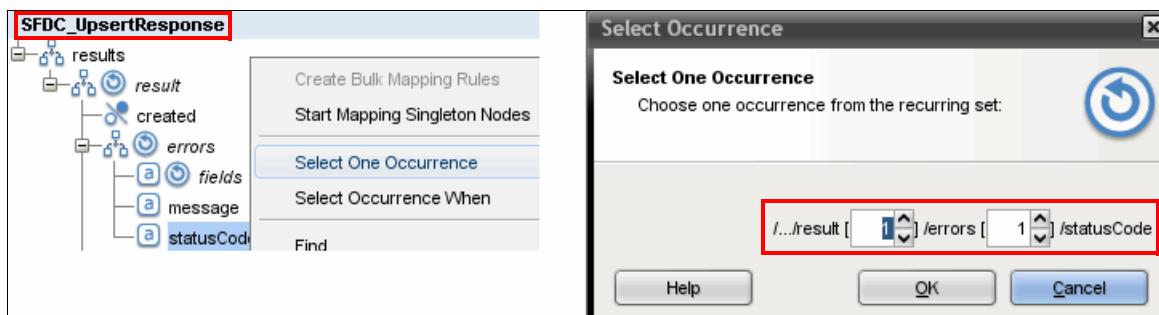


Figure 10-42 Selecting an occurrence of statusCode to map to the Concatenate function

2. For the successful upsert, set the message level to INFO and the message content to text similar to the following text:

Upsert of customer xxx successful

If you do not remember how to build the log message, refer to the example with the else branch starting with step 1 on page 379. The final if block looks similar to that shown in Figure 10-43 on page 384.

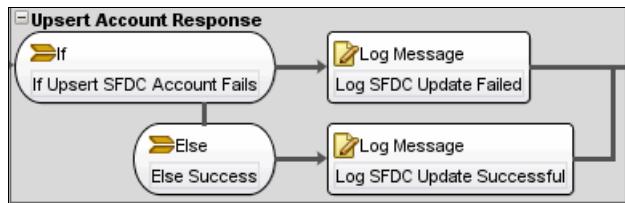


Figure 10-43 Upstart if block with log

Global catch

The final step is to add a global catch to the orchestration, which ensures that if a problem occurs, the error is captured and can be handled appropriately. If you do not create a global catch, any error that is not caught in a try-catch block causes the orchestration to fail with a generic error message that contains the information of the fault variables. A global catch provides flexibility in handling errors.

To define the global catch:

1. Right-click the starting point of the orchestration, and select **Add CatchAllBranch**, as shown in Figure 10-44, to create a new branch.



Figure 10-44 Global CatchAll

2. Click the CatchAll block, and rename the variables to GlobalCatchFaultName, GlobalCatchFaultData, and GlobalCatchFaultInfo. The result looks similar to that shown in Figure 10-45.

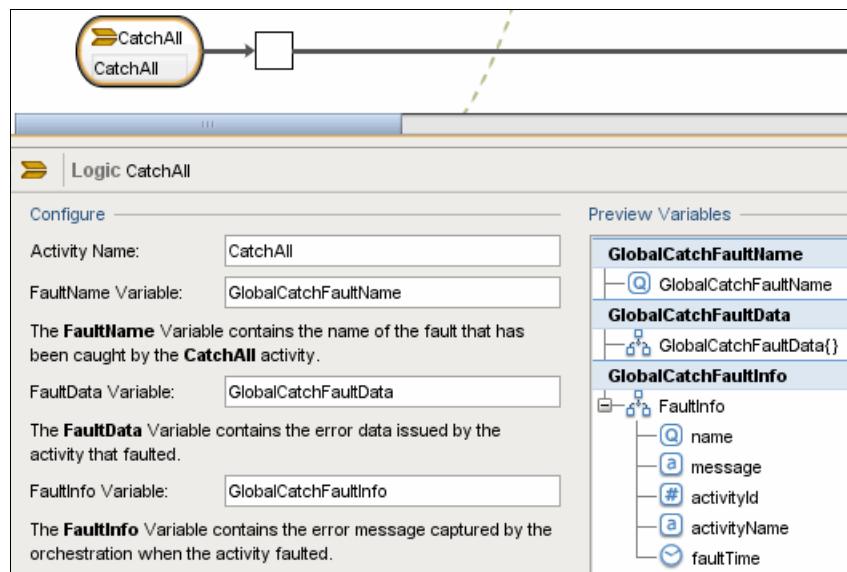


Figure 10-45 Global catch configuration

3. Add a Log Message activity to the catch block. For ease of use, you can simply reuse the Log Message activity we created in the try-catch block for the upsert (see step on page 381):
 - a. Go to the Log Message activity in the Upsert try-catch block. Right-click the activity, and select **Copy**.
 - b. Go to the empty activity in the catch branch of the global catch. Right-click the empty activity, and select **Paste**.
 - c. Click **Map Inputs**. Right-click one of the mappings, and select **Edit Function Graph**.
 - i. Right-click the SFDC_UpsertFaultInfo variable, and select **Remove Variable**.
 - ii. Click **Select Inputs**, and select the GlobalCatchFaultInfo variable.
 - iii. Because the elements in GlobalCatchFaultInfo are the same as in the SFDC_UpsertFaultInfo variable, the mapping is now done automatically. If the mapping does not come back, create the mapping again using the Concatenate function.
 - iv. Verify the mapping, and click **Apply Function Graph**.
 - v. Rename the activity to Log Error during Sync, and adjust the message within the concatenate slightly as follows:
- Fault xxx, error yyy in activity zzz during Sync at TTT

10.5.4 Reducing the SAP result set

If you are not sure how many records the SAP Get Customer List function will return or if the activity returns too many records, you can use a filter. A filter can also be helpful during normal and failure testing. The sections that follow explain how to create a filter that will reduce the SAP result set.

Creating a filter

To create a filter:

1. Put a terminate activity just behind the “Get Cust List” Invoke BAPI SAP activity to avoid the results being sent to Salesforce.com, as shown in Figure 10-46.

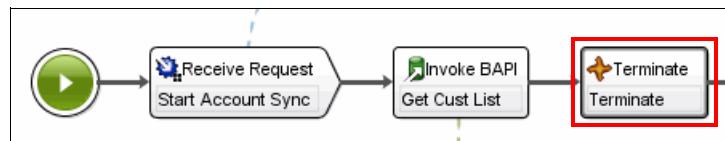


Figure 10-46 Get Cust List with terminate

2. Click the **Verify** tab, and start the orchestration within Studio.
3. Use the HTTP Post Utility to send a request to <http://<yourhost>/SyncSAPSFDC>.

4. Expand the instance tree to **Get Cust List** → **Map Outputs** → **To Orchestration**, and click the SAPSearchResult element to display the content of the element. The content is a list of customer numbers, shown in Figure 10-47. Depending on your SAP system, this list can be long.

```

<?xml version="1.0" encoding="UTF-8"?>
<sp_0:Customer.Search.Response xmlns:sp_0="urn:</pre>
<Return>
    <LOG_MSG_NO>000000</LOG_MSG_NO>
</Return>
<Multiple>
    <item>
        <CUSTOMER>0000000224</CUSTOMER>
    </item>
    <item>
        <CUSTOMER>0000001170</CUSTOMER>
    </item>
    <item>
        <CUSTOMER>0000001000</CUSTOMER>
    </item>
    <item>
        <CUSTOMER>0000001001</CUSTOMER>
    </item>
    <item>
        <CUSTOMER>0000001002</CUSTOMER>
    </item>

```

Figure 10-47 Customer list returned by SAP

Reducing the list by creating a filter

For testing purposes, this list is too long and needs to be reduced. You can reduce the list using several methods.

You can change the SAP query. (We do not demonstrate this method.)

You can also filter the list of returned customers in the output mapping of the SAP activity:

1. Go to the Get Cust List output mapping, and right-click the **item** element in SAPSearchResult.
2. Select the Filter Recurring Nodes option, and enter an XPATH-Expression filter, such as `starts-with(CUSTOMER, "0000")`. Click **OK**. The item element then gets the symbol **f** for function, as shown in Figure 10-48.



Figure 10-48 Create SAP filter within map

For more flexibility in filtering, you can also use a filter activity:

1. Create a variable named SAP_ListFilter of type string with the default value 0000.
2. Drag a Filter and Profile activity between the SAP activity and the Terminate activity, as shown in Figure 10-49.



Figure 10-49 Use case SAP to SFDC, create SAP filter

3. Rename the activity to Extract SAP Customers.
4. In the checklist, select **Configure**. Select the SAPSearchResult variable and the item element, as shown in Figure 10-50. The XPATH is created automatically.

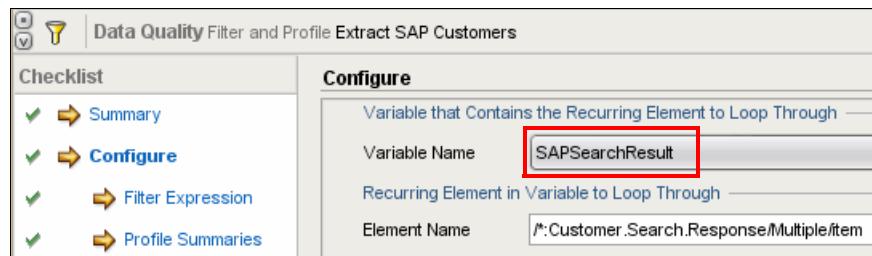


Figure 10-50 SAP filter configuration

5. Click **Filter Expression**. You can define the filter expression using two methods:
 - Define the filter expression, as shown in Figure 10-51.

Filter Expression			
	Left Hand Expression	Op	Right Ha...
Where	<code>starts-with(CUSTOMER, bpws:getVariableRaw('SAP_ListFilter'))</code>	=	<code>true()</code>
or	<code>bpws:getVariableRaw('SAP_ListFilter')</code>	=	<code>"0000"</code>

Figure 10-51 SAP filter expression

- Use the Advanced button, and insert the following expression:

```
starts-with(CUSTOMER, bpws:getVariableRaw('SAP_ListFilter')) = true() or
bpws:getVariableRaw('SAP_ListFilter') = "*"
```

The logic for both methods is the same.

If SAP_ListFilter is set to “*”, the filter returns all customer numbers. If the filter is set to anything else, the filter returns all customer numbers that start with that filter expression. For example, if filter is set to 1001, the filter expression returns all customer numbers starting with 1001, such as 1001 and 1001231.

You can also use another expression based on other XPATH functions, such as substring, contains, or other functions.

6. For the Profile Summaries definition, do not enter anything.
7. In Map Outputs, map the goodXML CUSTOMER field, as shown in Figure 10-52.

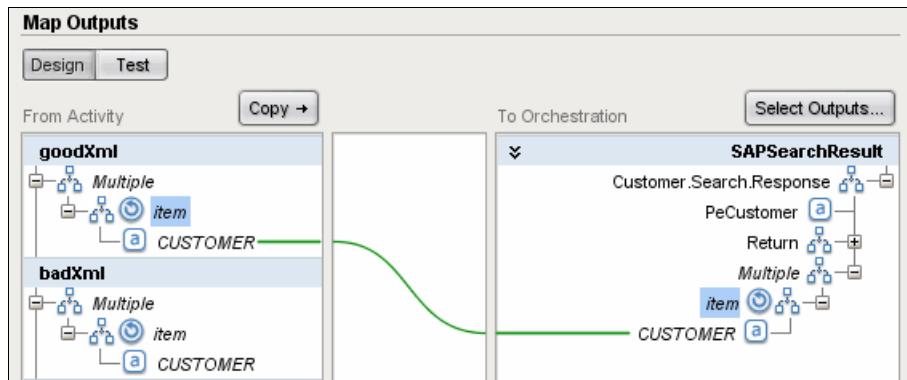


Figure 10-52 SAP filter output mapping

Testing the filter

Test to see whether the filter works:

1. Start the orchestration.
2. Use the HTTP Post Utility with the URL:
<http://CIStudioLars/SyncSAPSFDC>
3. Investigate the results returned. The output of the filter contains only the customer numbers that match the value of SAP_ListFilter.

Tip: If you get an error message, such as failed to filter data, SAP_ListFilter is null, ensure that you defined the SAP_Filter correctly in “Reducing the list by creating a filter” on page 386.

4. Add a filter to the HTTP request:
 - Select the Receive Request activity, and select **Map Outputs** in the checklist.
 - Click **Select Outputs**, and add the SAP_ListFilter variable.
 - Map the body from the activity to SAP_ListFilter, as shown in Figure 10-53 on page 389.

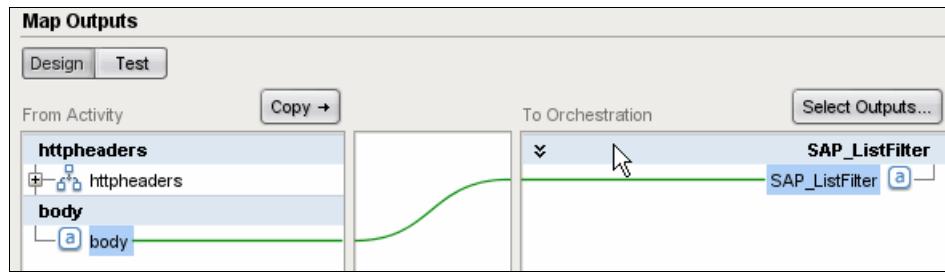


Figure 10-53 Map HTTP body to SAP_ListFilter

- d. Right-click the SAP_ListFilter variable, and select the Define Default Value option. Set a default value, for example, an asterisk (*).
- e. Click the small green dot in the Default Value field to create a configuration property from this filter named SAP_ListFilter_Default. Click **Create**. Click **OK**.
5. Test the orchestration using the HTTP Post Utility:
 - If you do not enter anything for the HTTP body, the value from the project configuration property SAP_ListFilter_Default is used as the filter criteria.
 - If you enter a search string in the body of the HTTP request, that value is used as filter criteria.

You now have a flexible filter that can be customized in the run time, either in the Web Management Console (WMC) using a configuration property or using an HTTP request parameter.

Before you continue: Remove the Terminate activity because this scenario does not need it.

10.5.5 Naming conventions and other definitions

A project and its assets must have the same style regardless of who wrote the asset. This can be achieved using conventions, such as naming conventions. Conventions make the project easier to use by others.

Naming conventions

Naming conventions can provide an easier understanding of the assets. Adjust the project now to fit to your naming conventions. Consider the following naming conventions:

- ▶ Variables

Before renaming any variable, first remove the variables that are not in use. (On the Variables tab, right-click a variable, and then select the Remove Unused Variables option.)

This scenario uses the naming convention that an endpoint-specific variable has the format *endpointname_valuedetails*. Each word starts with an uppercase letter, for example SAP_OneCustomer and SFDC_UpsertResponse.

To modify the variables to fit to your naming conventions, go to the Variables tab, and select the variable of choice. The middle right section shows the dependencies. In the lower right Properties section, you can then rename the variable.

- ▶ Configuration Properties

This scenario uses the same rules for configuration properties as for variables. To change them, go to **Project → Configuration Properties**, and select the property that you want to change.

- ▶ Activities

This scenario uses the naming convention where the words in the activity name start with an uppercase letter, for example, Upsert SFDC Account.

You can change the name of an activity by double-clicking the name in the orchestration graph or by selecting the activity and changing the name in the Checklist Summary.

- ▶ Orchestrations

This scenario uses the naming convention of an orchestration, the naming convention *endpoint1_endpoint2_purpose*, for example SAP_To_SFDC_SyncAccounts.

To rename an orchestration, right-click it in the Project tab, and select **Rename**.

- ▶ Projects

This scenario uses the naming convention where a project indicates the endpoints and the usage. The project here currently has the name inherited from the template, which is T0037_SAPCustomerToSFDCAccounts. To rename the project, save it under another name, for example, SAP_To_SFDC_SyncAccounts.

- ▶ Endpoints

If the endpoints in the project can be used for both inbound and outbound flow, the default name is sufficient. However, if an endpoint provides only one direction, for example with HTTP endpoints, add that information to the name, for example SAP and HTTP_In.

Also, rename the endpoint from Salesforce to Salesforce.com.

To rename an endpoint, right-click the endpoint in the Project tab, and select **Rename**.

- ▶ URLs

Use a naming convention for URLs that is similar to the convention for orchestrations. The URL used in the Receive Request activity is currently named /SyncSAPSFDC.

Unfortunately, this name does not specify in any way what is synchronized. So, a better approach is to use the URL /SAP_To_SFDC_SyncAccounts. Change the URL by clicking **Configure** in the Checklist for the activity, and then updating the URL.

Configuration properties

Configuration properties help to make the project more flexible without having to go back to Studio for changes. Consider the following examples of settings that can be made more flexible using a configuration property:

- ▶ You can change the HTTP_In endpoint configuration properties. Double-click the endpoint in the orchestration graph to open the properties.

As shown in Figure 10-54, the port is defined statically to port 80.

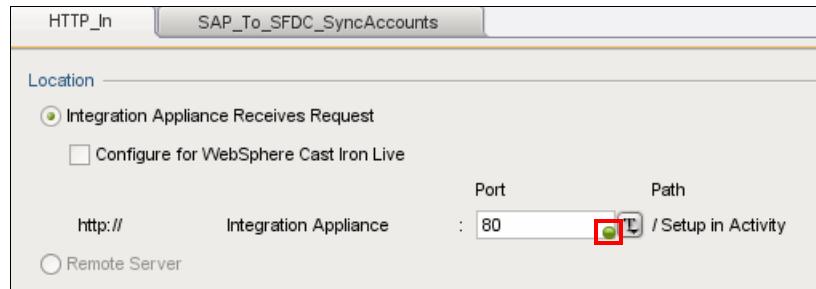


Figure 10-54 HTTP endpoint configuration

This setting might be sufficient in your current environment, but your operations team might want the endpoint listener on another port. So, it makes sense to create a configuration property named HTTP_In_Port that defines a specific port for the endpoint. To do this, click the green circle next to the 80 (indicated in Figure 10-54), and insert the name into the field. Click **Create**, and close the properties.

- ▶ You can change the HTTP Receive Request activity. As shown in Figure 10-55, the URL is specified statically in the Configure pane.

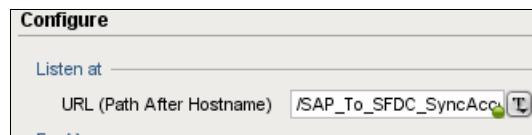


Figure 10-55 Receive request URL configuration

Replace the URL with a configuration property so that you can change the URL easily later on.

Project settings

You can make an activity more flexible by changing project settings. To view the current project settings, click **Project** → **Project Settings**. Figure 10-56 shows an example.

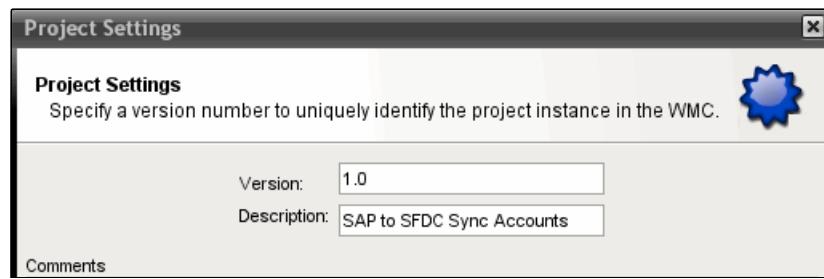


Figure 10-56 Project Settings configuration

Comments

Each activity provides the option to add comments that describe the activity in more detail, for example by providing the purpose of a query or an if-statement. Comments make the project more readable and are part of the documentation that can be generated automatically by selecting **File → Generate Project Documentation**.

10.5.6 Testing end-to-end

To test the orchestration end-to-end, start the orchestration, and send an appropriate HTTP request to the orchestration. Appropriate in this case means that you can use the filter created in “Reducing the list by creating a filter” on page 386 to reduce the number of customer records that flow if the SAP project has a lot of customers.

If you did not remove the terminate, as discussed in “Testing the filter” on page 388, do it now.

Testing successful inserts and updates

For the first end-to-end run, specify the filter so that it returns a few (two-to-five) records. If that test works as expected, increase the number.

To run the test:

1. Click the **Verify** tab, and start the orchestration.
2. Send an HTTP request using HTTP Post Utility to the orchestration. Specify an appropriate filter (for example 000001237) in the body, as shown in Figure 10-57.

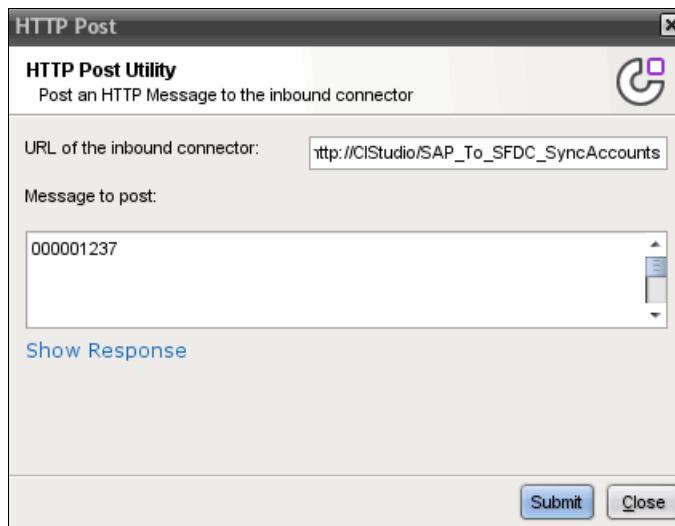


Figure 10-57 Call orchestration with filter 000001237

3. Take a look at the output of the orchestration. As shown in Figure 10-58 on page 393, the orchestration processed three SAP accounts and inserted or updated them to Salesforce.com successfully.

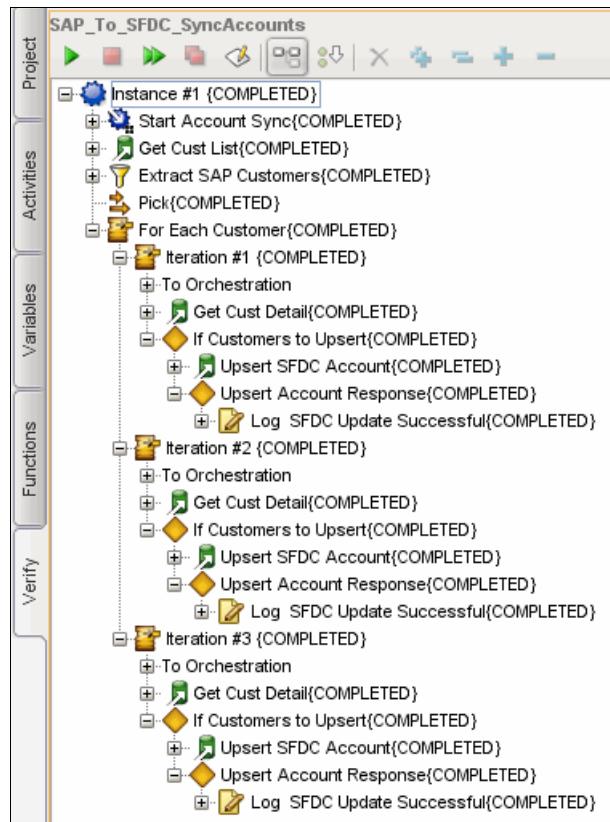


Figure 10-58 Successful test

You can expand each of the sections further to see the inbound and outbound mapping. For each variable, you can see the content at run time.

- Now log in to the Salesforce.com CRM system, and verify that the accounts were created successfully in Salesforce.com.
- Perform additional testing with any additional or updated accounts.

Test failures

This section discusses the types of scenarios where a failure can occur.

No matching record

Test a failure scenario by providing a search criteria in the HTTP request that does not return any matching records. As shown in Figure 10-59 on page 394, the For Each loop does not start because there is no record available. If you want to get an error message for such a scenario, add an if-statement just before the loop.

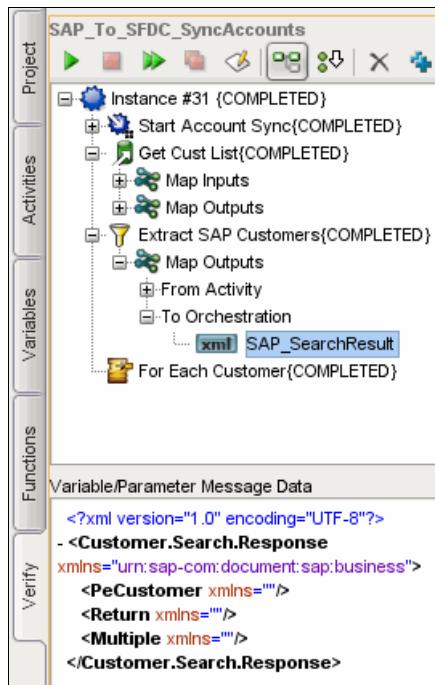


Figure 10-59 Test with no accounts

The Salesforce.com user rights not appropriate for update

If you have an integration user and an administrative user in Salesforce.com, change the profile of the integration user to read only for testing purposes. When you run the orchestration again, it fails as shown in Figure 10-60. An appropriate message is created.

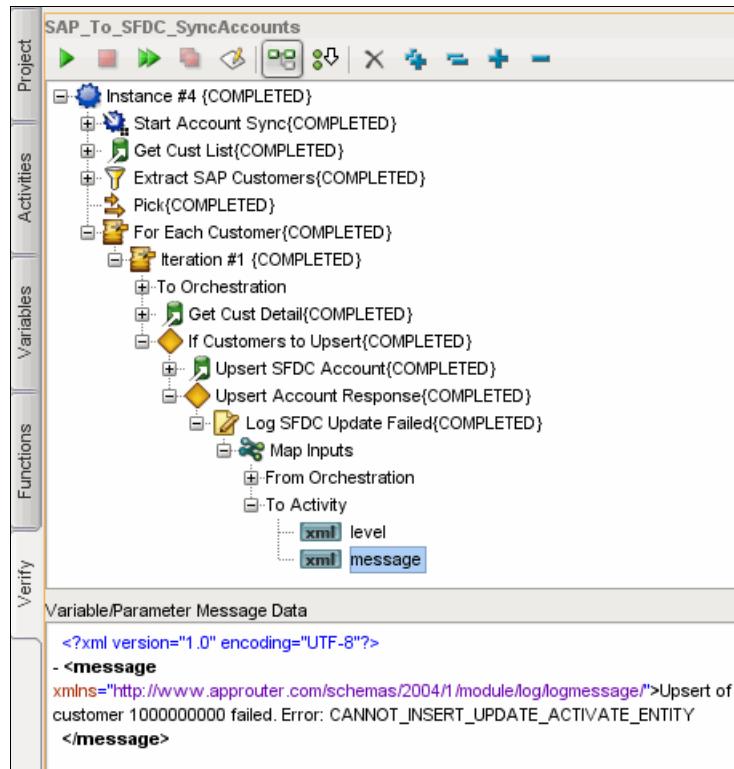


Figure 10-60 Test with upsert failed

SAP customer account without customer details

If a customer is deleted after the synchronization starts, you might run into the situation where the Get Cust List returns a customer number but there is no customer data available when running the SAP Get Details query.

Figure 10-61 shows the results in this situation. The orchestration failed in the Get Cust Detail activity, and the error is written in the global error catch block. Although it is good that the error was caught at all, this example shows that it might be better to put a try-catch block around the SAP Get Cust Detail activity, because this block can ensure that the whole synchronization does not fail when one SAP customer no longer exists.

Tip: You can force the same situation by tweaking the activity to retrieve an invalid customer number.

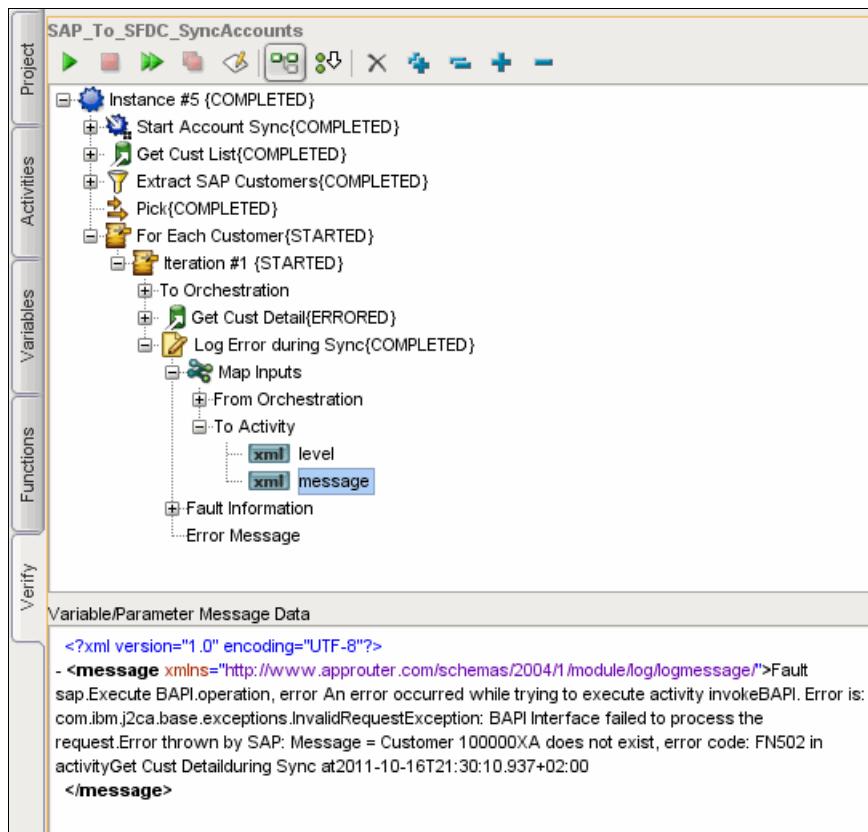


Figure 10-61 SAP Get Cust Detail failed

To avoid this situation, follow these steps:

1. Put the SAP Get Cust Detail activity into a Try-Catch block.
2. Rename the Try-Catch activity and the variables.
3. Put a Message Log activity into the catch block.
4. Because the Get Cust Detail was moved to the Try-Catch block, the scope of the SAP_CustDetail variable changed. To make the variable visible outside of the Try-Catch block, define a default value for the variable. To do so, click the **Variables** tab. Click the variable, and provide a default value, such as <Customer.GetDetail2.Response/>.

Figure 10-62 on page 396 shows the Try-Catch block.

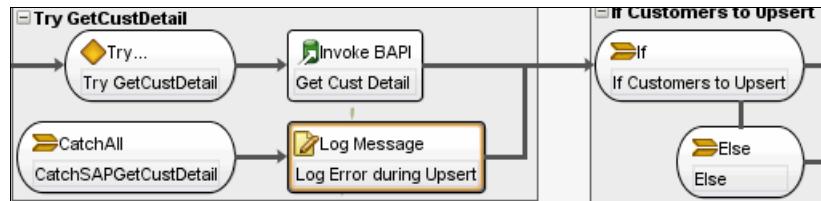


Figure 10-62 SAP Try-Catch block

If the Get Cust Detail activity fails, the orchestration goes into the Try-Catch block, and the message is logged. Then, the next activity, which is the If Customer To Upsert activity, validates whether or not the customer is updated.

This validation relies on the value in the SAP_CustDetail variable, which by default is set in the Get Cust Detail activity. However, in the failing scenario, the validation takes the default setting or any setting that was set before. This design is flawed, because if the Get Cust Detail activity fails, you do not want the orchestration to go into the validation. Instead you want the orchestration to continue with the next record in the loop.

To implement this logic, add the Continue activity just behind the Log Message in the catch block, as shown in Figure 10-63.

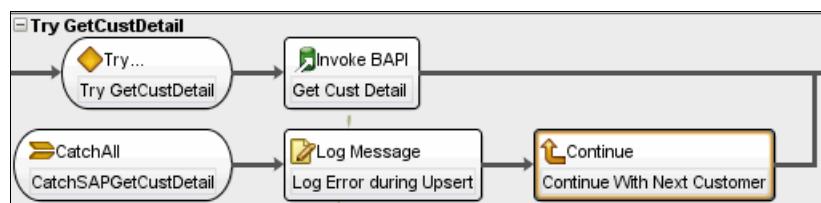


Figure 10-63 SAP Try-Catch block with continue

This configuration tells the orchestration to jump to the next item to process in the loop. Now, the handling of this error is better.

5. Test the orchestration again with a failing Get Cust Detail.

This time the orchestration catches the failure and jumps to the next item in the loop, as shown in Figure 10-64.

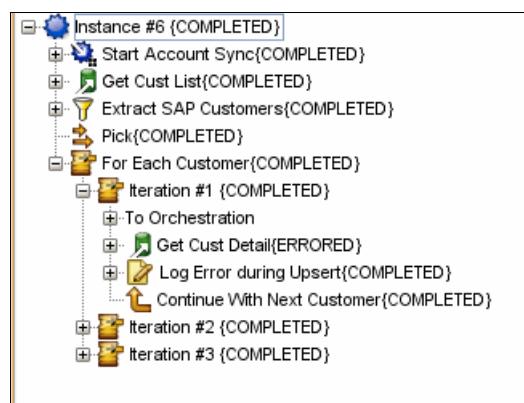


Figure 10-64 Test SAP Try-Catch block with continue

10.5.7 Using the Common Error Handler

Chapter 9, “Common error handlers” on page 349 describes the Common Error Handler. In Chapter 7, “Reusability with Template Integration Projects” on page 293, a TIP was created for the Common Error Handler. This TIP is available in the template repository. For this scenario, it is assumed that the Common Error Handler was published to the run time and is running. This section provides details about how to include the Common Error Handler in an orchestration.

Download the WSDL file

Begin by downloading the Common Error Handler WSDL file:

1. Download the WSDL file, and rename it to CommonErrorHandler.wsdl.
2. Import the WSDL into Studio.
3. Create a web services endpoint, and rename it to CommonErrorHandlerWS_Out.
4. Configure the endpoint to use the imported WSDL. For the location, specify the runtime of the Common Error Handler (data IP). Create a configuration property named CommonErrorHandlerLocation to contain the location.

The endpoint now looks similar to Figure 10-65, where the CommonErrorHandlerLocation is `http://<common_error_handler_runtime_data_ip>:80/ErrorLoggerRemote`.



Figure 10-65 Common Error Handler endpoint definition

Creating an Invoke Service activity

To create an Invoke Service activity in the global catch block:

1. Click the global catch branch to find the location for new activities, as shown in Figure 10-66.

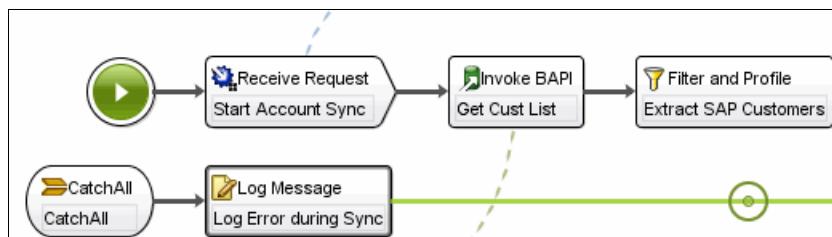


Figure 10-66 Insert location in global catch

- Click the CommonErrorHandlerWS_Out endpoint, and drag it to the location. Select **Invoke Service**, as shown in Figure 10-67.

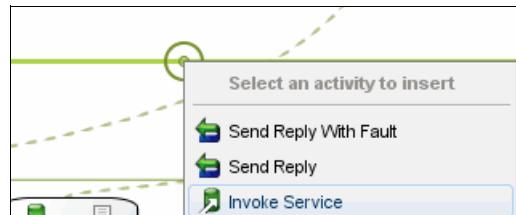


Figure 10-67 Insert web service invoke activity

- Click the web service, and rename it to Log Error.
- Click **Map Inputs**, and select the JobInfo and GlobalCatchFaultInfo variables as input.
- Create a link from the JobInfo variable to the CommonErrorHandler variable. The fields are mapped automatically based on the name.
- Create a link from the GlobalCatchFaultInfo variable to the CommonErrorHandler variable. The fields are mapped automatically based on the name.
- Compare the orchestration with that shown in Figure 10-68, and verify that all fields of the CommonErrorHandler variable are mapped.

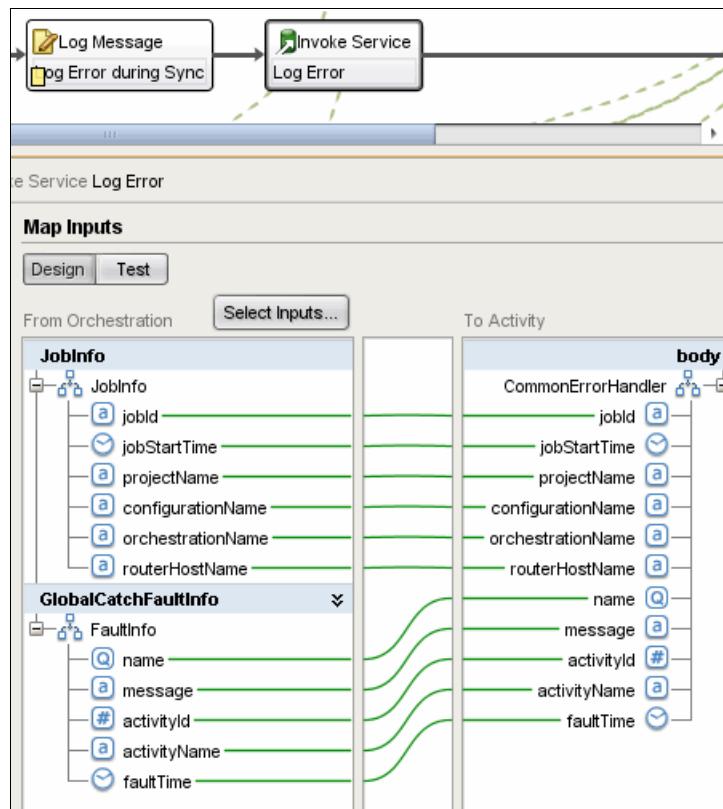


Figure 10-68 Configure Common Error Handler

Making the SAP Get Cust List activity fail

Now, force an error by making the SAP Get Cust List activity fail.

You can use several methods to make the activity fail. An easy method is to set the SAP_SalesOrg configuration property (renamed from SAPSalesOrg for this scenario) to an invalid value.

Follow these steps:

1. Set SAP_SalesOrg to 1000x.
2. Start the orchestration, and use the HTTP Post Utility to send a request.
3. The orchestration fails during the Get Cust List activity. In the global catch block, the Common Error Handler is called. View the orchestration output to verify that the Common Error Handler was invoked, as shown in Figure 10-69.

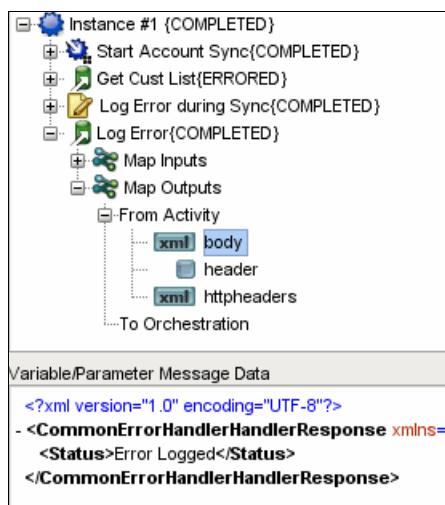


Figure 10-69 Test Common Error Handler

4. View the run time of the Common Error Handler.
5. Set the SAP_SalesOrg configuration property back to the original value.
6. Add the Common Error Handler invocation to the other Try-Catch blocks.

10.5.8 Synchronizing dedicated SAP customers to Salesforce.com

It does not make sense to use the orchestration as is to synchronize one dedicated account. The search will return all records, and you need only one record. In addition, you know the customer number.

If you need to synchronize single accounts, enhance the orchestration to handle a single record separately:

1. Add a new Receive Request activity that is identical to the Start Account Sync activity. Change the URL for the new activity to SAP_TO_SFDC_SyncOneAccount. To add another trigger activity, use the Pick Activity, which provides different branches for different Starter activities.
2. Move all the existing activities before the loop activity into the first branch. For details about which activities to move refer to Figure 10-70 on page 400.

3. With the move of the activities into the first branch, the scope of the SAP_SearchResult variable changes, meaning that the variable is no longer globally visible. To change this setting, set the default value for the SAP_SearchResult variable to <Customer.Search.Response/>.
4. Add a Map Variables activity to the second branch. The Map Variables activity maps the SAP_FilterList field into the SAP_SearchResult → Customer field. (Instead of the Map activity, you can also use a Read XML activity if you want to provide more than one customer number.)

Figure 10-70 shows the first part of the orchestration.

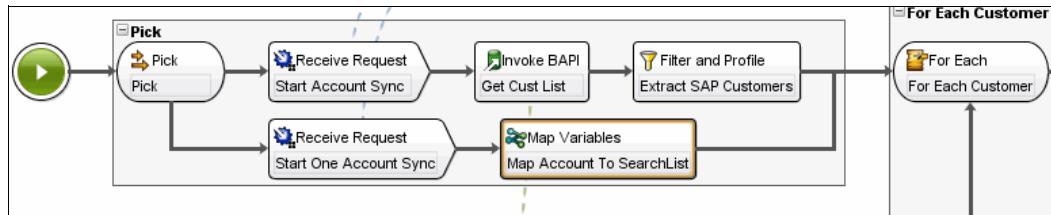


Figure 10-70 Extensions for synchronization of one account

10.5.9 Deploying and testing the orchestration on the Integration Appliance

The sections that follow explain how to deploy and test the orchestration on the Integration Appliance.

Runtime environment: The runtime environment in this scenario is assumed to be a virtual or physical appliance. If the runtime environment is Cast Iron Live, the orchestration needs only a change to the endpoint configuration to specify the Secure Connector for each endpoint that is behind a firewall, only for SAP in this case.

Publish from Studio

To publish the project from within Studio, follow these steps:

1. Verify if the project has not been published before. If it has been published before, it has to be in the Undeployed status or has to get a new version or name to be deployed successfully. If you are unsure whether the project was published previously, verify the status in the WMC.
2. Click **File → Publish Project**, as shown in Figure 10-71 on page 401.

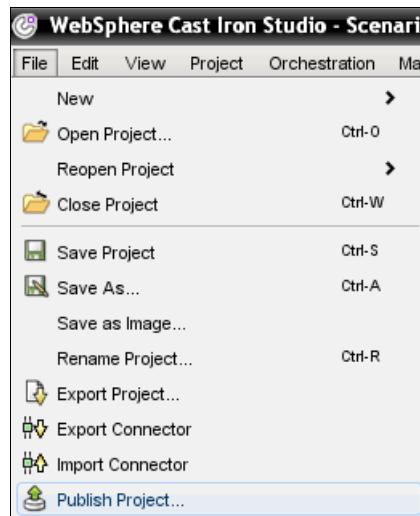


Figure 10-71 Publish the project

3. The project is validated automatically before it is published.
4. After successful validation, click **Yes** to save the project.
5. Enter or validate the host name for the target run time, the user name, and the password, as shown in Figure 10-72.

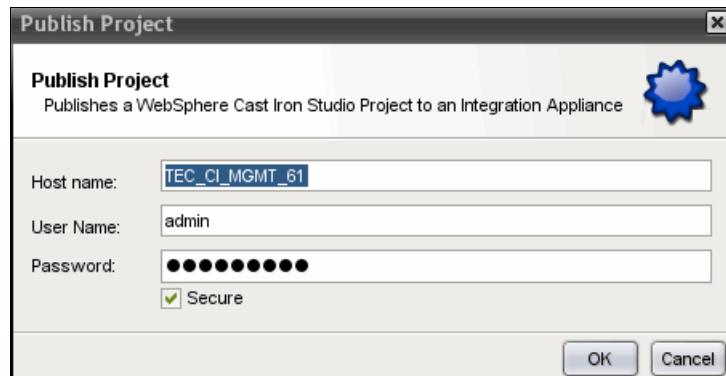


Figure 10-72 Login for publishing

To publish a project to the run time, the user must have the administrative or publisher role for the target run time.

Enter or verify the login data, and then click **OK**.

6. The project is published to the run time. If publishing was successful, a success message displays, as shown in Figure 10-73. Otherwise, you get an error message with some indication of why the publishing failed.



Figure 10-73 Project successfully published

Uploading a project using the Web Management Console

In some situations, it might not be practical to publish projects directly from Studio into the Integration Appliance. To publish a project from the WMC:

1. In Studio, export the project by selecting **File → Export Project**.
2. Specify the file name and directory for the project to export.
3. Wait until the success message displays, as shown in Figure 10-74.



Figure 10-74 Project successfully exported

4. In the WMC, select **Repository → Upload Project**.

5. Click **Browse**, and specify the .par file to be uploaded.

The project number and the project name and version are retrieved from the .par file, but you can change these settings, as shown in Figure 10-75.

6. Click **Upload** to upload the project.

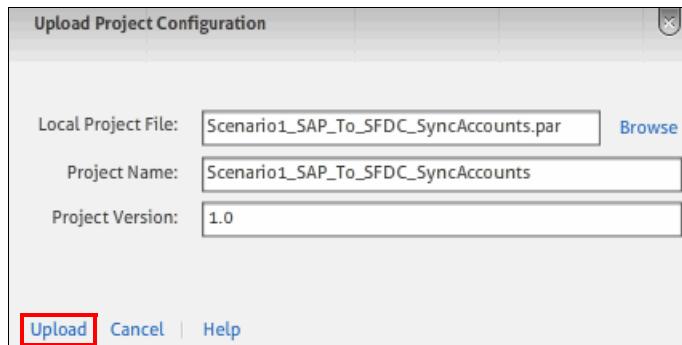


Figure 10-75 Upload a project using WMC

7. A success message is displayed, as shown in Figure 10-76. Be aware that this message closes quickly.



Figure 10-76 Project uploaded successfully

Testing the orchestration on the Integration Appliance

Now that the project is published successfully, go to the WMC of the run time to manage the project.

Tip: Although we use the term *project* in Studio, it is called a *configuration* in the run time.

As shown in Figure 10-77 on page 403, the status of the Scenario1_SAP_To_SFDC_SyncAccounts configuration is *Undeployed*.

Project Configurations						
Configuration	Running	Completed	Errored	Total	Actions	
▶ CommonErrorHandler (Running)	0	5	0	5		
▶ Scenario1_SAP_To_SFDC_SyncAccounts (Undeployed)	0	0	0	0		

Figure 10-77 Configuration status in WMC

Before you start the configuration, make sure that everything is in place, as follows:

1. Verify that the libraries for SAP are installed on the Integration Appliance:
 - a. Click **System** → **Upgrade** → **Update Connection Libraries**.
 - b. Ensure that the following SAP libraries are installed, as indicated in Figure 10-78:
 - sapjco3.jar
 - sapjco3.so
 - sapidoc3.jar
 If they are not installed, click **Update** to upload them; otherwise, click **Cancel**.
 - c. If the library names contain no number “3” in the name, for example sapidoc.jar, these .jar files are not SAP JCo Release 3 libraries but are older versions and are not supported. Uninstall any older libraries, and install appropriate libraries.

The screenshot shows two windows side-by-side. The left window is titled 'System Summary' and displays system information: Model (Cast Iron vA3000, Revision A), Version (Cast Iron Operating System 6.1.0.1.0000 (Thu Sep 15 14:16:52 UTC 2011)), and Serial Number (VMW5NKY7RKX54NMA). Below this is a 'System Summary' section with tabs for 'Upgrade Operating System' and 'Update Connector Libraries'. The right window is titled 'Update Connector Libraries' and contains instructions for managing connector libraries. It has three main sections: 'Connector' (listing Database, JD Edwards, JMS, PeopleSoft, SAP, Websphere MQ, Siebel), 'Installed Files' (listing sapjco3.jar, libsapjco3.so, and sapidoc3.jar, all marked with a red X), and 'Files to Add' (empty). At the bottom are buttons for 'Update', 'Reset', 'Cancel', and 'Help'.

Figure 10-78 Verify SAP libraries in the Integration Appliance

2. Click **Home** to return to the first panel, as shown in Figure 10-77.
3. Examine the configuration properties to ensure that the settings are correct. Click the configuration name to view the configuration details.

The Configuration Details panel displays all of the configuration properties that are defined in the project. In addition, it shows the settings for logging and other parameters. If the configuration is in the Unpublished status, click **Edit** to change the values, as indicated in Figure 10-79.

Verify that the settings are correct, and then start the configuration.

Configuration Details

Summary

Configuration: Scenario1_SAP_To_SFDC_SyncAccounts	# Orchestrations: 1
Status: Undeployed ➔	# Properties: 14
Last Published: 10/13/2011 06:55:27 AM	# Assets: 0
	# Downtimes: 0

Download

Orchestrations (1)

Name	Status	Logging Level	Log Synchronously	Max Simultaneous Jobs
SAP_To_SFDC_SyncAccounts	Enabled	Error Values	Disabled	10

Edit

Properties (14)

Name	Value
CommonErrorHandlerLocation	http://172.17.109.167:80/ErrorLoggerRemote
HTTP_In_Port	80
SAP_Client	800
SAP_Country	DE
SAP_Host	172.17.102.11
SAP_ListFilter_Default	*
SAP_Password	*****
SAP_SalesOrg	1000X
SAP_SearchFlag	2
SAP_SystemNumber	00
SAP_To_SFDC_SyncAccounts_URL	/SAP_To_SFDC_SyncAccounts
SAP_User	SAPUser
SFDC_Password	*****
SFDC_Username	SFDUser

Edit

Figure 10-79 Configuration details in WMC

Tip: If you start an orchestration in Studio on the Verify tab, only that orchestration is started. In the Integration Appliance, you start all orchestrations in the configuration at the same time. So, if for testing purposes you have different versions of the same orchestration in your configuration, disable the other versions, or the configuration will not start.

To disable an orchestration, go to the Configuration Details panel, and click **Edit**, as indicated in Figure 10-80. Then, disable the orchestration.

Configuration Details				
Summary				
Configuration: Scenario1_SFDC_To_SAP_SyncAccounts	# Orchestrations: 2			
Status: Running	# Properties: 11			
Last Published: 10/12/2011 11:12:06 PM	# Assets: 0			
	# Downtimes: 0			
Download				
Orchestrations (2)				
Name	Status	Logging Level	Log Synchrono	Max Simultaneous Jobs
SFDC_To_SAP_SyncAccounts	Enabled	Error Values	Disabled	10
SFDC_To_SAP_SyncAccounts_Query	Disabled	Error Values	Disabled	10
Edit				

Figure 10-80 WMC, disable orchestrations

4. To test the orchestrations, use Studio and the HTTP Post Utility. However, this time, specify the data IP address of the Integration Appliance as the target IP address.
5. Send several HTTP requests to the Integration Appliance, and after each request, return to the WMC to check the statistics with regard to running, completed, and finished configurations.
6. View the log by clicking **Logs** → **System Log**. Validate whether warnings or errors occurred.

10.5.10 Initial load of thousands of records

This scenario as described thus far is a good example for when you have a small amount of data or if you want to be flexible in how to validate or modify the data when moving data from SAP to Salesforce.com. However, if you need to synchronize thousands of records, the synchronization one-by-one in Salesforce.com takes too long. In addition, Salesforce.com has limits for the total API requests per each 24-hour period, and each use of the upsert activity is one API call.

For details about the limits refer to:

http://www.salesforce.com/us/developer/docs/api/Content/implementation_considerations.htm

Consider the following approaches to workaround these limitations:

- ▶ Split the load over separate days, for example, use the filter to split the customers into separate groups.
- ▶ Do multiple updates in one activity. The upsert activity allows you to update multiple accounts in one call, as described in detail here.
- ▶ Use the Salesforce.com bulk API. Salesforce.com provides a bulk API for asynchronous updates. The bulk API allows you to send a bulk request in one call and then retrieve the response within another call. There are templates available that demonstrate this method.

If the number of customer records to be synchronized is large, it does not make sense to read the records from SAP one-by-one, to validate them one-by-one, or to insert or update them into Salesforce.com one-by-one.

The idea here is to use APIs that allow the management of recurring elements. The orchestration is simplified, as shown in Figure 10-81.

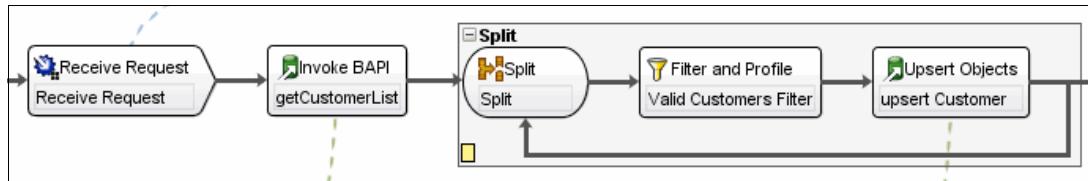


Figure 10-81 The orchestration modified for updating chunks

The orchestration contains the following activities:

- ▶ The Receive Request activity

It receives a request with or without a search criteria and triggers the orchestration.

- ▶ The SAP activity Get Customer List.

Previous examples used the Customer Search function to retrieve a list of customer numbers. Now, the orchestration uses the Customer2 GetList function, which returns a recurring element with all customers that match the criteria. With the customer number, it also returns the basic data, such as name, address, and so on.

- ▶ The Split activity

Although SAP can return hundreds of account records in one chunk, it does not make sense to send them in one chunk synchronously to Salesforce.com. So, you can use the Split activity to split the recurring element into chunks of an appropriate size (for example 20, 50, or 100 records). The Split activity is similar to a loop but returns a recurring element with a subset of the original elements. Refer to Figure 10-82.

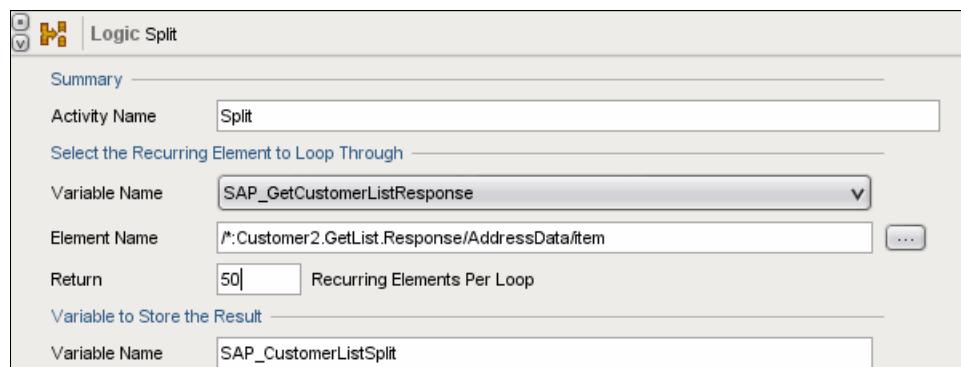


Figure 10-82 Use the Split activity

- ▶ The Filter and Profile activity

Previous examples use the Filter and Profile activity to filter the customer numbers based on a search criteria. Now, the orchestration uses the Filter activity to validate multiple datasets in one step. Figure 10-83 shows the filter criteria.

Filter Expression			
	Left Hand Expression	Op	Right Hand Expression
Where	CUSTOMER	!=	"
and	NAME	!=	"

Figure 10-83 Filter criteria for chunks

The Filter activity returns a recurring element for valid accounts, a recurring element for invalid accounts, and statistics.

- ▶ The Upsert activity

As before, the Upsert activity inserts or updates accounts in Salesforce.com.

The activity returns a recurring element that contains details for each account record whether or not the upsert was successful.

Consider the following additional information:

- ▶ The orchestration, as shown in Figure 10-81 on page 406, is not complete because it does not contain any kind of error handling.
- ▶ If you want to do more complex validation, replace the Filter activity with a loop that validates each record separately and that flags each record as good or bad. Then, filter the good records and use those records for the upsert.

10.6 Use Case 2: Synchronizing from Salesforce.com to SAP

The scenario described in 10.5, “Use case 1: Synchronizing from SAP to Salesforce.com” on page 360 uses a TIP to build an orchestration that synchronizes customer data from SAP to Salesforce.com. In this use, an orchestration will be built that allows data synchronization from Salesforce.com to SAP. Although you can reuse parts of existing TIPs for this example, this section demonstrates the development of an orchestration without a TIP.

10.6.1 Building the orchestration

Building the orchestration requires the following steps:

- ▶ Retrieving an HTTP request with the time for synchronization
- ▶ Getting the updated accounts from Salesforce.com
- ▶ Looping through all accounts
- ▶ Extracting and validating the SAP customer number
- ▶ Enriching Salesforce.com account records with data from SAP
- ▶ Updating the customer record in SAP
- ▶ Handling invalid Salesforce.com accounts
- ▶ Handling exceptions and errors

Additional resource: You can use the poll activity for Salesforce.com. However, a regular polling for updates can be expensive in terms of resources. Alternatively, for an HTTP request, you can configure Salesforce.com to send a request for each update. This type of configuration is out of the scope of this book. You can find an example in the TIP repository.

This scenario creates the orchestration within its own project instead of adding it to the existing project. Using separate projects for each direction of synchronization allows you to manage and update both independently. Some assets from the first project are reused in the second project to streamline the development.

1. Open the project created in 10.4, “Overview of the use cases for this scenario” on page 360, and save it as another project name. This scenario uses SAP_To_SFDC_syncAccounts as the name for the first project and SFDC_To_SAP_syncAccounts as the name for the new projects.
2. Click **Project → Project Settings**, and change the description of the project to something meaningful.

3. Remove the existing orchestrations from the new project.
4. Create a new orchestration, and rename it to SFDC_To_SAP_SyncAccounts.

Now, you can continue to build the parts of the orchestration.

Retrieving an HTTP request with the time for synchronization

We do not want to synchronize all records from Salesforce.com but only those that were changed (created or updated) since the last synchronization occurred. The time of the last synchronization will be delivered within the HTTP request as part of the body.

To retrieve an HTTP request in the orchestration:

1. Click the **HTTP_In** endpoint, and drag it to the canvas.
2. Select **Receive Request**, as shown in Figure 10-84.

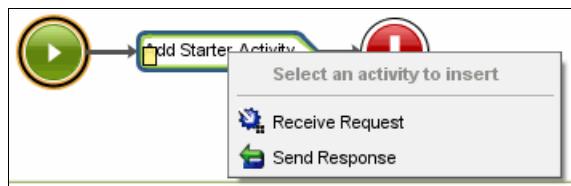


Figure 10-84 Add Receive Request

3. In the Checklist, click **Summary**, and set Activity Name to **Receive Sync Time**.
4. In the Checklist, click **Pick Endpoint**, and set the endpoint to **HTTP_In**, if not already set.
5. In the Checklist, click **Configure**:
 - a. Set the URL to **SFDC_To_SAP_SyncAccounts**, and then create a configuration property for the URL.
 - b. Leave the message type to **text**.
 - c. Select **Requires a Reply**.
6. In the Map Outputs section:

Create a new variable called **SyncTimeString**, and map this variable to the HTTP Request body. Create another variable named **SyncTimePattern** to contain the format pattern. Use the **yyyy/MM/dd HH:mm:ss** pattern, which means the date and time string has the following format:

year/month/day/hours/minutes/seconds

To map outputs:

- a. Click **Select Outputs**. Because you want to create a new variable, click **New**.
- b. Under Primitive Types, click **string**, and then click **Next**. Enter **SyncTimeString** for the variable name, and then click **Finish**.
- c. The **SyncTimeString** variable is selected as the output variable, so click **OK**.
- d. Create a link from the element **body** to the **SyncTimeString** variable.
- e. Click **Select Outputs**. Click **New**, and create another variable of type **string**. Enter **SyncTimePattern** for the variable name, and then click **Finish**.
- f. The **SyncTimePattern** variable is selected as the output variable, so click **OK**.

- g. Right-click the SyncTimePattern variable, and select **Define Default Value**. Set the default value to yyyy/MM/dd/HH/mm/ss.
- h. Click the bullet to create a configuration property called *SyncTimePattern*.

Figure 10-85 shows the final mapping.

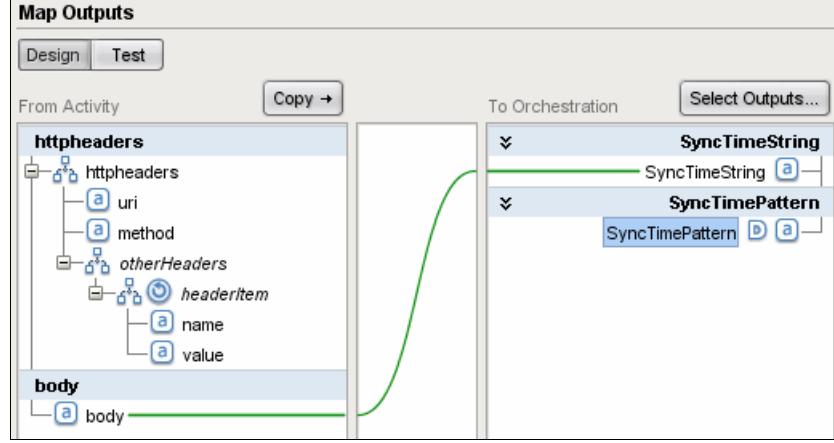


Figure 10-85 Receive Request Map Outputs

Parsing the received SyncTimeString based on the pattern

To parse the received SyncTimeString:

1. Drag a Map Variables activity to the canvas, to the right of the Receive Request activity.
2. Click the name of the activity, and rename it to Parse SyncTimeString.
3. Click **Select Inputs**, and select **SyncTimeString**.
4. Click **Select Inputs**, and select **SyncTimePattern**.
5. Click the Functions tab, and drag the Read Date String function to the middle of the map.
6. Link first the SyncTimeString variable and then the SyncTimePattern variable to the Read Date String function.
7. Click **Select Outputs → New**, and create a variable of type dateTIme.
8. Enter SyncTime for the variable name, and then click **Finish**.
9. The SyncTime variable is selected as the output variable, so click **OK**.

10. Link the Read Date String function to the SyncTime variable. Double-click the Read Date String function. Verify that the mapping is the same as that shown in Figure 10-86, and then click **OK**.

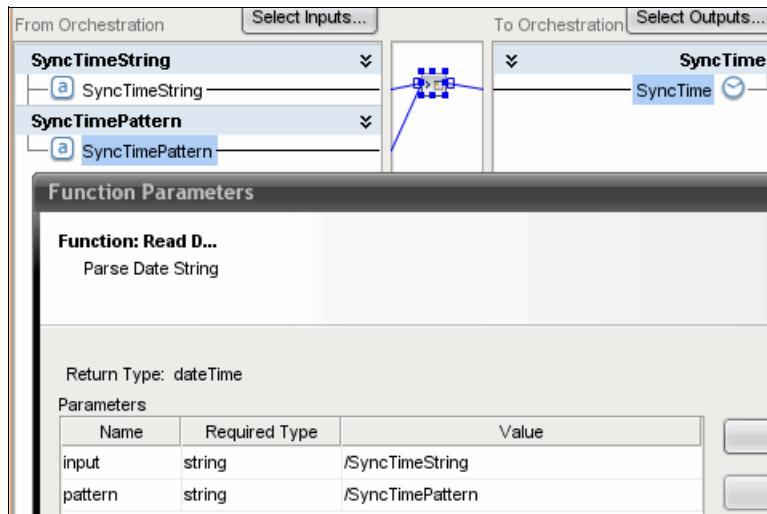


Figure 10-86 Parse SyncTime

11. Right-click the Read Date String function, and select **Apply Function Graph**.

Sending a customized response

The entire synchronization process might take longer than the HTTP Request timeout. Therefore, send a customized response that tells that the request was received:

1. Click the **HTTP_In** endpoint, and drag it on the canvas right to the **Parse SyncTimeString** activity. Select the **Send Response** option.
2. Click the **Send Response** activity name, and rename it to **Send Acceptance**.
3. Click **Map Inputs**, and create a return message using these steps:
 - a. Click **Select Inputs** and select the **SyncTime** variable.
 - b. Click the Functions tab and drag the **Concatenate** function to the middle of the map.
 - c. Create a link from the **SyncTime** variable to the **Concatenate** function.
 - d. Create a link from the **Concatenate** function to the **body** variable.
 - e. Double-click the **Concatenate** function, and create a message, such as
Sync request has been retrieved, sync time: xxx

For details about how to create a message, look at Figure 10-87 on page 411. Click **OK**, and apply the function graph.

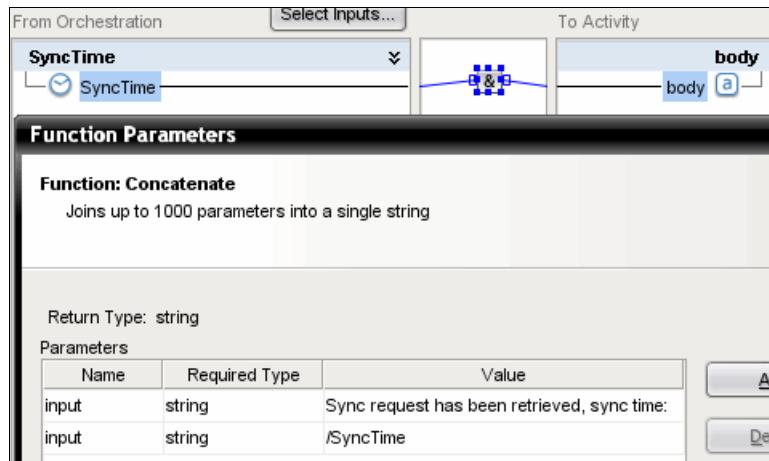


Figure 10-87 Create response message

4. Save the orchestration (**File → Save**).

The entire orchestration now looks like Figure 10-88. Although it is not a complete orchestration from a business point of view, it is a complete orchestration from a technical point of view and can now be tested.

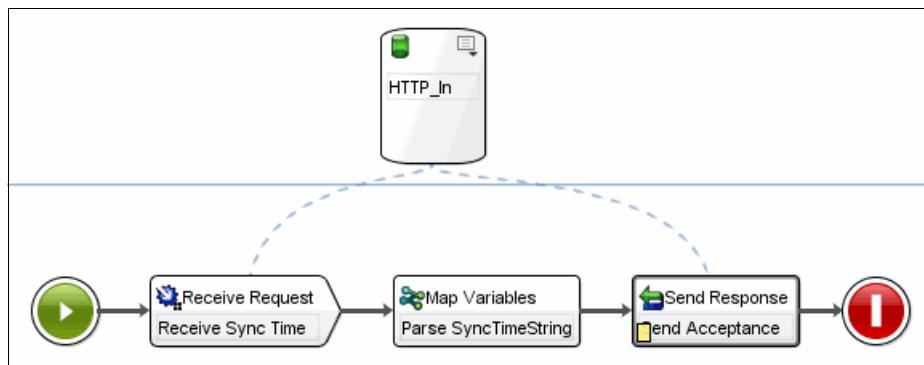


Figure 10-88 Simple request-response flow

Starting and testing the orchestration

The next step is to test the orchestration. Information about testing is in 10.5.6, “Testing end-to-end” on page 392.

1. Click the Verify tab and start the orchestration.
2. Test the orchestration using the HTTP Post utility with the following settings:
 - URL: `http://<yourhostname>/SFDC_To_SAP_SyncAccounts`
 - HTTP body: `2011/10/01/10/00/00`
3. Click in the HTTP Post Utility on the Show Response panel, and then click **Submit**. A message similar to the following message displays:

`Sync request has been retrieved, sync time: 2011-10-01T10:00:00+02:00`

Refer to Figure 10-89 for details.

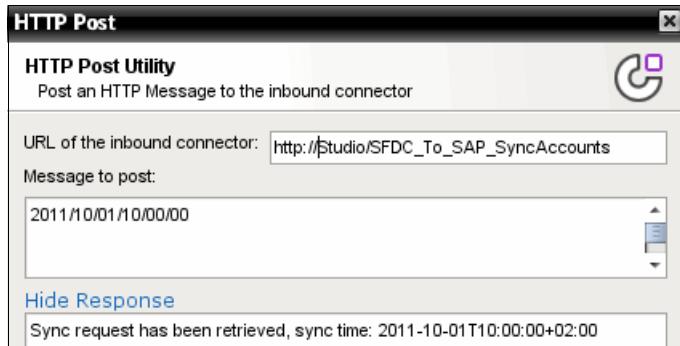


Figure 10-89 Simple request-response flow test

Now, start the orchestration again, and this time test it with the test HTTP body content.

This time, the test fails, as shown in Figure 10-90, and no response is sent back to the calling application. Click **OK** to continue.

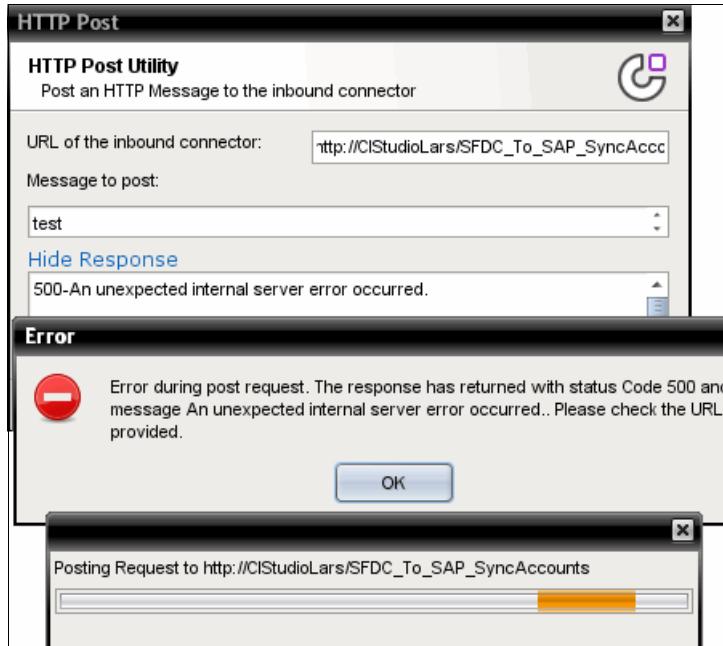


Figure 10-90 Date parsing error

Putting a Try-Catch block around the activity

To avoid error handling, put a Try-Catch block around the Parse SyncTimeString activity:

1. Drag the Try activity to the left of the Parse SyncTimeString activity.
2. Rename it to Try Parsing by clicking the description.

Parsing the time string

1. Move the Parse SyncTimeString activity into the Try branch, and then click it to open the configuration.

2. The SyncTime variable is defined in the activity. Because this activity is now part of the Try-Catch block, it is not visible outside of the Try-Catch block. It must be visible outside of the Try-Catch block, so you need to set a default value for the variable as follows:
 - a. Go to the Variables tab. Select the SyncTime variable.
 - b. In the lower corner, set the Default Value to 2011-10-01T10:00:00+02:00. This value is a valid value for the variable as demonstrated in “Starting and testing the orchestration” on page 411. You can also replace this value with another value. However, remember that the variable is set based on the value in the Receive Sync Time activity.
3. Move the Send Acceptance activity in the Try branch.
4. Click the Catch activity, and rename the following activities and variables:

The Catch activity to	CatchSyncTimeParse
The FaultName variable to	SyncTimeParseFaultName
The FaultData variable to	SyncTimeParseFaultData
The FaultInfo variable to	SyncTimeParseFaultInfo

Log error messages

1. Add a Log Message activity to the Catch branch, and rename it to Log Parse SyncTime Error.
2. Click **Map Inputs**, and create an error message:
 - a. Click **Select Inputs**, and select the SyncTimeParseFaultInfo, SyncTimeString, and SyncTimePattern variables.
 - b. Click the Functions tab, and drag the Concatenate function in the middle of the map.
 - c. Create a link from all elements of the SyncTimeParseFaultInfo variable to the Concatenate function.
 - d. Create a link from the SyncTimePattern and SyncTimeString variables to the Concatenate function.
 - e. Create a link from the Concatenate function to the message variable.
 - f. Double-click the Concatenate function, and create an error message, such as the following message:
 SyncTime Parsing Error aaa, message: bbb in activity ccc (ID: dd), at time: eee, expected datetime pattern: fff, received datetime: ggg.
 - g. Click **OK**, and apply the function graph.

Sending a response

1. Add another Send Response activity to Catch branch, and rename it to Send SyncTime Failure.
2. Click **Map Inputs**, and create a return message using these steps:
 - a. Click **Select Inputs**, and select the SyncTimeString and SyncTimePattern variables.
 - b. Click the Functions tab, and drag the Concatenate function to the middle of the map.
 - c. Create a link from the SyncTimeString and SyncTimePattern variables to the Concatenate function.
 - d. Create a link from the Concatenate function to the body variable.
 - e. Double-click the Concatenate function, and create a message, such as
 Sync request failed due to invalid time format, received time: xxx, defined pattern: yyy. Then click on OK and apply the function graph.

Starting and testing the organization with your changes

Figure 10-91 shows how the orchestration that you just built should look.

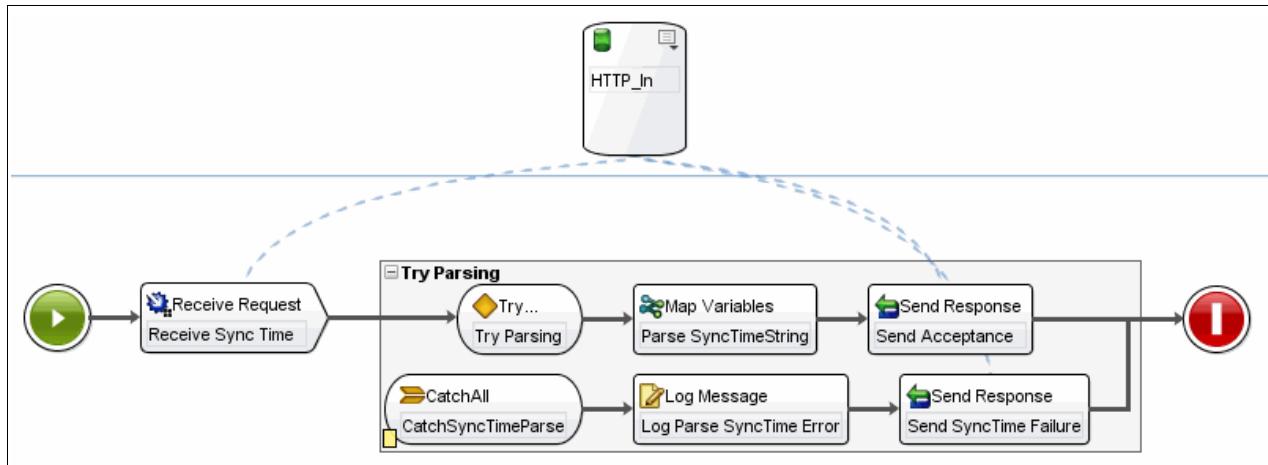


Figure 10-91 Resolve parsing error with try-catch

Start the orchestration, and test it again with the test HTTP body. This time the orchestration sends a response back with the following content:

Sync request failed due to invalid time format, received time: test, defined pattern: yyyyymm/MM/dd HH:mm:ss.

Remember to save the orchestration.

Getting the updated accounts from Salesforce.com

To get the updated Salesforce.com accounts, use the Get Updated Objects activity. Configure the activity as follows:

1. Click the Salesforce.com endpoint, and drag it to the canvas to the right of the Try-Catch block.
2. Select **Get Updated Objects**, as shown in Figure 10-92.

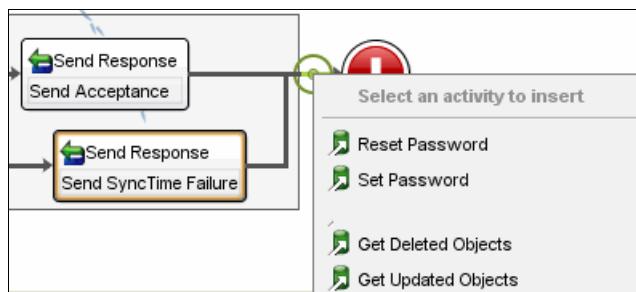


Figure 10-92 Add Get Updated Objects

3. In the Checklist, click **Summary**, and set Activity Name to Get Updated Accounts From Salesforce.com.
4. In the Checklist, click **Pick Endpoint**, and set the endpoint to Salesforce.com if not already set.
5. In the Checklist, click **Configure**.

To configure the activity, connect to Salesforce.com, and retrieve a list of available objects, including the standard objects and any custom objects. Follow these steps:

- Click **Browse**. Select the Account object, as shown in Figure 10-93, and click **OK**.

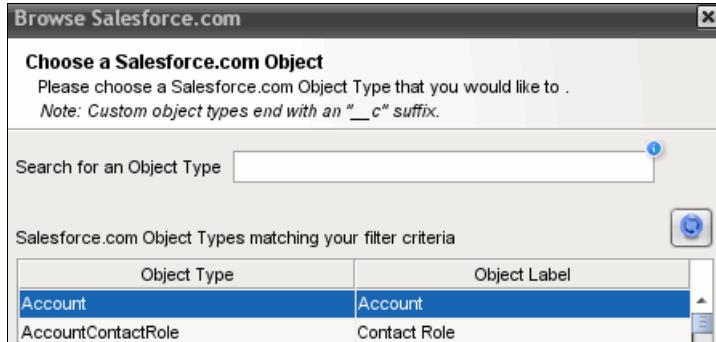


Figure 10-93 Browse Salesforce.com objects

- The Salesforce.com connector returns the list of fields that are available in the Account object. This list includes standard and custom fields. Select the fields that you need for the integration. Be aware that the more fields that you select the more performance overhead you produce. At a minimum, however, select the fields shown in Figure 10-94 because you need them for the integration. The list in the figure is reduced for better visibility.

Configure			
Salesforce.com Object Type			
Object Type	<input type="text" value="Account"/>	<input type="button" value="Browse..."/>	
Object Field Preferences			
Select the fields you would like to map by selecting the check box next to the field name. <small>Note that custom fields have an <code>_c</code> suffix for their field name.</small>			
<input type="checkbox"/>	Name	Label	Type
<input checked="" type="checkbox"/>	Id	Account ID	id
<input checked="" type="checkbox"/>	Name	Account Name	Name
<input checked="" type="checkbox"/>	BillingStreet	Billing Street	Text Area(255)
<input checked="" type="checkbox"/>	BillingCity	Billing City	Text(40)
<input checked="" type="checkbox"/>	BillingState	Billing State/Province	Text(20)
<input checked="" type="checkbox"/>	BillingPostalCode	Billing Zip/Postal Code	Text(20)
<input checked="" type="checkbox"/>	BillingCountry	Billing Country	Text(40)
<input checked="" type="checkbox"/>	Phone	Account Phone	Phone
<input checked="" type="checkbox"/>	Fax	Account Fax	Phone
<input checked="" type="checkbox"/>	AccountNumber	Account Number	Text(40)
<input checked="" type="checkbox"/>	CreatedDate	Created Date	Date/Time
<input checked="" type="checkbox"/>	CreatedByld	<input checked="" type="checkbox"/> Created By ID	Lookup(User)(User)
<input checked="" type="checkbox"/>	LastModifiedDate	Last Modified Date	Date/Time
<input checked="" type="checkbox"/>	LastModifiedByld	<input checked="" type="checkbox"/> Last Modified By ID	Lookup(User)(User)
<input checked="" type="checkbox"/>	ExtAccountId_c	ExtAccountId	Text(15) (External ID) (UN...)
<input checked="" type="checkbox"/>	SalesOrganisation_c	SalesOrganisation	Text(10)

Figure 10-94 Get Updated Objects—Fields

- Map the following inputs:

- Click **Select Inputs**, and select **SyncTime**.
- Create a link from the SyncTime variable to the startTime variable.

7. Map Outputs.

The left part of the map shows a variable named *objects*. The *objects* variable contains the recurring Account element, which has as elements all the fields that you selected previously.

Because you need all fields from the elements of the *objects* variable, copy the entire structure:

- a. Click the *objects* variable, and then click **Copy**.
 - b. Select **objects**, and click **Create** to create a variable with the name *objects*.
 - c. Click the Variables tab, and rename the variable to SFDC_UpdatedAccounts.
8. At this point, you completed the steps to retrieve the updated accounts from Salesforce.com based on SyncTime. Your orchestration will look similar to that shown in Figure 10-95. (The Try-Catch block is collapsed for better visibility in this figure.)

Save the orchestration.

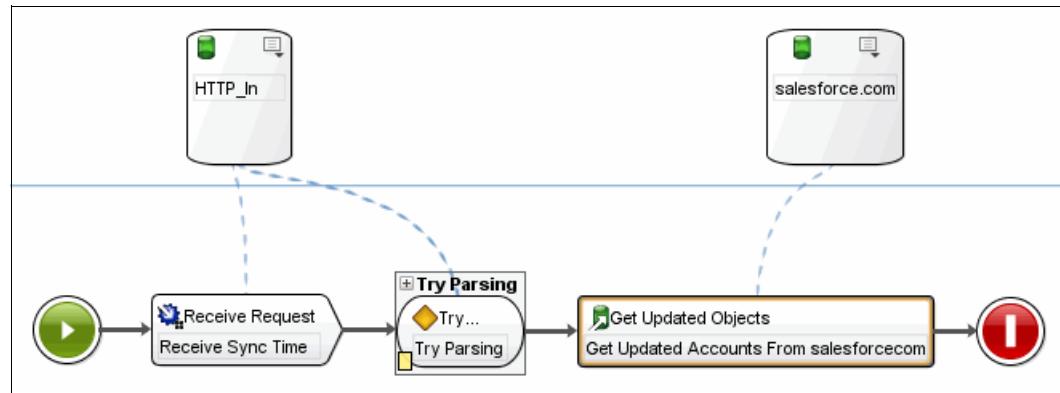


Figure 10-95 Orchestration with Get Updated Objects

9. Complete the following test:

- a. Go into the Salesforce.com portal, and update some accounts.
- b. Start the orchestration.
- c. Use the HTTP Post Utility, and send in the body a current sync time in the format *yyyy/MM/dd HH:mm:ss*.

All the accounts that were changed since the last synchronization (the time included in the HTTP request) are returned. In Studio, compare your results with those shown in Figure 10-96.

```

<?xml version="1.0" encoding="UTF-8"?>
- <objects xmlns="urn:sobject.partner.soap.sforce.com"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <Account>
  <Id>001A000000lGamlAE</Id>
  <Name>Customer002</Name>
  <BillingStreet>Customer002_Street</BillingStreet>
  <BillingCity>Customer002_Street</BillingCity>
  <BillingState>02</BillingState>
  <BillingPostalCode>20002</BillingPostalCode>
  <BillingCountry>DE</BillingCountry>
  <Phone>12345678-002</Phone>
  <Fax xsi:nil="true"/>
  <AccountNumber>CUSTOMER002</AccountNumber>
  <CreatedDate>2011-10-09T21:28:13.000Z</CreatedDate>
  <CreatedById>005A000000011polAA</CreatedById>
  <LastModifiedDate>2011-10-11T05:32:04.000Z</LastModifiedDate>
  <LastModifiedBy>005A0000000TTeJIAW</LastModifiedBy>
  <ExtAccountId_c>SAP_CUSTOMER002</ExtAccountId_c>
  <SalesOrganisation_c>1000</SalesOrganisation_c>
</Account>
- <Account>
  <Id>001A000000lGcNIAU</Id>
  <Name>Customer003</Name>

```

Figure 10-96 Test with Get Updated Objects

Looping through all accounts

This scenario updates accounts in SAP only if the account already exists in SAP. So, you must validate for each account whether or not it is an SAP account. If the account is an SAP account, the orchestration updates SAP with the new account data from salesfroce.com.

To update a record in SAP, you must deliver mandatory data, such as sales organization, name, city, postal code, street, country language, and currency. Although most of the data is available in Salesforce.com, some data is not available and must be read from SAP before updating SAP.

Because the BAPI calls to SAP are on a per customer basis, the orchestration cannot do a bulk query to get the missing data for all customers in one query. Instead, it must loop through the list of customers and enhance the data one-by-one.

To implement this logic:

1. Drag a For Each activity to the right of the Get Updated Objects activity.
2. Change the settings of the For Each activity as follows:
 - a. Rename the Activity Name to Loop Through Salesforce.com Accounts.
 - b. Rename the Variable Name to SFDC_UpdatedAccounts.
 - c. For the Element Name, click the button to the right of the field, and select **Account**.
 - d. Rename the Single Element Variable Name to SFDC_OneAccount.
 - e. Compare your settings to those shown in Figure 10-97.

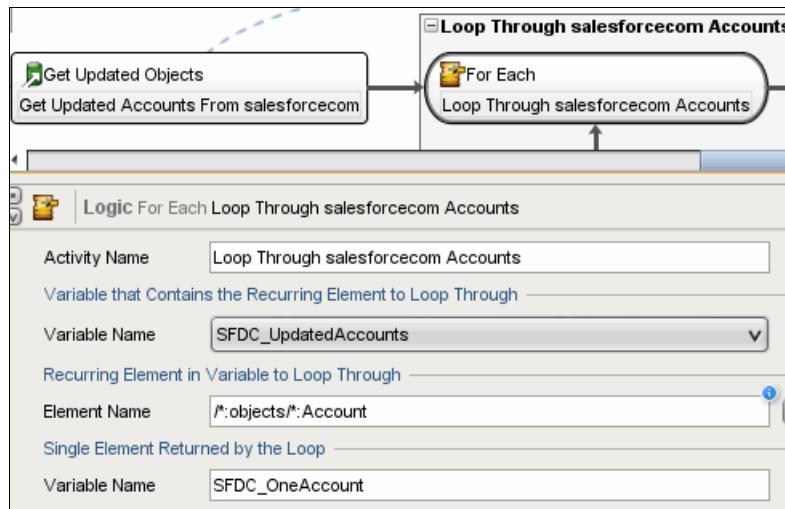


Figure 10-97 For Each loop definition

3. Save the project.

Extracting and validating the SAP customer number

This section explains how to extract and validate the SAP customer number.

The Salesforce.com account can be validated as an SAP account by validating the external ID field. This field must use the SAP_xxx format, where xxx is an SAP customer number. This example does not validate whether the SAP customer number exists in SAP but does validate the overall format. This scenario first extracts the SAP customer number and then validates it.

To validate the account:

1. Drag a Map Variables activity to the right of the For Each activity but within the loop. Click the activity name, and change it to Extract SAP Customer Name.
2. Click **Select Inputs**, and select the SFDC_OneAccount variable.
3. Click **Select Outputs**, and create a new string variable for the SAP number by clicking **New**.
 - a. Under **Primitive Types**, click **string**, and then click **Next**.
 - b. Enter SAP_CustomerNumber for the variable name, and click **Finish**.
 - c. The SAP_CustomerNumber variable is selected as the output variable, so click **OK**.
4. To extract the SAP customer number, use the Substring After function. This function looks for a defined string and returns, as a result, the string after that. In this case, look for SAP_, so that Substring After returns from SAP_xxx just xxx. Follow these steps:
 - a. Click the Functions tab, and drag the Substring After function to the middle of the map.

- b. Create a link from the ExtAccountID__c element from the SFDC_OneAccount variable to the Substring After function.
- c. Create a link from the Substring After function to the SAP_CustomerNumber variable.
- d. Double-click the Substring After function, and insert as matchString the value SAP_.

The mapping now looks similar to that in Figure 10-98.

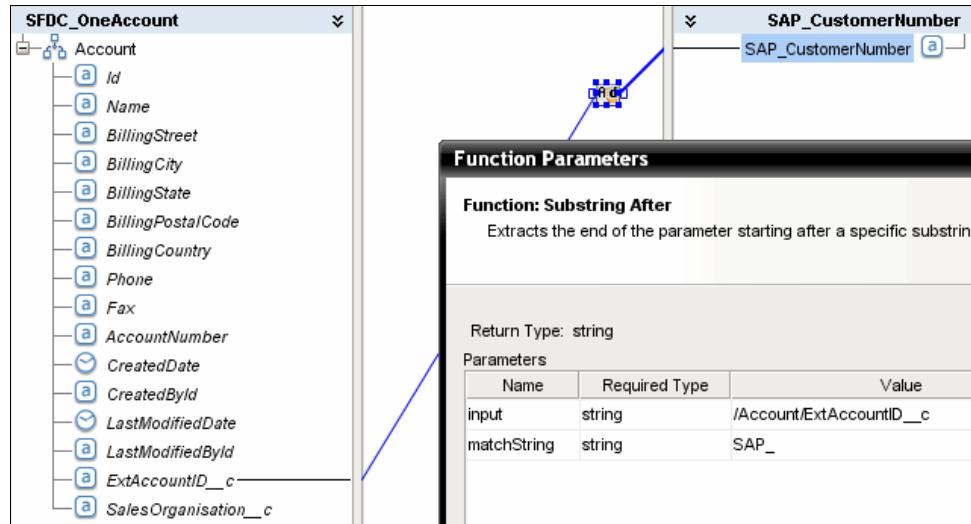


Figure 10-98 Extract SAP customer number

- e. on the function and select **Apply Function Graph**.
5. Validate whether the value that was extracted to SAP_CustomerNumber is not empty:
 - a. Drag an If-Then activity to the right of the Map Variables function but within the loop.
 - b. Click the If part of the activity, and change the name to If SAP Customer.
 - c. For the Where clause, select for Left Hand Expression the SAP_CustomerNumber variable, for Operation the != option, and for the Right Hand Expression enter two single quote marks (') as the value.
 - d. Click the Else part of the activity, and change the name to No SAP Customer.
 6. Save the project.

Enriching Salesforce.com account records with data from SAP

As noted previously, this scenario needs additional data from SAP that is not stored in Salesforce.com. To get this data, use the Invoke BAPI activity with the Customer2 business object and the GetDetail2 method. To configure the SAP activity:

1. Click the SAP endpoint, and drag it on the canvas to the right of the If activity but within the If block.
2. Select **Invoke BAPI**, as shown in Figure 10-99 on page 420.

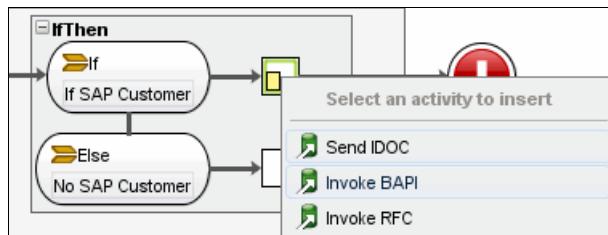


Figure 10-99 SAP activities

3. In the Checklist, click **Summary**, and set Activity Name to Get Customer Details From SAP.
4. In the Checklist, click **Pick Endpoint**, and set the endpoint to **SAP** if not already set.
5. In the Checklist, click **Configure**.

To configure the activity, connect to SAP and retrieve a list of available objects, including the standard objects and custom objects. Select the related method from that list.

To configure the activity:

- a. Click **Browse**. The Obtaining list of Business Objects from SAP message displays for a short time, and then the BAPI list displays.
- b. Click the **Business Object Repository** line, and then enter the first letter of the search string customer2. In the Search For entry field, enter the next letters of the search string. While you are entering the business object name, the cursor moves to the right position in the list, as shown in Figure 10-100.

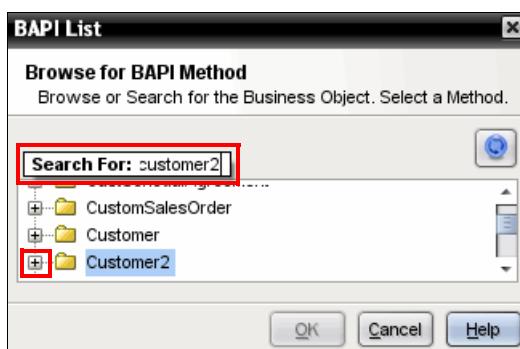


Figure 10-100 Browse SAP Business Objects

- c. Expand the Customer2 object by clicking the plus sign (+).
- d. Select the GetDetail2 method, and click **OK**.

A Please Wait notification displays with the Generating schemas statement. Compare your configuration panel with that shown in Figure 10-101 on page 421.

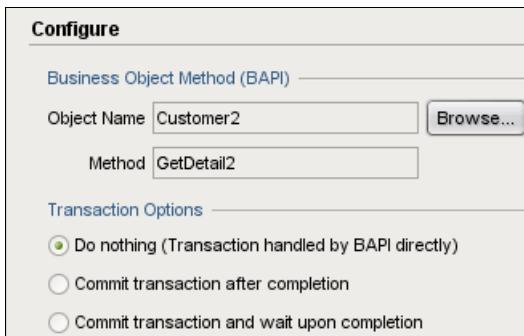


Figure 10-101 Use case SFDC To SAP, configure Customer2 Get Detail

6. In the Checklist, click **Map Inputs**:

Map the SAP account number into the appropriate input field:

- Click **Select Inputs**, and select **SAP_CustomerNumber**.
- Create a link from the SAP_CustomerNumber variable to the @CustomerNo element of the request variable.

7. In the Checklist, click **Map Outputs**.

This scenario does not sort elements at this moment. So, for now, keep all the returned data:

- Click the response variable to highlight it. Click **Copy**.
- Select the response output parameter, and click **Create** to create an orchestration variable named response and to create the mapping.
- Click **Variables**, and rename the response variable to SAP_CustomerGetDetailsResponse.

8. Remember to save the project before you continue.

Updating the customer record in SAP

Now, all of the required data is received, and you can update SAP using a BAPI call with the Customer2 business object and the ChangeFromData1 method:

- Click the SAP endpoint, and drag it to the canvas to the right of the Get Customer Details From SAP activity but within the If block.
- Select **Invoke BAPI**, as shown in Figure 10-102.

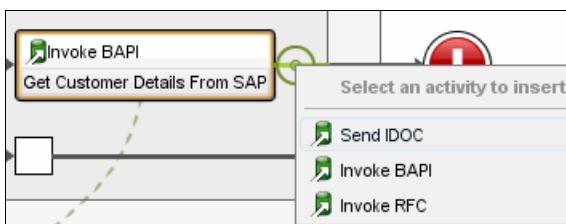


Figure 10-102 Create SAP update activity

- In the Checklist, change the following settings:
- In the Checklist, click **Summary**, and set Activity Name to Update SAP Account.
- In the Checklist, click **Pick Endpoint**, and set the endpoint to SAP, if not already set.
- In the Checklist, click **Configure**.

You must complete the same steps that you previously performed for the Get Customer Details From SAP activity. Although the business object is the same (Customer2), the method to select it this time ChangeFromData1.

The configuration panel looks similar to Figure 10-103.

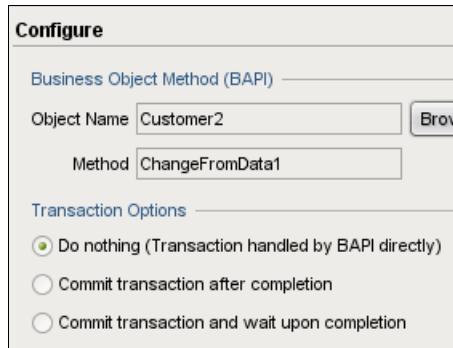


Figure 10-103 Configure Customer2 ChangeFromData

7. In the Checklist, click **Map Inputs**.

Map data from Salesforce.com and from the GetDetail request to the inbound structure. In addition, tell SAP which of the fields that are delivered to use for the update by marking these fields with an *X* in the CompanyDataX input element structure.

Map the following inputs:

- a. Click **Select Inputs**, and select the SAP_CustomerNumber variable.
- b. Create a link from the SAP_CustomerNumber variable to the @CustomerNo element of the variable request.
- c. Click **Select Inputs**, and select the SFDC_OneAccount variable.
- d. Collapse the PersonalData, PersonalDataX, OptionalPersonalData, and OptionalPersonalDataX output sections because you will not map into these blocks.
- e. Create a link from the SFDC_OneAccount variable to the CompanyData element structure of the variable request for the following elements:

Name to	NAME
BillingStreet to	STREET
BillingCity to	CITY
BillingPostalCode to	POSTL_COD1
BillingCountry to	COUNTRY
Phone to	TEL1_NUMBR
Fax to	FAX_NUMBER
SalesOrganisation__c to	SalesOrganisation

Compare the mapping with that shown in Figure 10-104 on page 423.

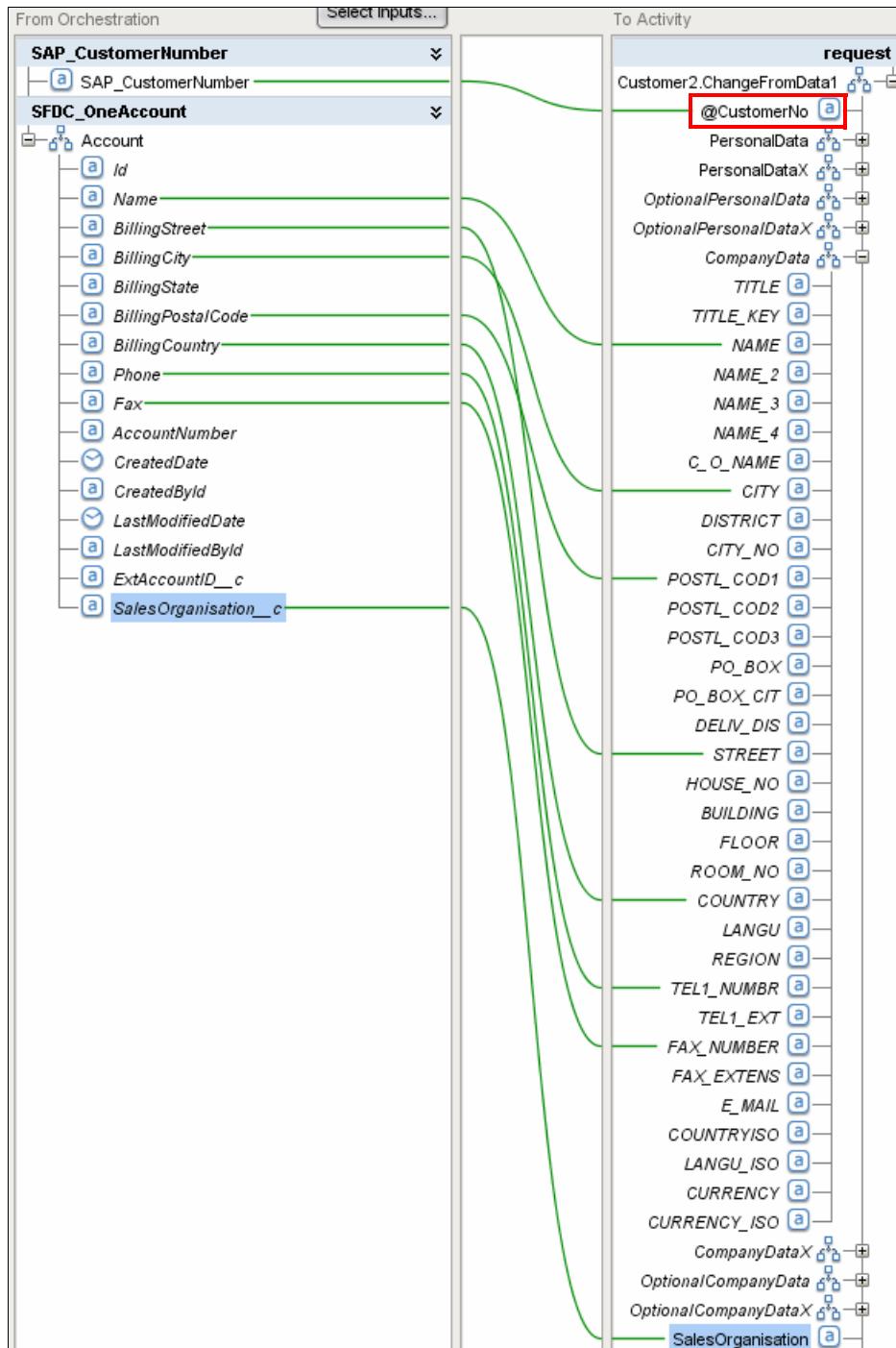


Figure 10-104 Customer2 ChangeFromData Input Map 1

- f. You also need data from the SAP query. Click **Select Inputs**, and select the SAP_CustomerGetDetailsResponse variable.
- g. Create a link from the CustomerAddress element structure of the SAP_CustomerGetDetailsResponse variable to the CompanyData element structure of the request variable for the following elements:

LANGU to	LANGU
LANGU_ISO to	LANGU_ISO

Do not use auto mapping for parts of the SAP structure because you want only these fields mapped.

- h. Create a link from the CustomerGeneralDetail element structure of the SAP_CustomerGetDetailsResponse variable to the CompanyData element structure of the request variable for the following elements:

CURRENCY to	CURRENCY
CURRENCY_ISO to	CURRENCY_ISO

Do not use auto mapping for parts of the SAP structure because you want only these fields mapped.

Tip: If your SAP system does not return a value for the currency, define a default value in the input mapping for the SAP update activity or in the output of the SAP Get Details activity. Otherwise, you will run into the following error during update:

An error occurred while trying to execute activity Execute BAPI. Error is:
com.ibm.j2ca.base.exceptions.InvalidRequestException: BAPI Interface failed to process the request.Error thrown by SAP: Message = Internal error: "The "Currency" field for structure PI_PERSONALDATA is blank, error code: F2887

- i. Now, specify the fields to be updated by defining a default value of *X* for the related fields in the output structure CompanyDataX. Update the following fields:

- NAME
- STREET
- CITY
- POSTL_COD1
- COUNTRY
- TEL1_NUMBR
- FAX_NUMBER

Define, for each field, an individual configuration property to provide more flexibility on which fields are updated.

- j. Compare the mapping for the CompanyDataX with the mapping in Figure 10-105.

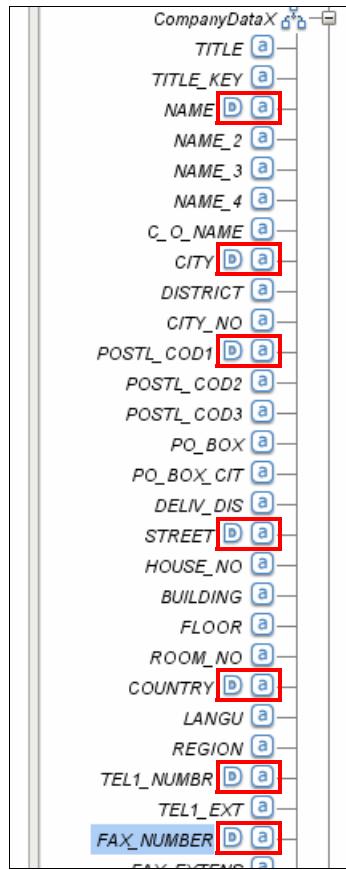


Figure 10-105 Customer2 ChangeFromData Input Map 2

8. In the Checklist, click **Map Outputs**.

At this time, you do not need to sort elements. Keep all of the returned data.

Map the following outputs:

- Click **response** to highlight it. Click **Copy**.
- Select the response output parameter, and click **Create** to create an orchestration variable named *response* and to create the mapping.
- Click the Variables tab, and rename the response variable to SAP_CustomerChangeFromDataResponse.

Remember to save the project before you continue.

Handling invalid Salesforce.com accounts

As shown in Figure 10-106, the Else branch of the validation is empty.

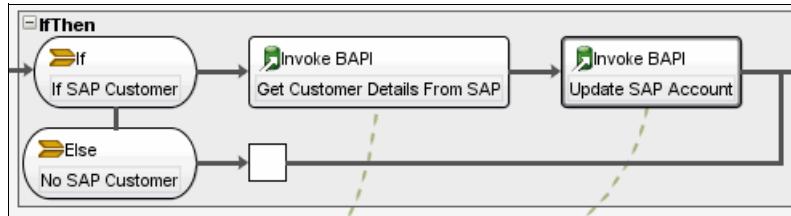


Figure 10-106 Validation Else branch

To fix this issue:

1. Drag a Log Message activity as a replacement for the empty activity onto the Else branch. Name the activity Log Invalid Salesforce.com Account.
2. In the Map Inputs section:
 - a. Set the level to WARNING.
 - b. Select as input the SFDC_OneAccount variable.
 - c. Select the Id, Name, and ExtAccountID__c elements as input and the message element as output. Use the Concatenate function to build a message, such as:
Error during synchronization. Invalid Salesforce.com account: id=xxx, name=yyy, ExtAccountID=zzz.

Handling exceptions and errors

If you now run the orchestration as is without any error handling, any failure with Salesforce.com or SAP stops the orchestration. The orchestration as is does not have a Try-Catch block to handle errors during access to Salesforce.com or SAP.

Create a global catch to avoid any kind of error:

1. Right-click the starting point of the orchestration, and select **Add Catch Branch**, as shown in Figure 10-107.



Figure 10-107 Add global catch

2. Rename the catch to GlobalCatchAll and rename the variables to GlobalFaultName, GlobalFaultData, and GlobalFaultInfo.
3. Add a Log Message activity to the branch, and rename it to Log Global Error. Set the level to ERROR, and set the message as follows:
Critical error during synchronization, name= aaa, message=bbb, id=ccc. activity name = ddd, fault time: eee.
4. Use the GlobalFaultInfo variable and the Concatenate function to build this function.

The global catch block now looks, as shown in Figure 10-108 on page 427.

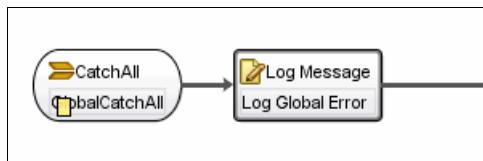


Figure 10-108 Global catch

Although the global catch block is required to have some kind of error handling, this basic configuration is not sufficient.

As an example, perform a test with a simulated SAP failure during SAP update. You can force this failure by removing the currency value from the mapping. Figure 10-109 shows the result. It shows that Salesforce.com returned three updated records. So the loop should have had three iterations. However, because the first one failed during the SAP update, the orchestration went into the global catch block, which then logged the error. Thus, no further updates were done.

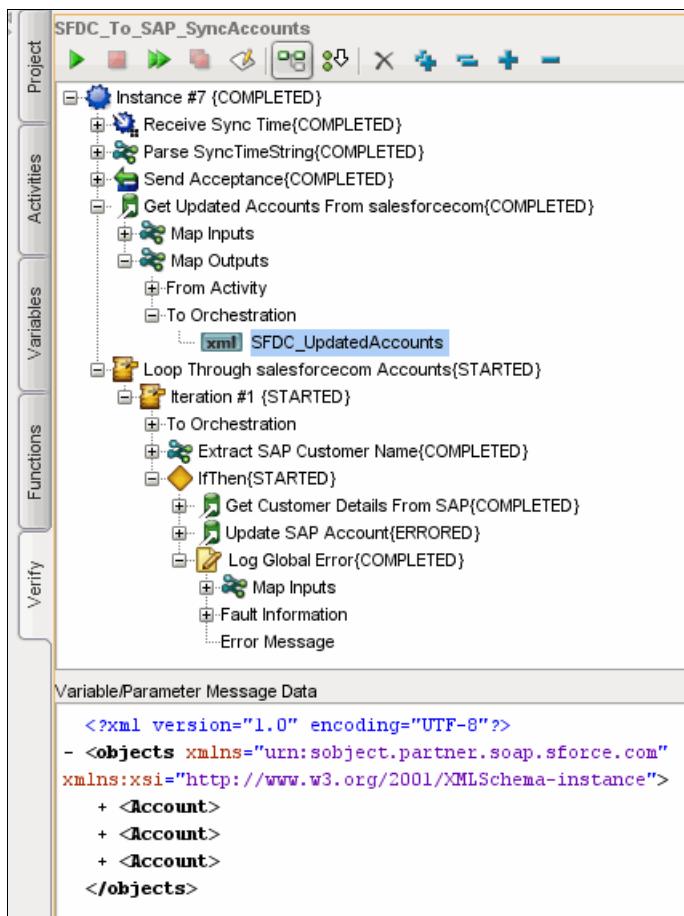


Figure 10-109 Test with failing SAP update but global catch

To avoid this error, put the SAP activities into a Try-Catch block:

1. Drag a Try-Catch activity between the If SAP Customer activity and the Get Customer Details From SAP activity.
2. Rename the Try activity to Try Working With SAP.
3. Move the two SAP activities to the Try-Catch block.

4. Click the catch block, and rename the catch to CatchSAP.
5. Rename the catch variables to SAPFaultName, SAPFaultData, and SAPFaultInfo.
6. Add a Log Message activity to the catch block:
 - a. Copy the Log Message activity from the global catch block to the SAP catch block. Rename it to Log SAP Error.
 - b. Click **Map Inputs**, and edit the function graph.
 - c. Remove the GlobalFaultInfo variable and add the SAPFaultInfo variable. The fields are remapped.
 - d. Double-click the **Concatenate** function, and change the error message to the following text:

Error during synchronization, access to SAP failed. name= aaa, message=bbb, id=ccc. activity name = ddd, fault time: eee.

The Try..Catch block should now look like Figure 10-110.

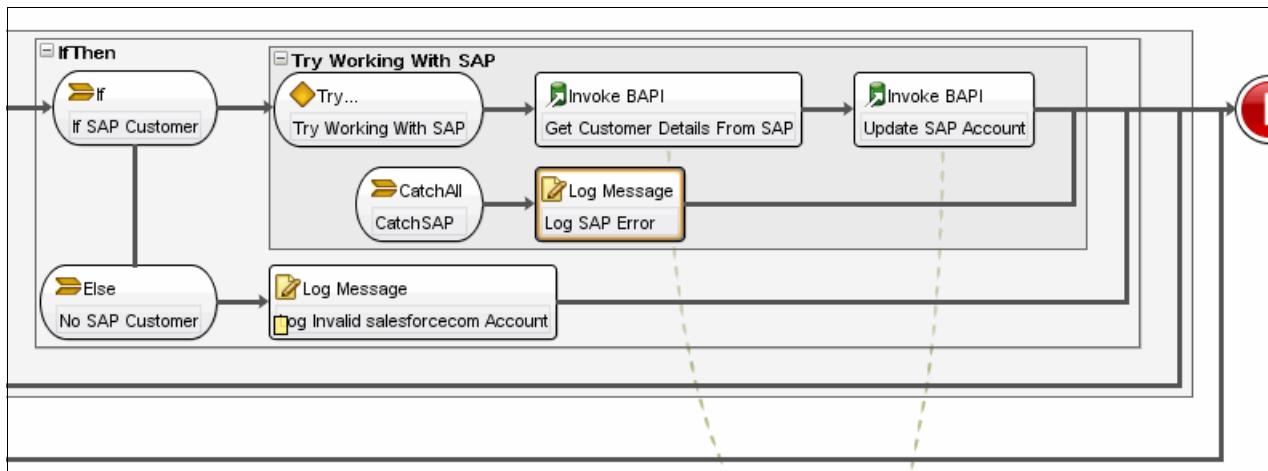


Figure 10-110 Try..Catch block

Running the test again with the failing SAP update, the error is caught by the SAP catch block, and an error message is issued of the following form:

```

Error during synchronization, access to SAP failed. Name=sap.Execute
BAPI.operation, message=An error occurred while trying to execute activity
invokeBAPI. Error is: com.ibm.j2ca.base.exceptions.InvalidRequestException: BAPI
Interface failed to process the request.Error thrown by SAP: Message = Internal
error: "The "Currency" field for structure PI_PERSONALDATA is blank, error code:
F2887, id=126, activity name =Update SAP Account, fault time:
2011-10-12T01:29:16.776+02:00

```

Importantly, the orchestration continues the loop with the next account, as shown in Figure 10-111 on page 429.

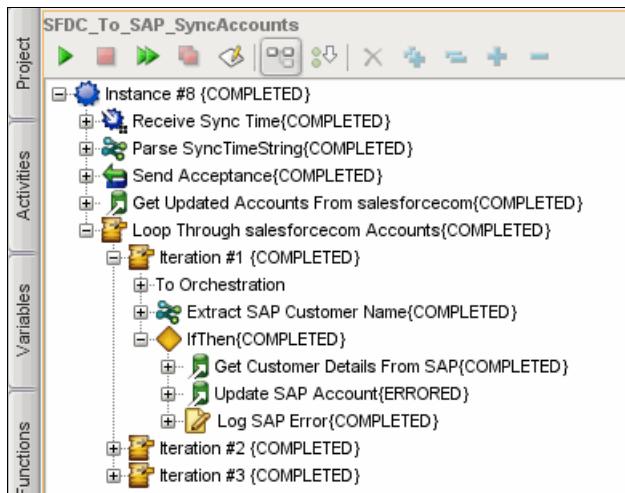


Figure 10-111 Test with failing SAP update but SAP catch

Now, the error handling is acceptable. The last step includes the Common Error Handler. For details about how to add the Common Error Handler, refer to 10.5.7, “Using the Common Error Handler” on page 397. Add the Common Error Handler to every catch block of your orchestration.

10.6.2 Deploying and testing the orchestration on the Integration Appliance

Deploying and testing is similar to the process described in 10.5.9, “Deploying and testing the orchestration on the Integration Appliance” on page 400. The main steps are:

1. Publish the project from Studio directly on the appliance. Alternatively, you can export it as a .par file and upload it from the WMC, as described previously.
2. Go to the WMC, validate the configuration properties, and start the configuration.
3. Within Studio, use the HTTP Post Utility with the Integration Appliance data IP address as the target, and send several requests to the Integration Appliance. After each request, return to the WMC to check the statistics about running, completed, and finished configurations.
4. Go to **Logs** → **System Log** to validate if warnings or errors occurred.

10.6.3 Enhancing the orchestration to avoid feedback loops

In an environment where you have two peer systems, A and B, each system listens on its own input queue. After a message arrives in the input queue, it is sent to the peer into the input queue for that system.

If you put anything into the input queue of system A, system A picks it up and sends it to the input queue of system B. System B picks up the new data and sends it to the input queue of system A. System A picks up the new data and sends it to the input queue of system B. And so on and so on, which is the definition of a *feedback loop*.

How is a feedback loop related to this scenario? Let us assume that there are two active orchestrations:

- ▶ Orchestration A looks for updates in SAP and synchronizes them to Salesforce.com.
- ▶ Orchestration B looks for updates in Salesforce.com and synchronizes them to SAP.

If any data set in SAP or Salesforce.com is updated, a feedback loop occurs. How can you avoid these loops?

Avoiding feedback loops

Many customers avoid feedback loops by having one leading system and allowing only unidirectional synchronization. This discussion provides another approach that allows bidirectional communication. First, consider the problem with the scenario with the two orchestrations that you created in this chapter.

The question then is how can you avoid having a customer record, that was synchronized from SAP to Salesforce.com using the SAP_To_SFDC_SyncAccounts orchestration, picked up by the SFDC_To_SAP_SyncAccounts orchestration and then synchronized back?

You can work with time stamps to compare the last time the record was changed in both systems but this can take much effort. A better approach is to do the synchronization based on the originator of the change. For example, if you know that the last change to a record was done by the SAP_To_SFDC synchronization, you do not need to synchronize that record in the opposite direction. But how do you store the origin of the change?

You can use an extra field in the customer record to store the origin, but then an additional field has to be maintained. With this method, you must make sure that this field is updated when manual changes occur or when changes that are not by that orchestration are done. Otherwise, there is the risk of losing the changes.

Fortunately, many systems already provide a field with the originator of the last change. In Salesforce.com, for example, there is a field called LastModifiedById. This field is maintained automatically. So you just need to identify whether the person who modified the record is a real person or the orchestration to know whether or not the change should be synchronized.

Identifying the integration user ID

You can identify the originator by defining in Salesforce.com an integration user, meaning one user who is used only for the integration from the orchestration and who does nothing else. If the LastModifiedById field contains that integration user ID, the record does not have to be synchronized. Then modify the SFDC_To_SAP_SyncAccounts orchestration to avoid feedback loops by identifying the originator of a change:

1. Run the SFDC_To_SAP_SyncAccounts again in Studio.
2. On the Verify tab, expand the tree to show one loop iteration the Get Updated Accounts Map Output branch until you find the variable SFDC_UpdatedAccounts. Refer to Figure 10-109 on page 427.)
3. Click the **SFDC_UpdatedAccounts** variable to find a record with different IDs for creators or modifiers. Our testing found a record with the following fields:
 - <CreatedById>005A00000011pokIAA</CreatedById>
 - <LastModifiedById>005A0000000TTeJIAW</LastModifiedById>
4. Match the IDs to your Salesforce.com users by looking into Salesforce.com for that record. Again, our testing found the following users:
 - ID 005A00000011pokIAA is the ID of the integration user, which is referred to as IntegrationUserID.
 - ID 005A0000000TTeJIAW is the ID of the Salesforce.com worker, which is referred to as WorkerID.

Options to restrict synchronization to specific changes

To identify the user for updated records, compare the LastModifiedById element of the SFDC_UpdatedAccounts variable with the IntegrationUserID.

You can use the Filter and Profile activity to implement this configuration into the existing orchestration. You then define the filter to return only the updates that are not modified by the IntegrationUserID.

The drawback with this approach is that you first retrieve a bunch of data when you perhaps need only a few records. In the simple scenario that performs an initial load from SAP to Salesforce.com using the orchestration described in 10.4, “Overview of the use cases for this scenario” on page 360, the disadvantage is that you receive a lot of records but none of them require synchronization.

So, you must modify the SFDC_To_SAP_SyncAccounts orchestration to be more intelligent.

Changing the orchestration

To preserve the original orchestration, change the version of the project to version 2.0. Go to **Projects → Project Settings** to change the version to 2.0.

Tip: To keep the source of different versions of the project, either save the versions of the project with distinct names or keep them uploaded on the virtual appliance. The virtual appliance provides the option to download the projects, including source.

Open the SFDC_To_SAP_SyncAccounts orchestration, and look at the Get Updated Accounts From Salesforce.com activity.

The Map Inputs panel shows that there is no option to specify anything more than the input for the startTime variable. To use a more intelligent query that only returns accounts that were not modified by the IntegrationUserID, replace this activity with a more flexible activity. This example uses the Salesforce.com QueryObjects activity.

Follow these steps:

1. Click the Salesforce.com endpoint, and drag it to the canvas to the right of the Get Updated Accounts From Salesforce.com activity.
2. Select **QueryObjects**.
3. In the Checklist, click **Summary**, and set Activity Name to Get Manually Updated Accounts From Salesforce.com.
4. In the Checklist, click **Pick Endpoint**, and set the endpoint to Salesforce.com, if it is not already set.
5. In the Checklist, click **Configure**.

The configuration approach for the Query Objects activity is different from the Get Updated Objects activity. For the Query Objects activity, you must define the query, but for the Get Updated Objects activity, your query is built by inspection of the endpoint and retrieval of the metadata and schemas.

Example 10-1 shows the query for this example.

Example 10-1 Query manually updated accounts from Salesforce.com

```
SELECT Id, AccountNumber, BillingCity, BillingCountry, BillingPostalCode,
BillingState, BillingStreet, ExtAccountID__c, Fax, LastModifiedById,
```

```
LastModifiedDate, CreatedById, CreatedDate, Name, Phone, SalesOrganisation__c
FROM Account
WHERE LastModifiedDate > $SyncTime AND LastModifiedById != $IntegrationUserID
AND ExtAccountId__c != ''
```

This query returns all selected fields if the account was modified after SyncTime, not by the integration user, and if the external ID is not empty.

Enter the query, and then click the check mark to validate the query (as indicated in Figure 10-112).

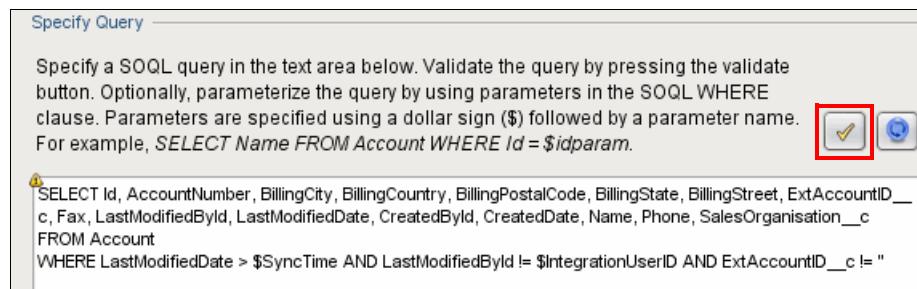


Figure 10-112 Use case SFDC to SAP, configure Query Object

Tips: Some hints for validation:

- ▶ The Validate Query function (triggered by clicking the check mark) first checks the syntax of the query and then connects to Salesforce.com to validate the fields.
- ▶ If validation fails during syntax, ensure that the validation is using the correct quotes.
- ▶ If validation fails during field validation, make sure Studio can connect to Salesforce.com.
- ▶ If the validation still fails, you can get additional hints by clicking **Orchestration → Validate**.

6. In the Checklist, click **Map Inputs**.

The query defines two parameters, \$SyncTime and \$IntegrationUserID. Create the following mappings:

- Click **Select Inputs**, and select **SyncTime**.
- Map SyncTime from the left to SyncTime on the right.

Right-click **IntegrationUserID**, and define the default value 005A00000011pokIAA. Click the bullet to create a configuration property named IntegrationUserID.

7. In the Checklist, click **Map Outputs**.

- a. Click **Select Outputs**, and select **SFDC_UpdatedAccounts**.
- b. Create a link from the objects element of the objects variable to the objects element of the SFDC_UpdatedAccounts variable. This mapping links all the Salesforce.com data fields as shown in Figure 10-113 on page 433.

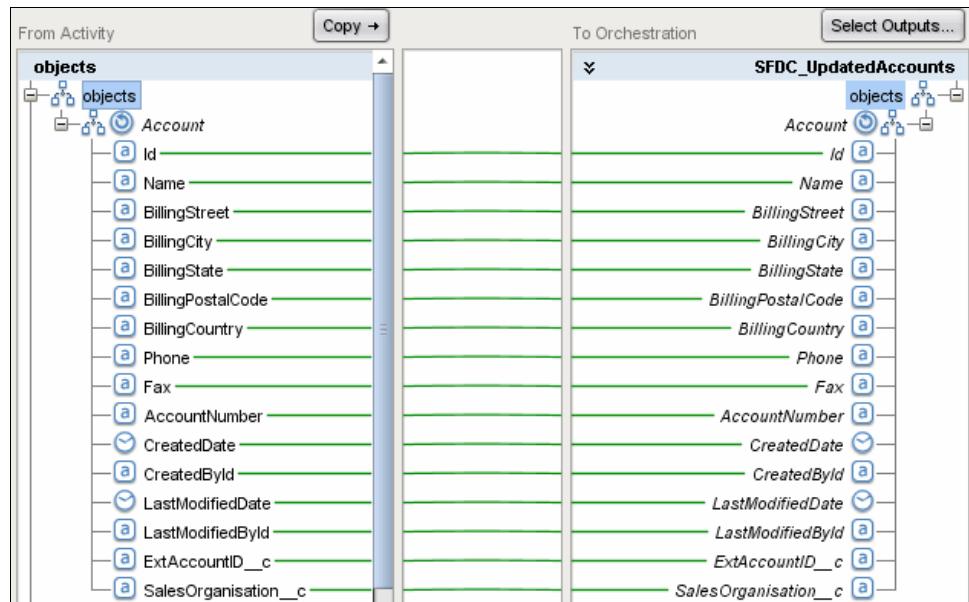


Figure 10-113 Query map outputs

8. Remove the Get Updated Accounts From Salesforce.com activity.

The orchestration now looks as shown in Figure 10-114.

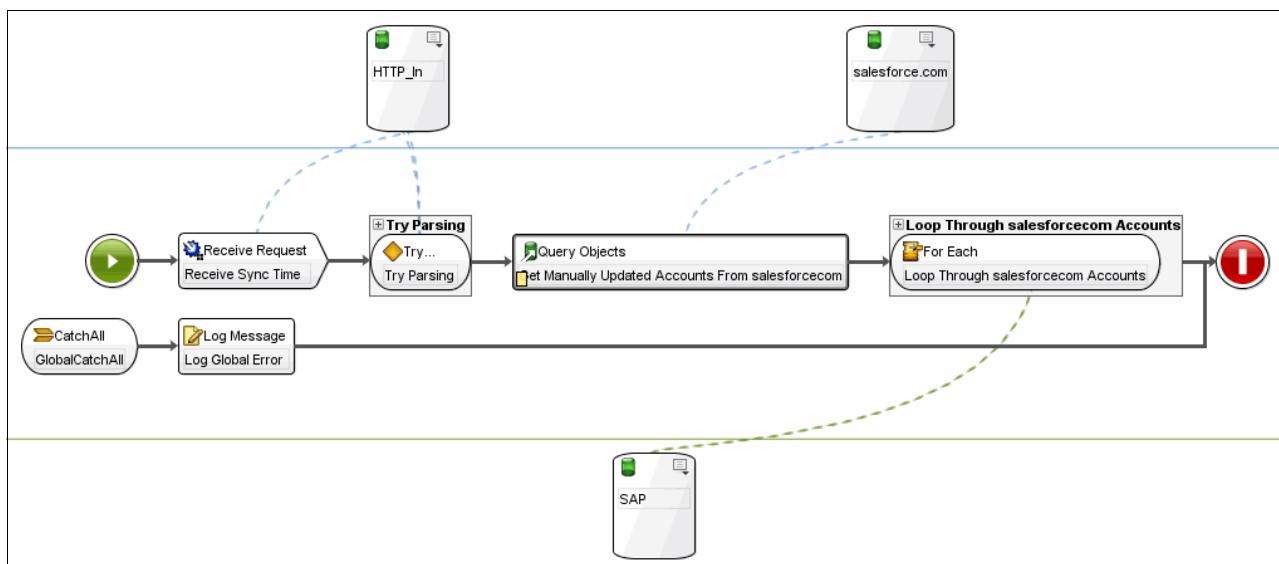


Figure 10-114 Orchestration with query

Testing the new orchestration

Now you can test whether the new orchestration works as designed:

1. Use the SAP_To_SFDC_SyncAccounts orchestration to synchronize some accounts from SAP to Salesforce.com:
 - a. Start the orchestration, if not already started.
 - b. Start the HTTP Post Utility.
 - c. Use the URL http://<yourhost>/SAP_To_SFDC_SyncAccounts.
 - d. In the body, enter the filter criteria for the customer numbers.

2. Verify which accounts were updated.
3. Go to the Salesforce.com portal, and validate that these accounts were modified by the integration user.
4. Use the SFDC_To_SAP_SyncAccounts orchestration with the feedback loop detection (the one with the Query Object activity) to synchronize some accounts from SFDC to Salesforce.com:
 - a. Start the orchestration, if not already started.
 - b. Start the HTTP Post Utility.
 - c. Use the URL http://<yourhost>/SFDC_To_SAP_SyncAccounts.
 - d. In the body, enter the synctime in the format yyyy/MM/dd/hh/mm/ss.
5. View the output within Studio and notice that none of the accounts that were synchronized from SAP to Salesforce.com are synchronized back to SAP, as shown in Figure 10-115.

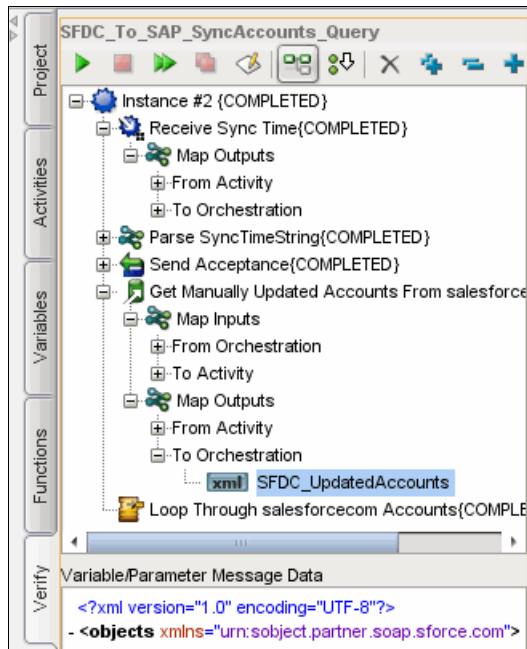


Figure 10-115 Test with query returned no record

6. Log in to the Salesforce.com portal with a user other than the integration user.
7. Change one or more account records.
8. Run the test from step 4 again with the same time stamp. All changed account records are now returned by Salesforce.com and synchronized to SAP.



Scenario: CRM to cloud calendar services

This chapter describes a scenario that uses an integration to synchronize calendar events in a customer relationship management (CRM) system (NetSuite) to Google Calendar. It includes the following topics:

- ▶ WebSphere Cast Iron concepts demonstrated in this scenario
- ▶ Scenario overview
- ▶ Signing in to Cast Iron Live
- ▶ Creating a project
- ▶ Modifying the orchestration
- ▶ Publishing and deploying the project to the development environment

Disclaimer: The intent of this tutorial is to demonstrate the use of WebSphere Cast Iron Live to integrate and utilize third party applications, and does not claim any endorsement or affiliation with the listed products. All product and company names are trademarks or registered trademarks of their respective holders, and IBM disclaims any ownership in such third-party marks. Use of such third-party marks does not imply any affiliation with or endorsement by or for IBM or WebSphere Cast Iron Live. The use of any third party trademarks, logos, or brand names is for informational and instructional purposes only, and does not imply that such trademark owner has authorized IBM to promote its products or services.

11.1 WebSphere Cast Iron concepts demonstrated in this scenario

This chapter demonstrates the use of the following WebSphere Cast Iron concepts:

- ▶ Accessing and using a Template Integration Project(TIP)
- ▶ Using configuration properties
- ▶ Using multiple starter activities
- ▶ Using the email activity
- ▶ Handling errors
- ▶ Using Cast Iron Live to integrate to two cloud services (NetSuite and Google)
- ▶ Initiating an orchestration in Cast Iron Live using an HTTP request

11.2 Scenario overview

Business requirements determine the choice of whether to use the Cast Iron cloud service (WebSphere Cast Iron Live) or an on-premise Integration Appliance. This scenario uses the software as a service (SaaS) applications NetSuite and Google. Because both applications reside in the cloud, we chose to use Cast Iron Live in this scenario to have an “all cloud” scenario.

The TIP that is used in this scenario contains orchestrations to enable connectivity and updates to the Google Calendar. This primary orchestration will be extended to enable the update of your Google Calendar from NetSuite and to show how synchronization of CRM-based data can occur. The scenario uses a shared variable to hold the date and time of the last synchronization job that was run. A variable that has the Shared property set to true persists in the runtime environment between orchestration jobs.

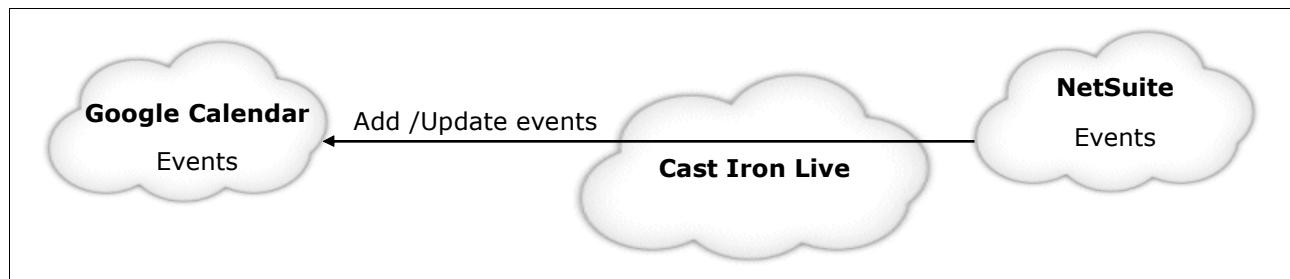


Figure 11-1 Use case scenario

11.2.1 Prerequisites

To follow this scenario, you need the following accounts or logins:

- ▶ A NetSuite account
- ▶ A Google account
- ▶ A Cast Iron Live login

If you do not have a Cast Iron Live account, the scenario can be developed and verified in Studio installed on your PC. The Studio steps are the same as detailed in this chapter, but you cannot initiate Studio from Cast Iron Live or publish the project to a Cast Iron Live environment. To create the project, select the Studio menu **Solutions → Search for TIPs**, search for Google Calendar. Select **NetSuite _Google_syncCalendar_CastIronLive**, and create the project.

11.2.2 Scenario description

This scenario illustrates an example of using a TIP that provides the required orchestrations for the example integration. The TIP contains the basic orchestration for synchronizing a NetSuite calendar to a Google Calendar. The means of using the Google API for logging in to a Google account and creating and updating the Google Calendar are provided as separate orchestrations in the project.

The primary orchestration completes the following steps, as illustrated in Figure 11-2:

- ▶ Start is initiated by a schedule activity.
- ▶ Use a shared variable to record the last job date it run.
- ▶ Find NetSuite events that were created or updated after this date.
- ▶ Update the last job date variable with the date and time after finding the NetSuite events.
- ▶ Log in to Google.
- ▶ Loop through each NetSuite event and complete one of the following actions:
 - Update the existing Google event.
 - Create a new event in Google if the event does not exist.
- ▶ Store the Google event ID in NetSuite.

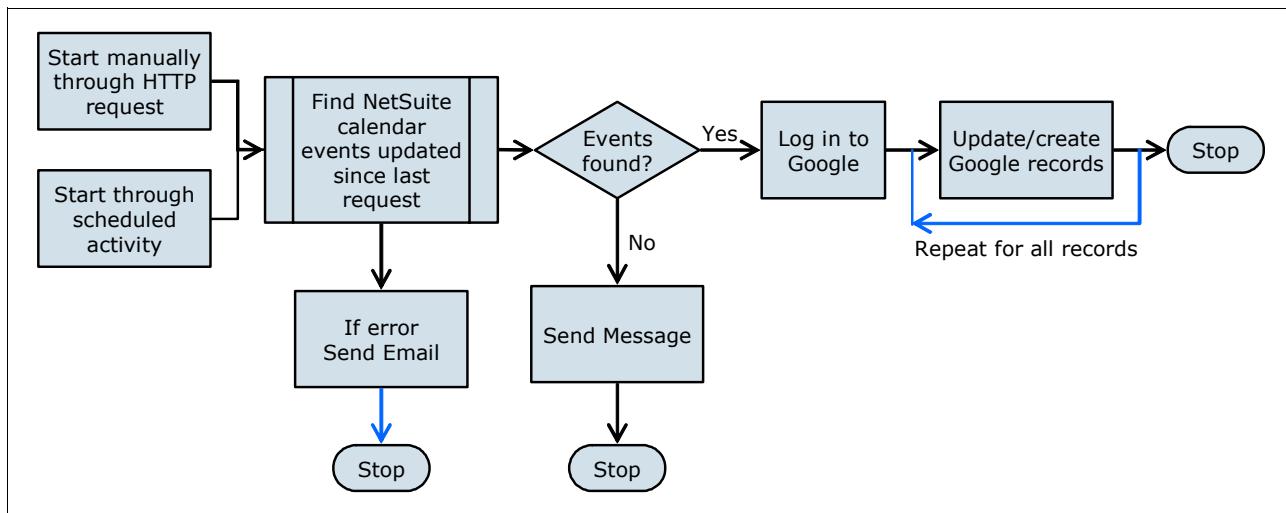


Figure 11-2 Orchestration workflow

This scenario modifies the primary orchestration as follows:

- ▶ Invoke the orchestration either for a scheduled activity or using HTTP Request
- ▶ Send an email to the Google account if an error occurs

11.3 Signing in to Cast Iron Live

To sign in to Cast Iron Live:

1. In a web browser go to the Cast Iron Live website:
<https://cloud2.castiron.com/login>
2. Enter your user name and password, and click **Log In**. The Cast Iron Home tab displays, as shown in Figure 11-3 on page 438.

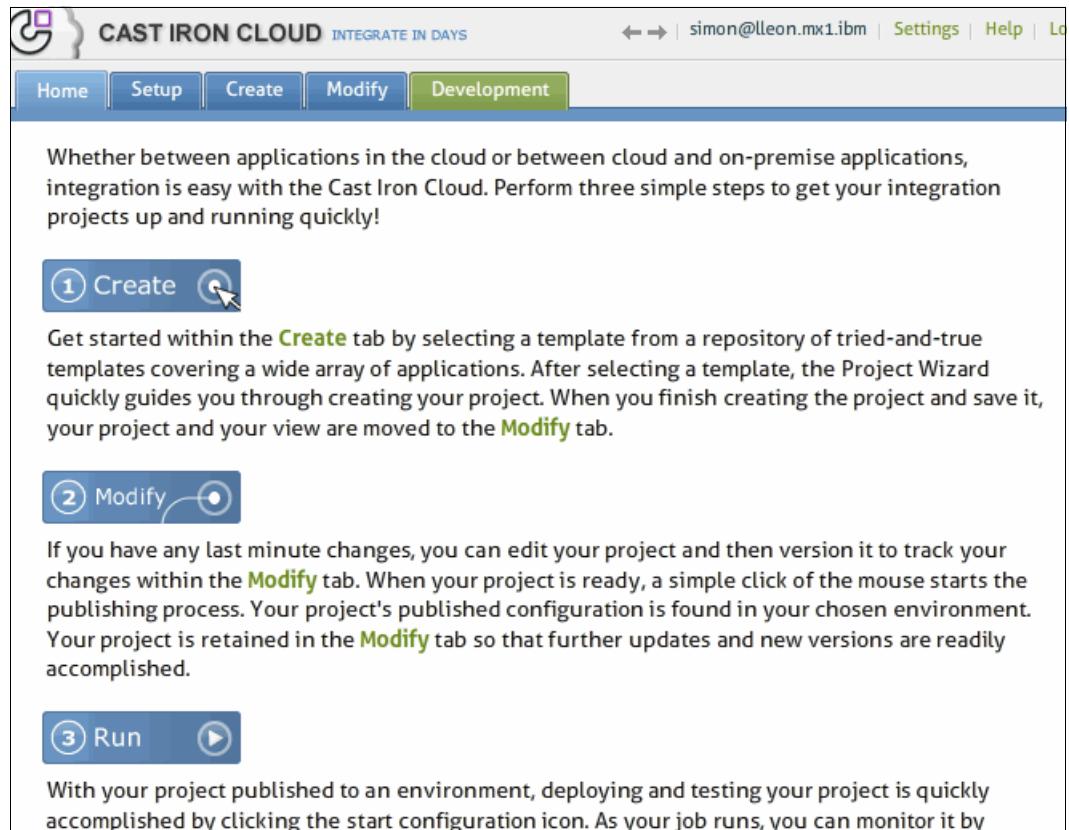


Figure 11-3 Cast Iron Live Home tab

Tip: If your session in Cast Iron Live times out, you cannot save the project in your Studio session to Cast Iron Live. Either set the Cast Iron Live timeout to a long time (or never) or save the project to your local disk.

11.4 Creating a project

You can download a Template Integration Project (TIP) from the Cast Iron community. A TIP provides a fast way of getting started. After you download the TIP, you can modify the orchestrations that it contains. The TIP for this scenario connects to NetSuite and then writes the events found to the Google Calendar. This scenario extends that orchestration.

To use the TIP:

1. Go to the Create tab, and click **Target Endpoint**, as shown in Figure 11-4 on page 439.

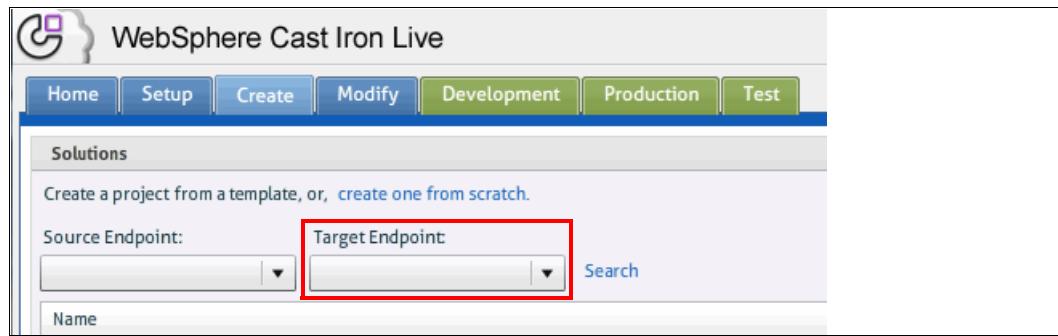


Figure 11-4 Cast Iron Live Create tab

- From the Target Endpoint drop-down menu, select **Google Calendar**, as shown in Figure 11-5.

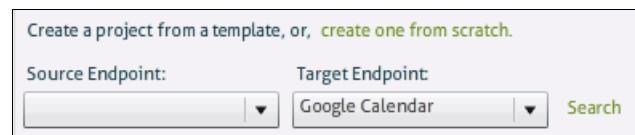


Figure 11-5 Create project selections in Cast Iron Live

- Click **Search**. The list of TIPs with Google Calendar as the **Target Endpoint** displays, as shown in Figure 11-6. This list can vary.

Solutions			
Create a project from a template, or, create one from scratch .			
Source Endpoint:	Target Endpoint:	Search	
	Google Calendar		(3 matches)
Name	Source	Target	
NetSuite_Google_syncCalendar_CastIronLive	NetSuite Calendar	Google Calendar	
T0199-GoogleSalesforceCalSyncUsing_REST	Salesforce Rest API	Google Calendar	

Figure 11-6 List of TIPs using Google Calendar as the target endpoint

- Click **NetSuite_Google_syncCalendar_CastIronLive**. Template details display in the bottom pane, as shown in Figure 11-7.

Template Details	
Description: This TIP demonstrates the real-time synchronization of NetSuite Calendar Events with Google Calendar Events using Google REST APIs. This tip uses polls data from NetSuite through a scheduled mechanism to sync it with Google Calendar on the WebSphere Cast Iron Live form factor!	
Project Name:	NetSuite_Google_syncCalendarCLOUD_WZ
Creator:	lleon@mx1.ibm.com
Category:	USECASES
Keywords:	NetSuite,Google,Calendar,Event,Events
Create a project based on this template.	
Created:	04/29/2010 01:39:54 AM
Last Modified:	10/21/2011 11:58:35 PM
Downloads:	0
Rating:	(1.0/5.0) Read reviews

Figure 11-7 Template details for the selected TIP

- Click **Create a project based on this template**. A new browser window opens that shows an Initializing status bar. When complete, the TIP Configuration Wizard opens in the new browser window and displays the Introduction step, as shown in Figure 11-8. Click **Next**.

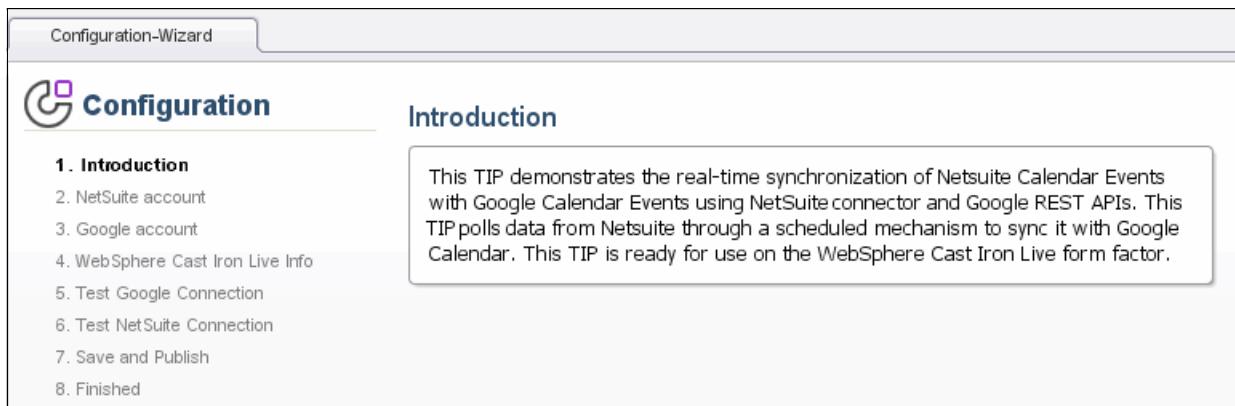


Figure 11-8 TIP configuration wizard step 1: Introduction window for the TIP configuration

- In step 2, enter the NetSuite account details, as shown in Figure 11-9, and then click **Next**.

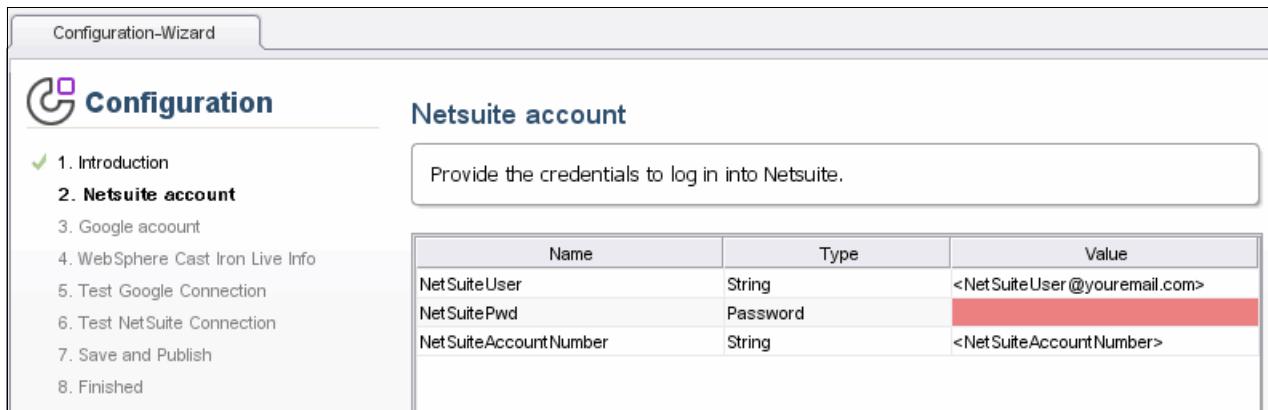


Figure 11-9 TIP configuration wizard step 2: NetSuite account details

- In step 3, enter the Google account details, as shown in Figure 11-10 on page 441, and then click **Next**.

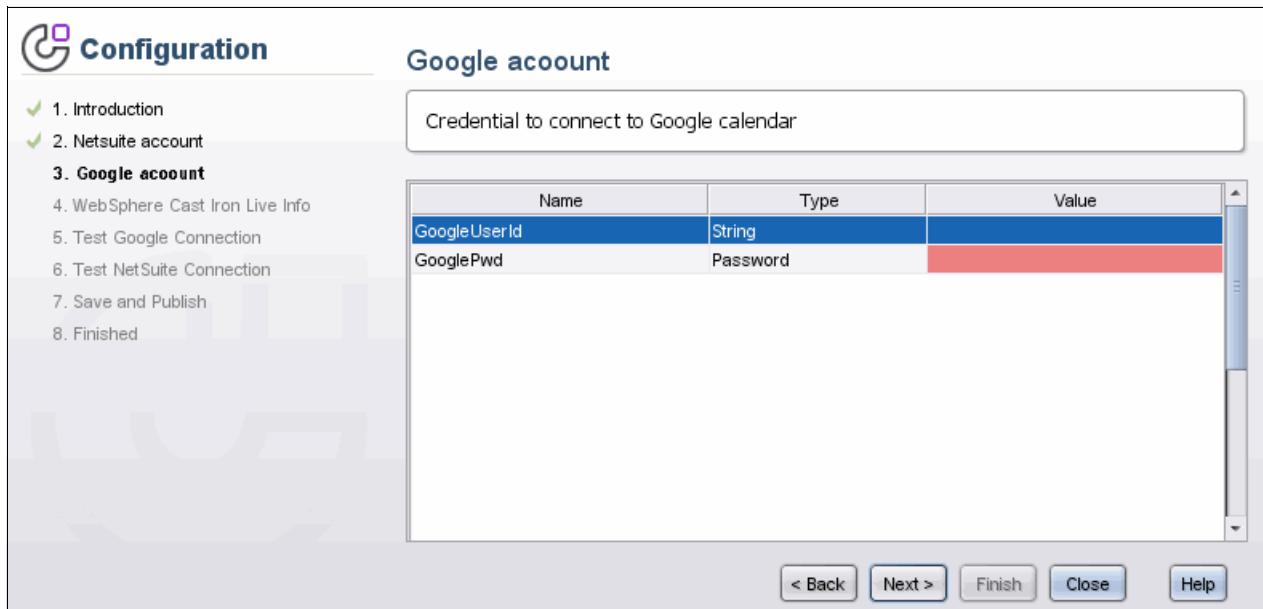


Figure 11-10 TIP configuration wizard step 3: Google account details

8. In step 4, enter the Cast Iron Live details, as shown in Figure 11-11, and then click **Next**.

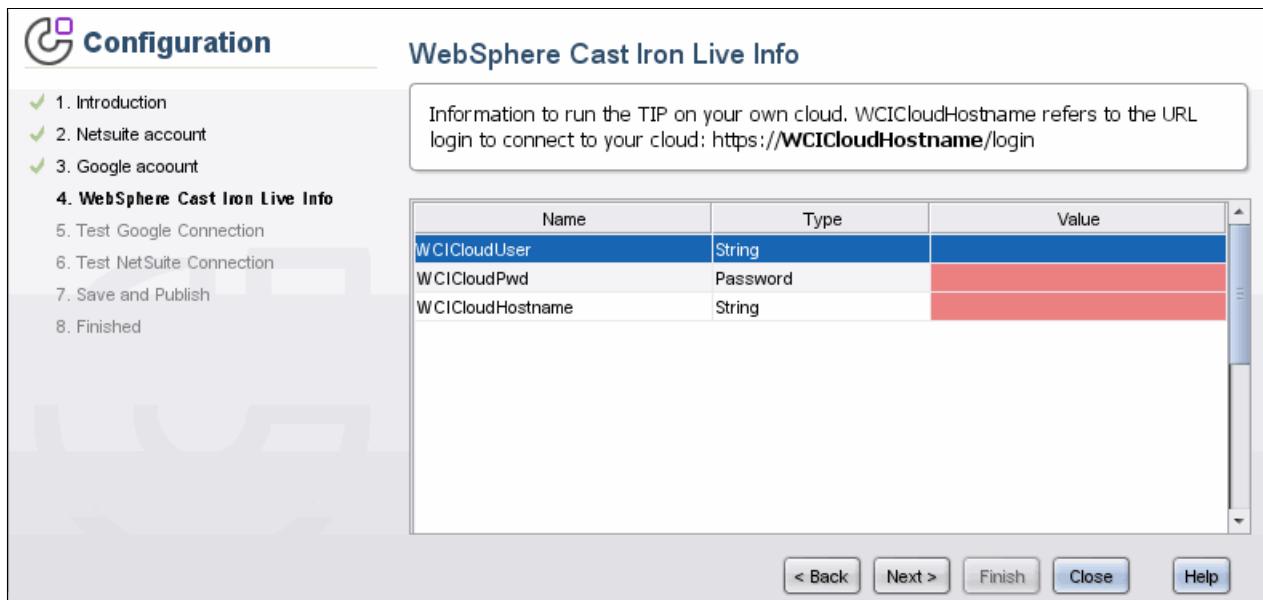


Figure 11-11 TIP configuration wizard step 4: Cast Iron Live details

9. In step 5, click **Test Connection** to test the Google connection, as shown in Figure 11-12 on page 442. A dialog box opens to confirm a successful connection. If the connection is successful, close the dialog box, and then click **Next**.

 Configuration

1. Introduction
2. NetSuite account
3. Google account
4. WebSphere Cast Iron Live Info
5. Test Google Connection
6. Test NetSuite Connection
7. Save and Publish
8. Finished

Test Google Connection

Click "Test Connection" button to test connection to Google.com

Integration Appliance Receives Request
 Configure for WebSphere Cast Iron Live

	Port	Path
https://	Integration Appliance	: 80 <input type="button" value="T"/> / Setup in Activity

Remote Server
 Invoke service in WebSphere Cast Iron Live

Host Name	Port	Path
https:// <input style="background-color: #e0f2e0; border: 1px solid #ccc; width: 100px; height: 20px; border-radius: 5px;" type="text" value="GoogleHostName"/> 	: 443 <input type="button" value="T"/> 	/ Setup in Activity

Login

Log into the Server as an Anonymous User
 Log into the Server with User Name and Password

Authentication	Basic <input type="button" value="▼"/>	Realm <input style="width: 100px; height: 20px; border: 1px solid #ccc; border-radius: 5px;" type="text"/> <input type="button" value="T"/>
User Name	<input style="width: 200px; height: 20px; border: 1px solid #ccc; border-radius: 5px;" type="text"/> <input type="button" value="T"/>	
Password	<input style="width: 200px; height: 20px; border: 1px solid #ccc; border-radius: 5px;" type="text"/> <input type="button" value="T"/>	

Security

None
 HTTPS Client Certificate Alias Name

Connection Pool Options

Maximum Connections  

Connection Timeout

Time out after   second(s) when establishing a connection to the Endpoint.

Proxy

Connect via a Proxy Server

Authentication	Basic <input type="button" value="▼"/>	Realm <input style="width: 100px; height: 20px; border: 1px solid #ccc; border-radius: 5px;" type="text"/> <input type="button" value="T"/>
Host Name	<input style="width: 200px; height: 20px; border: 1px solid #ccc; border-radius: 5px;" type="text"/> <input type="button" value="T"/>	
Port	<input style="width: 100px; height: 20px; border: 1px solid #ccc; border-radius: 5px;" type="text"/> <input type="button" value="T"/>	
User Name	<input style="width: 200px; height: 20px; border: 1px solid #ccc; border-radius: 5px;" type="text"/> <input type="button" value="T"/>	
Password	<input style="width: 200px; height: 20px; border: 1px solid #ccc; border-radius: 5px;" type="text"/> <input type="button" value="T"/>	

Remote Endpoint Configuration

Figure 11-12 TIP configuration wizard step 5: Testing the Google connection

10. In step 6, click **Test Connection** to test the NetSuite connection. A dialog box opens to confirm a successful connection. If the connection is *unsuccessful*, return to step 2 by clicking the Back button, and then enter the NetSuite account details again, as shown in Figure 11-9 on page 440. If the connection is successful, close the dialog box, and then click **Next**.

Configuration

Test NetSuite Connection

Click on "Test Connection" button to test connection to Netsuite site

NetSuite Customer Login

- Email Address: NetSuiteUser
- Password: NetSuitePwd
- Account Number: NetSuiteAccountNumber

Login Options

- Login normally
- Login to specified URL

Login URL: https://webservices.netsuite.com/services/NetSuitePort_2009_1
Version: _2009_1

Web Services Plus License

Connection Pool Options

- Minimum Connections: 1
- Maximum Connections: 1
- Maximum Idle Time: 120 minute(s)
- Maximum Wait: 300 second(s)

Time out after 300 second(s) when establishing a connection to the Endpoint.

Proxy

Connect via a Proxy Server

Authentication: Basic
Realm: [empty]

Host Name: [empty]
Port: [empty]
User Name: [empty]
Password: [empty]

Update

Update wsdl to _2011_1

Test Connection

Figure 11-13 TIP configuration wizard step 6: Testing the NetSuite Connection

11. In step 7, click **Finish**.

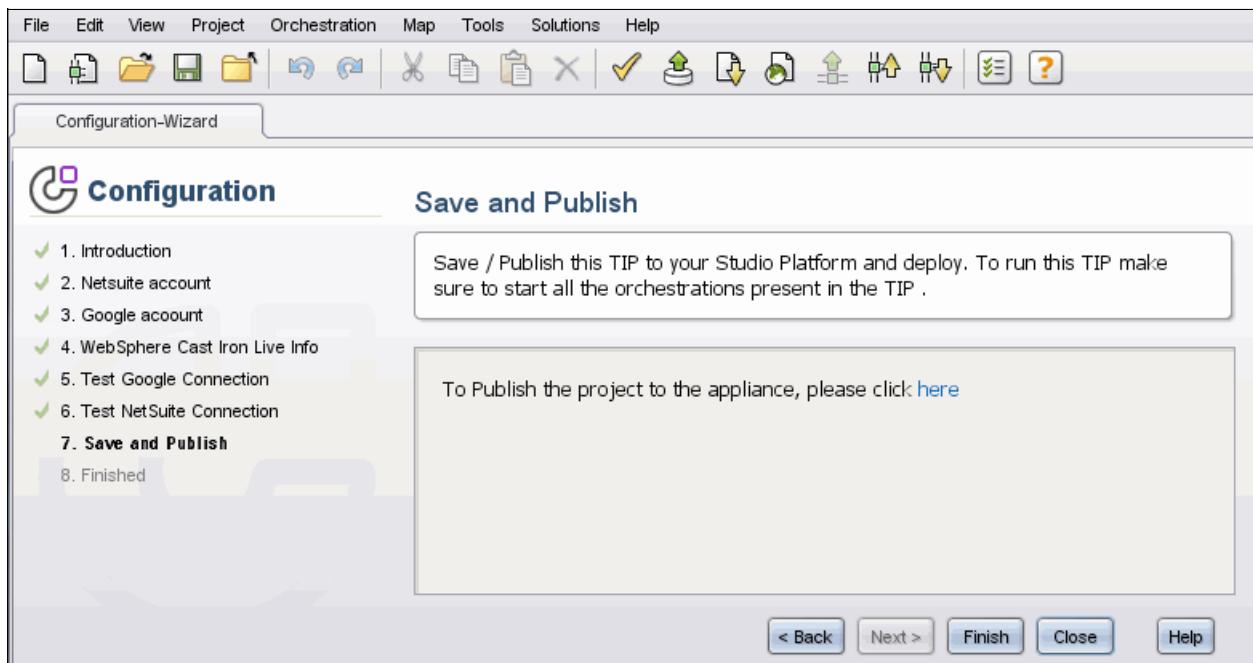


Figure 11-14 TIP configuration wizard step 7: Save and publish

12. In step 8, provide a review of the TIP at this point. Click **Close**.



Figure 11-15 TIP configuration wizard step 8: Finished

You have now completed the configuration steps for the project and have a completed set of orchestrations. When using a TIP, if you do not need to make any modifications, you can publish the project to the Cast Iron runtime. Having completed the TIP configuration, you now have access to all orchestrations in Studio and can develop these. In this scenario, you modify the primary orchestration.

11.5 Modifying the orchestration

The TIP, shown in Figure 11-16, provides most of the requirements of this scenario. In addition to starting the orchestration jobs on a schedule, this scenario adds an HTTP Receive Request activity so that the orchestration can be invoked manually. In addition, this scenario adds an error handler to the NetSuite Search Records activity that sends an email to the Google account if the activity fails.

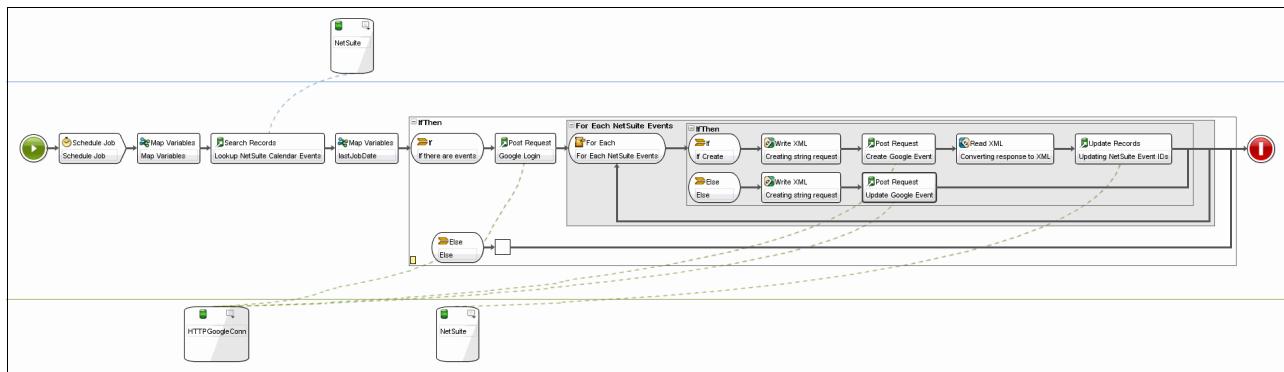


Figure 11-16 *NetSuite_Google_syncCalendar* orchestration

11.5.1 Add the HTTP Receive Request starter activity to the orchestration

To add the HTTP Receive Request starter activity in addition to the Schedule start, use the Pick activity. This activity enables more than one starter activity to be included in the orchestration. Follow these steps to add the activity:

1. The project has multiple orchestrations. Ensure that you have the primary orchestration, **NetSuite_Google_syncCalendar**, open in the workspace, as shown in Figure 11-17. If the primary orchestration is not open, go to the Project tab on the right, and then under Orchestrations, double-click the **NetSuite_Google_syncCalendar** orchestration to open the orchestration to a tab on the workspace.

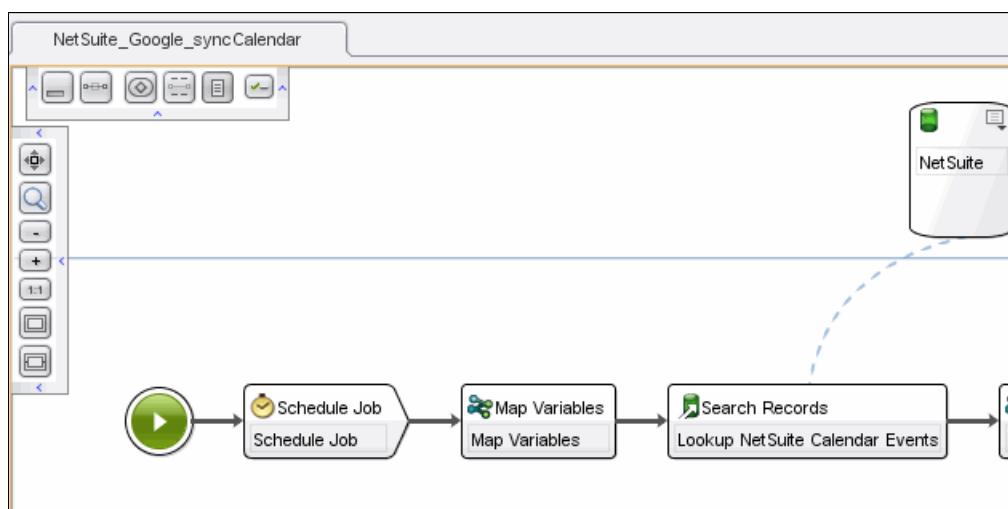


Figure 11-17 *NetSuite_Google_syncCalendar* orchestration open in the Studio workspace

2. Go to the Activities tab, and expand the **Logic** section, as shown in Figure 11-18.

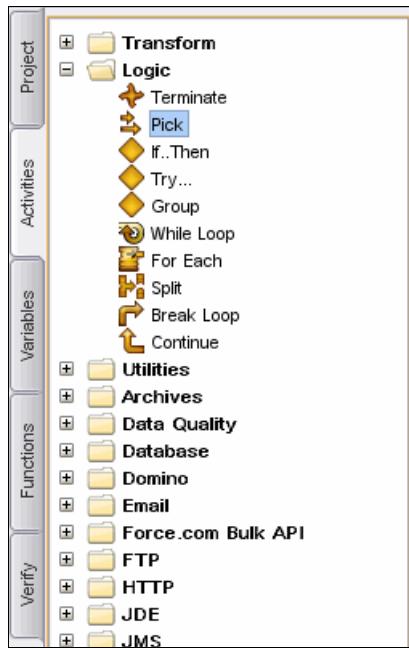


Figure 11-18 The Pick activity in the Logic section of the Activities tab

3. Drag the Pick activity to the empty orchestration as the first activity in the orchestration in front of the Schedule Job activity, as shown in Figure 11-19. A Pick group is created and shows two empty starter activities. The Pick group enables more than one starter activity to be used in the orchestration. If you want to have more than two starter activities, right-click the Pick item, and use the Add Receive Branch menu item.

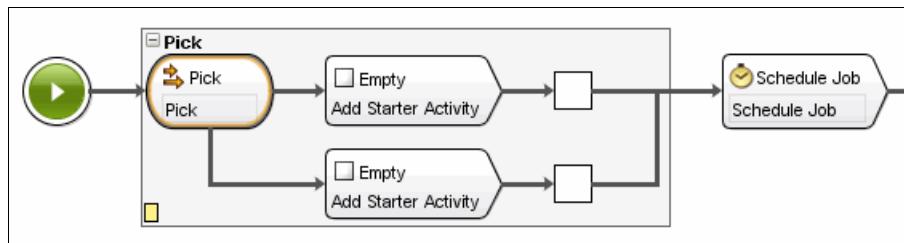


Figure 11-19 Orchestration showing the Pick activity to allow multiple starter activities

4. Drag the Schedule Job activity in the orchestration to the uppermost Add Starter Activity of the Pick group, as shown in Figure 11-20. When the prompt displays, click Yes in the Replace Activity dialog.

Note: Rather than drag, you can use cut and paste for moving activities in the orchestration. Cut and paste is available through standard shortcut keys, the right-click context menu, and through the main menu.

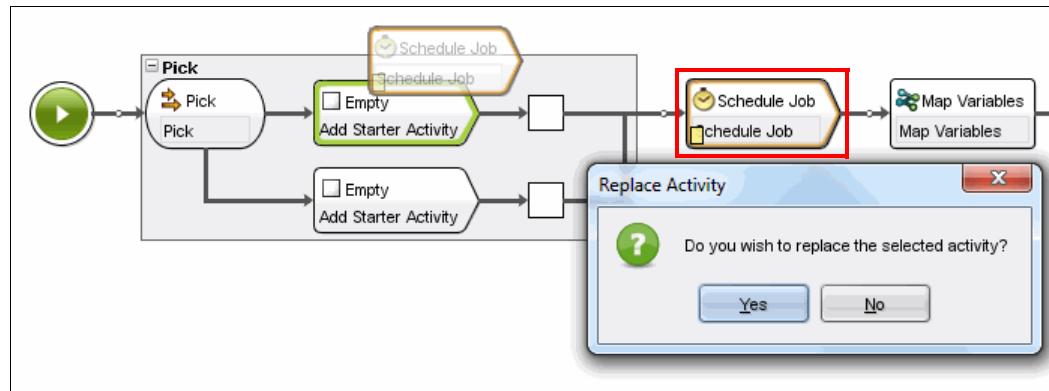


Figure 11-20 Moving the Schedule Job activity into the Pick group

5. Open the HTTP section on the Activities tab, and drag the Receive Request activity to the remaining Add Starter Activity of the Pick group. Figure 11-21 shows the Checklist for the HTTP Receive Request activity.

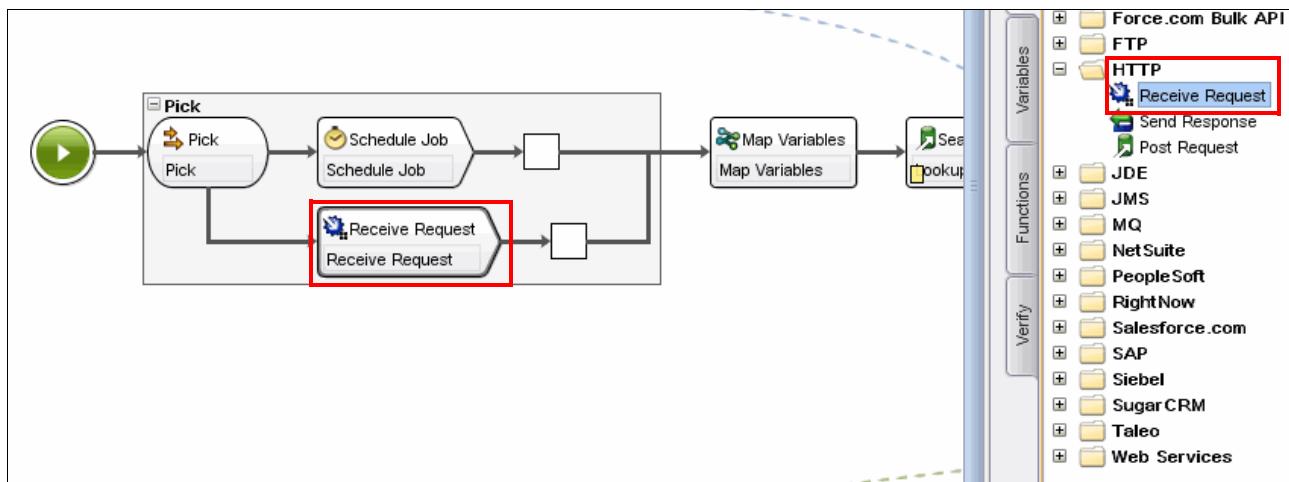


Figure 11-21 Pick group with HTTP Receive Request as a starter activity with Schedule Job

6. Click **Pick Endpoint** in the checklist for the HTTP Receive Request activity.

7. Click **Browse**, and select **HTTP Listener**. Refer to Figure 11-22.

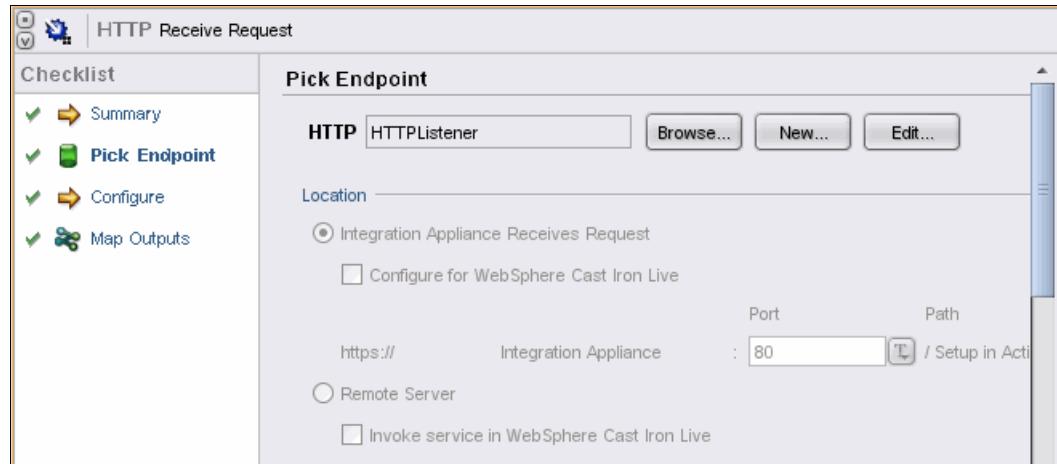


Figure 11-22 Endpoint selection for HTTP Receive Request

8. Click **Configure** in the Checklist. Enter `/syncNetSuiteGoogleCalendar` in the text box for URL (Path After Hostname). Select the **Requires a Reply** option. Refer to Figure 11-23.

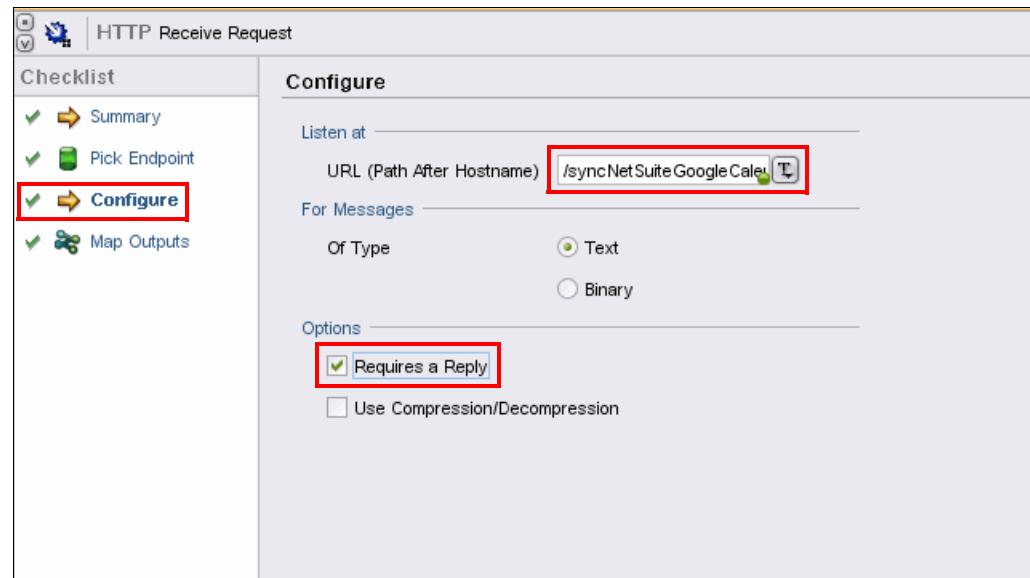


Figure 11-23 Configure details for HTTP Receive Request

11.5.2 Performing an initial test on the NetSuite activity

You can test the project in Studio, either in whole or for specific activities. The following steps test only on the NetSuite Search Records activity to verify that it can retrieve information from NetSuite.

To perform an initial test:

1. Right-click the **Search Records** activity (Lookup NetSuite Calendar Events), and then select **Verify Activity**, as shown in Figure 11-24. A dialog box opens starting the orchestration.

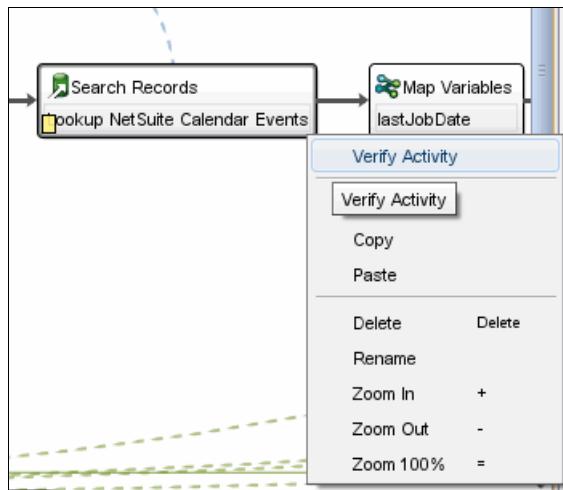


Figure 11-24 Verify Activity menu option

2. When the dialog box closes, go to the Verify tab, as shown in Figure 11-25.

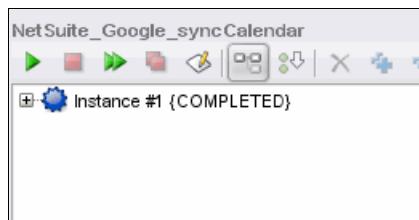


Figure 11-25 Verify tab showing NetSuite Search Records activity

3. Click **Instance #1 (COMPLETED)**, and then click the **Expand All** icon (⊕).

Instance #1 shows the details for the job of the NetSuite Search Records activity only. The runtime engine in Studio shows all messages in to and out of each activity.

Refer to Figure 11-26.

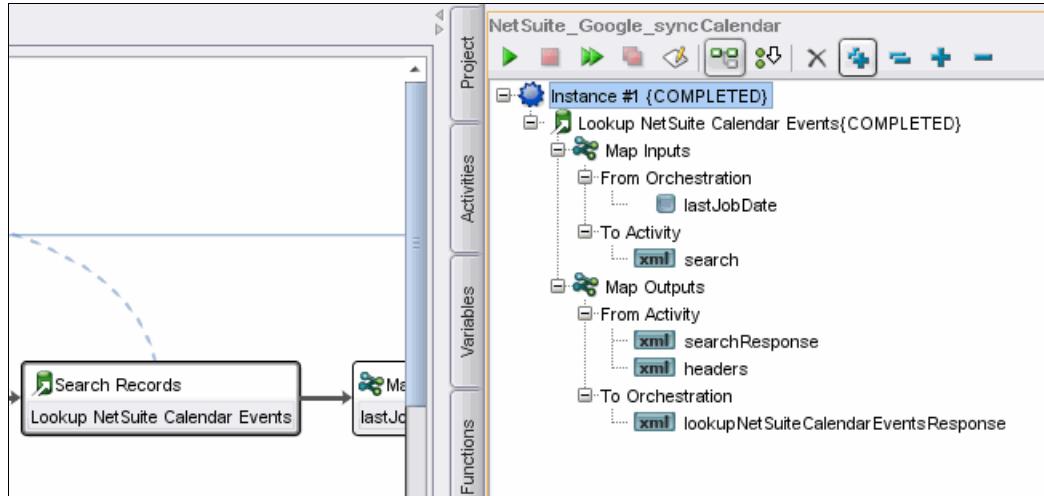


Figure 11-26 Verify tab showing the expanded NetSuite Search Records activity

- Click the last entry, **lookupNetSuiteCalendarEventsResponse**. In the Variable/Parameter Message Data pane at the bottom, you can now see the data retrieved from NetSuite as an XML document. Refer to Figure 11-27.

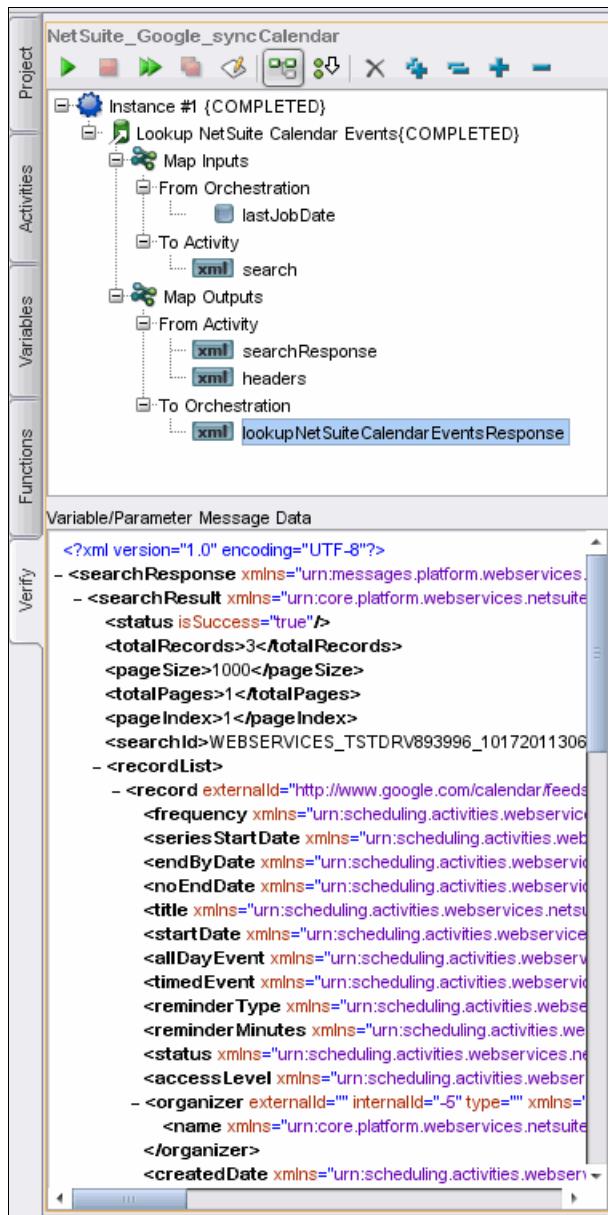


Figure 11-27 Studio Verify showing data for a variable

11.5.3 Adding a response to the HTTP Receive Request

If the orchestration is started by an HTTP Receive Request, this scenario needs to send a response. (This is a setting on the HTTP Receive Request activity.) To avoid an error in the orchestration when it is started through a schedule job, a flag must be set to identify which starter activity was used.

To add a response:

1. In the Variables tab, right-click, and then select **Create New Variable**:
 - a. Select the primitive type **Boolean**, and click **Next**.

- b. Name the variable **HTTPStarterActivity**, and click **Finish**.
2. Click the **Schedule Job** activity.
 - a. Click **Map Outputs** in the Checklist.
 - b. Click **Select Outputs**, and select the **HTTPStarterActivity** variable. Click **OK**.
 - c. Right-click the **HTTPStarterActivity** in the To Orchestration area, and select **Define Default Value**. Set a default value of False for this variable, and click **OK**, as shown in Figure 11-28.

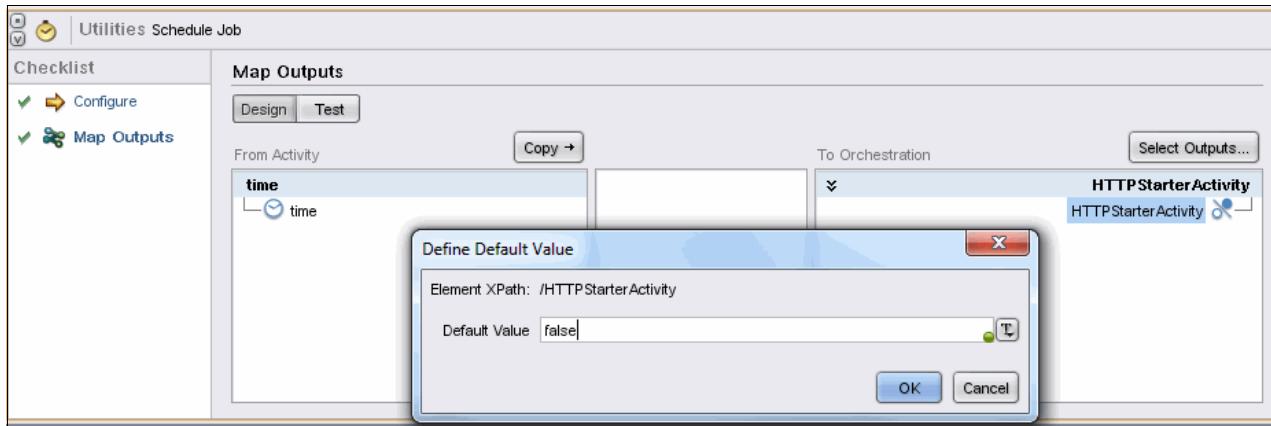


Figure 11-28 Setting a variable value in Map Outputs for Schedule Job activity

3. Click the activity **Receive Request**, and then select **Map Outputs**.
 - a. Click **Select Outputs**.
 - b. Select the **HTTPStarterActivity** variable.
 - c. Set its default value to True, as shown in Figure 11-29. Click **OK**.

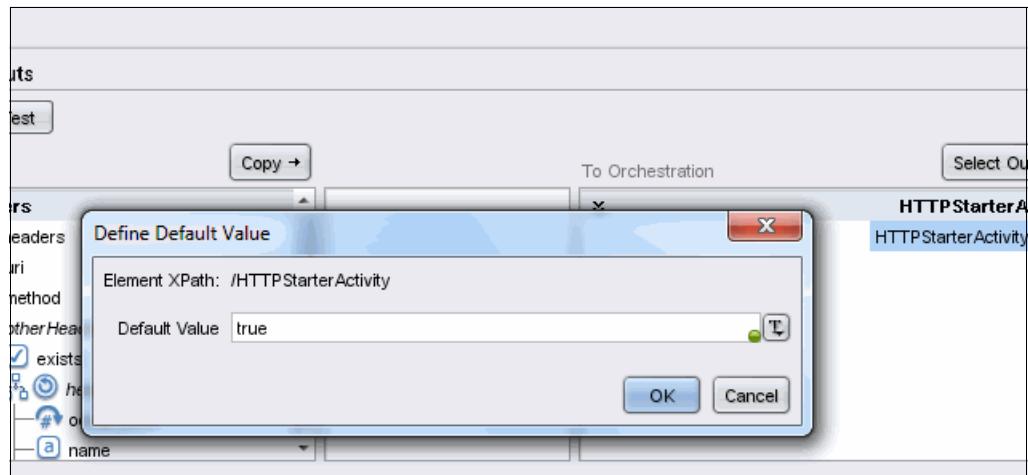


Figure 11-29 Setting a variable value in Map Outputs for HTTP Receive Request activity

Now the variable is set, and a response can be sent to the HTTP request.

4. Drag an If...Then activity to the orchestration within the existing If...Then activity and outside of the For Each NetSuite Events activity, as shown in Figure 11-30.

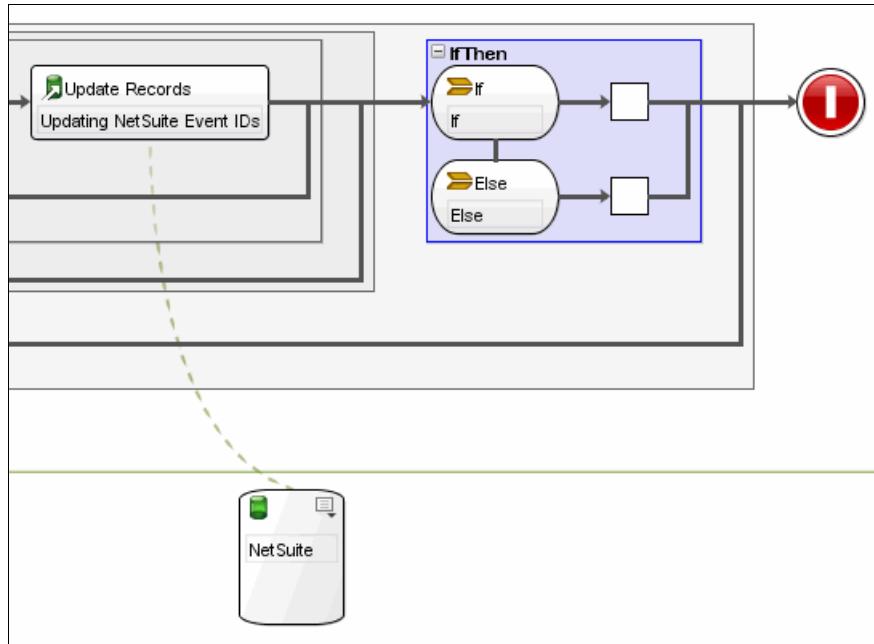


Figure 11-30 If...Then block added to the end of the orchestration

- Rename the If item. Right-click, and then select **Rename**. Enter the following name:
If HTTP Start
- Click the Left Hand Expression text box, and then click the Define New Variable Expression icon (...).
- Select **HTTPStarterActivity** in the drop-down list shown in Figure 11-31. Select the variable **HTTPStarterActivity**, and click **OK**.

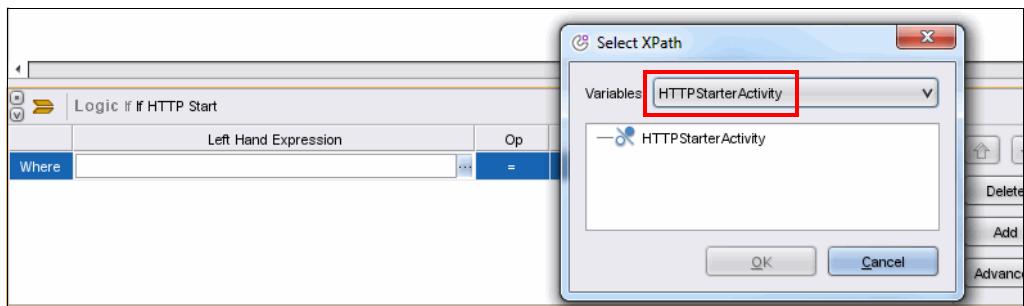


Figure 11-31 Setting of Left Hand Expression for If activity

- Set the Op value to “=” (equals).
- Click the Right Hand Expression text box, and enter `True()`.
- Drag an **HTTP Send Response** activity to the If...Then block after the If item.
- In the Checklist, select **Map Inputs**, and set the default value for the body as follows:
Events have been synchronized

Figure 11-32 on page 454 shows the completed If...Then block.

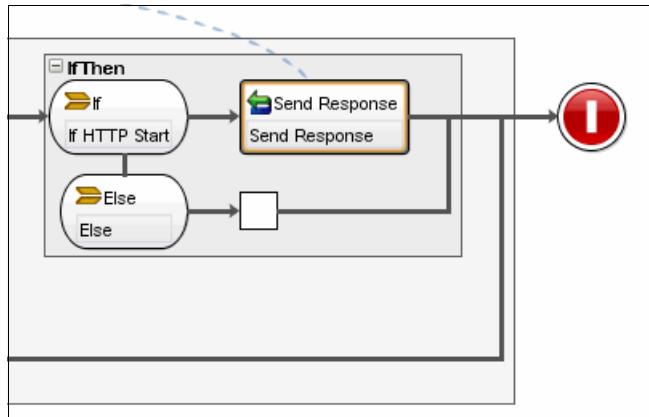


Figure 11-32 Completed IfThen block to send an HTTP response

11.5.4 Adding a response if no events are to be synchronized

After searching NetSuite for events that were updated, a check is done in the orchestration to verify that there are some events. The *Else* branch is followed if there are no events and if the orchestration is started through an HTTP Receive Request, a response is sent.

To add a response in this case:

1. Right-click the If...Then block that you created previously, as shown in Figure 11-33. Select **Copy**.

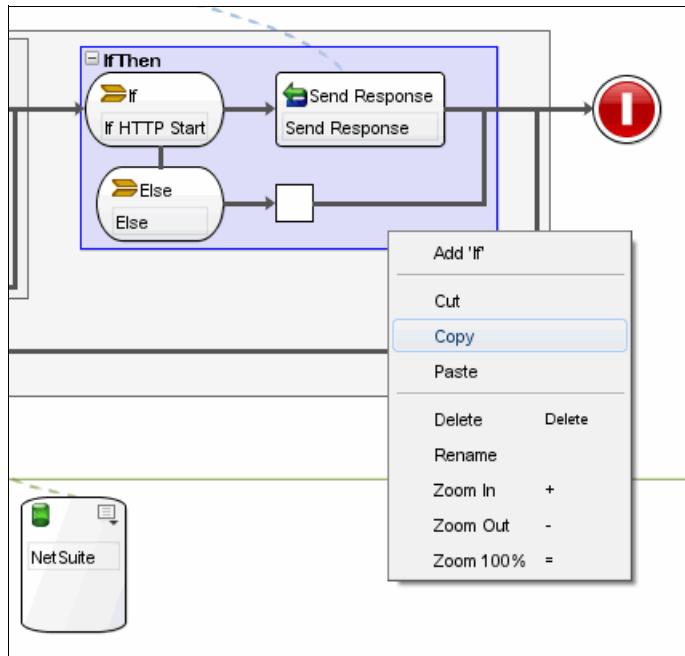


Figure 11-33 Copying an entire If...Then activity block

- Paste the block into the orchestration in the Else branch of the first If...Then activity in the orchestration, as shown in Figure 11-34.

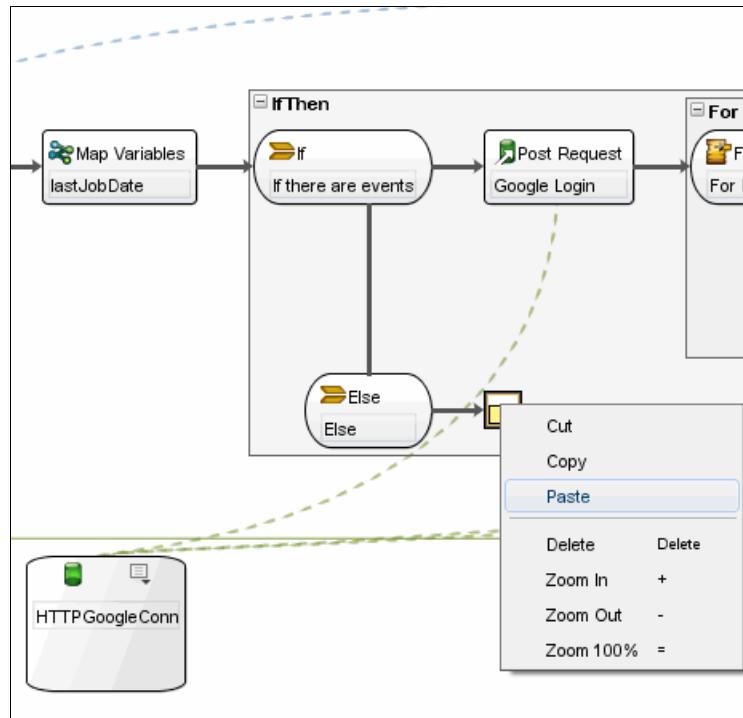


Figure 11-34 Pasting an IfThen activity block

- Click the **Send Response** activity.
- Select **Map Outputs** in the checklist, and set the default value for the body as follows:
No events found to be synchronized

11.5.5 Catching errors that occur while searching NetSuite

This scenario adds a simple Try...Catch error handler to send an email if accessing NetSuite fails. To catch errors that occur while searching NetSuite:

- From the Logic section of the Activities tab, drag a Try activity to the left of the Search Records (Lookup NetSuite Calendar Events) activity, as shown in Figure 11-35.

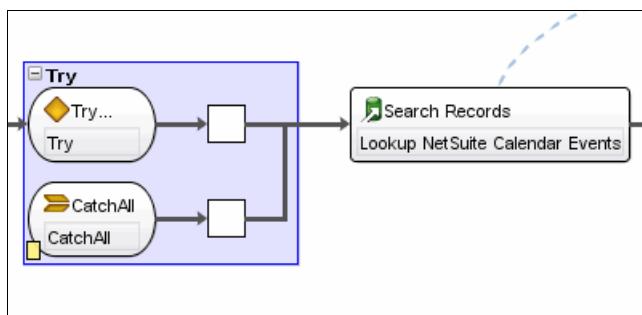


Figure 11-35 Adding a Try activity to the orchestration

- Drag the Search Records activity into the Try block, as shown in Figure 11-36 on page 456.

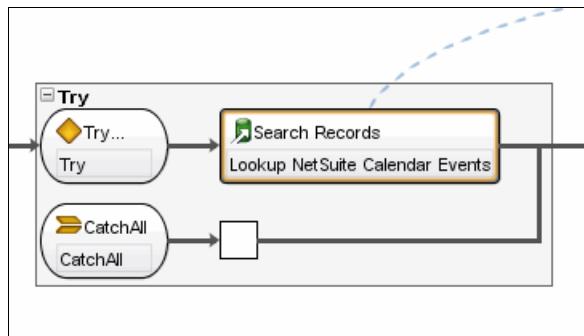


Figure 11-36 Search Records activity in the Try block

3. Go to the Project tab. Right-click **Endpoints**, and then select **Create Endpoint → Email**. A new tab is created with the email configurations, as shown in Figure 11-37 on page 457.
 - a. Select the **Send Email to Server** option.
 - b. Enter the host name and port for the email server, in this case, `smtp.gmail.com` as the host name and port 465.
 - c. Select the **Log into the Server with User Name and Password** option.
 - d. Use the drop-down in the User Name and Password fields to select the configuration parameters for your Google user name and password.
 - e. Select **Email over SSL**.
 - f. Click **Test Connection**.

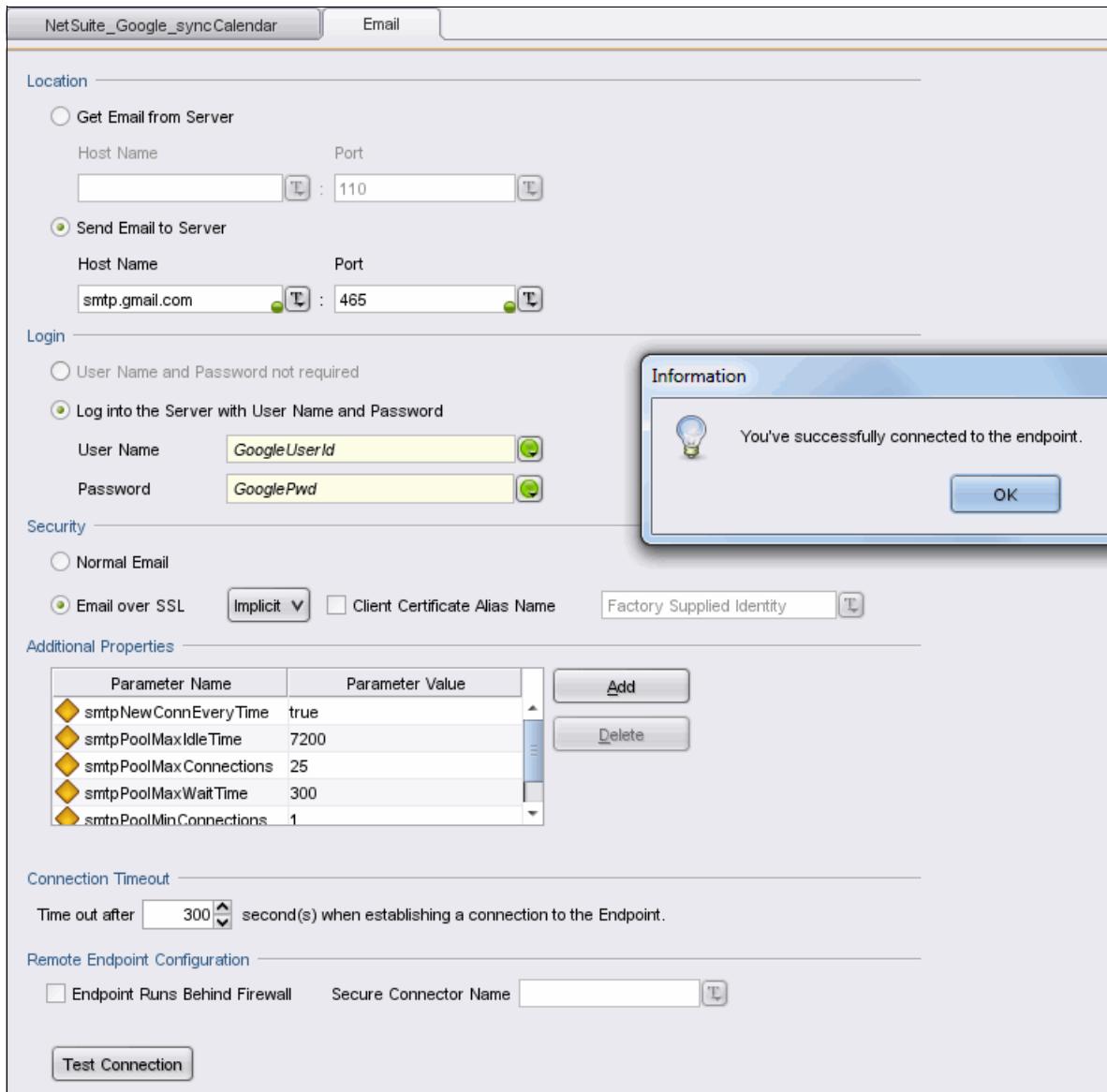


Figure 11-37 Configuration for Email endpoint

4. Select the workspace tab for the orchestration.
5. From the Email section on the Activities, drag a Send Email activity to the CatchAll branch of the Try activity, as shown in Figure 11-38.

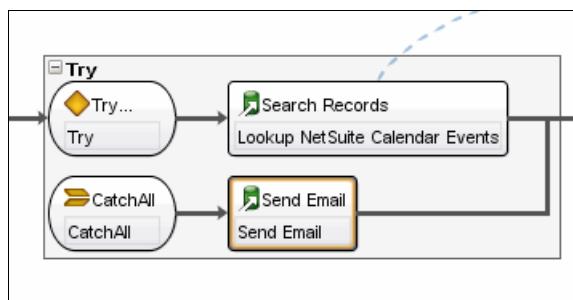


Figure 11-38 Send Email activity

6. Select **Pick Endpoint** in the Checklist. Click **Browse**, select the Email endpoint, and then click **OK**.
7. Click **Configure** in the Checklist.
 - a. Enter the email details. In this scenario, set the From and To email addresses as the Google email address and leave the CC and BCC blank.
 - b. Enter the following text for the Subject, as shown in Figure 11-39:
NetSuite Search Records Failure



Figure 11-39 Email Configure checklist settings

8. Click **Map Inputs** in the Checklist. The To Activity section requires the body of the email. This scenario sends the job details.
9. Click **Select Inputs**, and select the JobInfo variable. Click **OK**.
10. From the Functions tab, drag a Concatenate function into the middle untitled pane of Map Inputs:
 - a. From the From Activity pane map the jobId, jobStartTime, and projectName to the Concatenate function box in the middle section. To form a map, click the field, and then drag to the function. Refer to Figure 11-40.

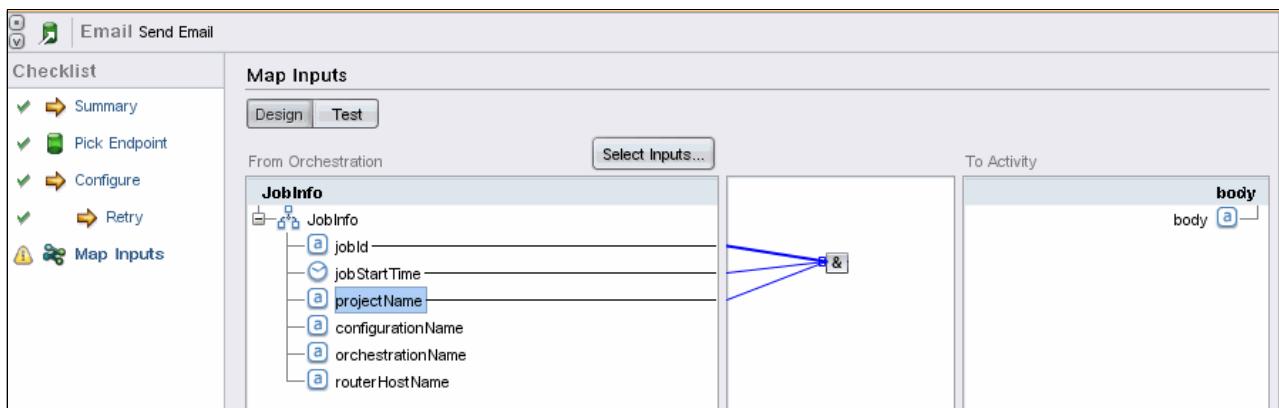


Figure 11-40 Mapping fields to a Concatenate function

- b. Map the Concatenate function in the middle pane to the body in the To Orchestration pane.
11. Double-click the Concatenate function:
 - a. Add three empty fields by clicking **Add** three times.

- b. Use the Up button to position these fields above each of the three job fields, as shown in Figure 11-41.

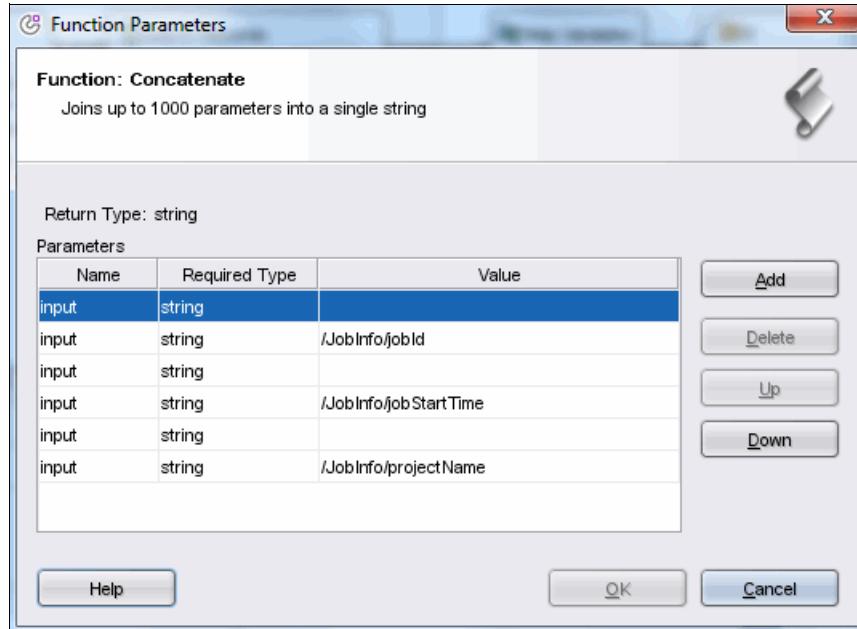


Figure 11-41 Configuring the Concatenate function

12. In the Functions Parameter dialog box, complete these steps (refer to Figure 11-42):

- In the first empty text box, enter Job ID. Ensure that there is a space at the end of the text string.
- In the second empty text box, enter Job Start Time. Ensure that there is a space at the end of the text string.
- In the third empty text box, enter Project Name, and then click **OK**. Ensure that there is a space at the end of the text string.

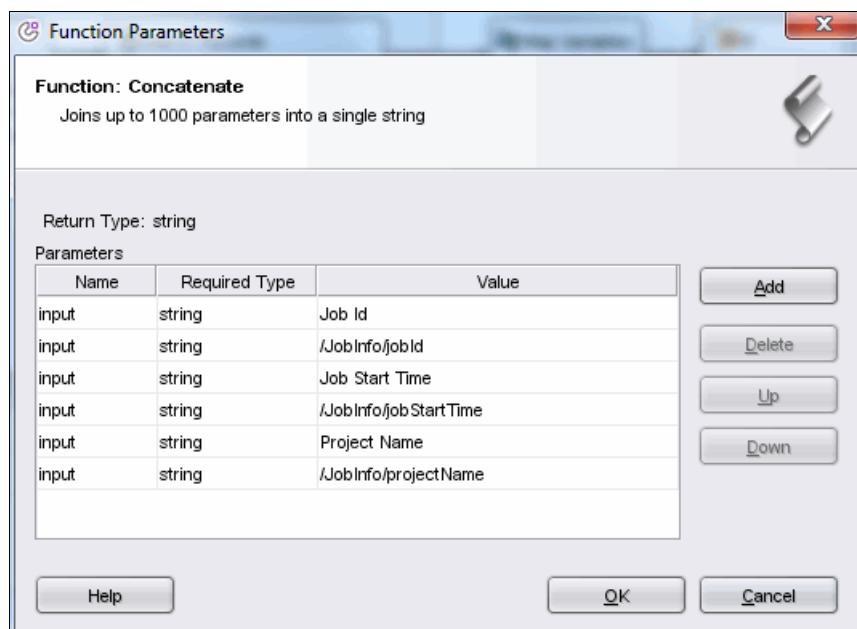


Figure 11-42 Completed Concatenate configuration

13. Right-click the Concatenate function, and select **Apply Function Graph**, as shown in Figure 11-43.

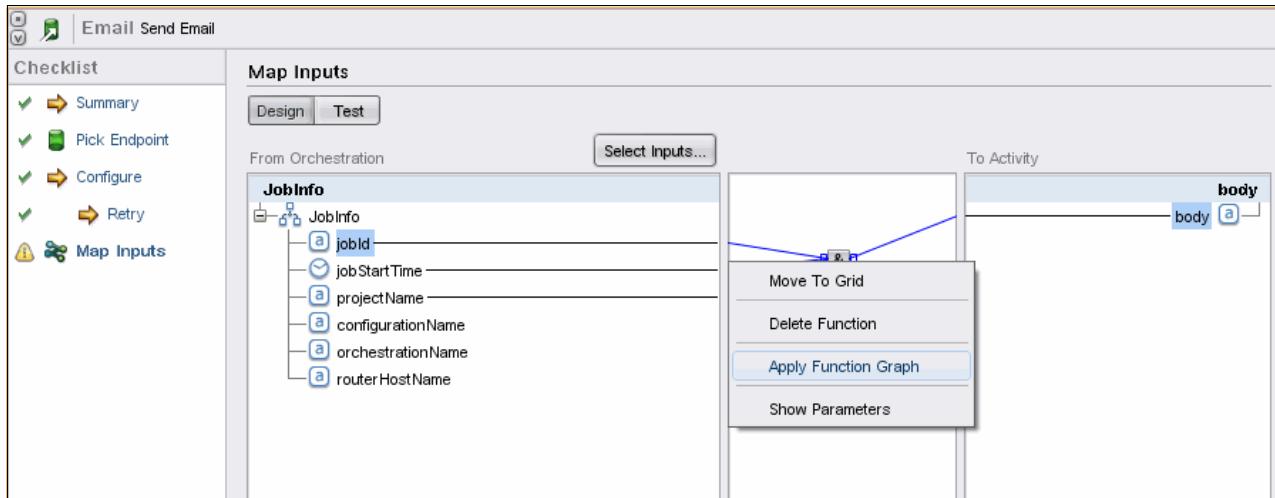


Figure 11-43 Apply Function Graph

14. Save the project by selecting **File → Save**.

11.5.6 Error handling

This scenario added a single error handler around the NetSuite Search Records activity. A handler must also be added around all essential activity to enable quick diagnosis of any errors. You can add a generic CatchAll branch to the orchestration by right-clicking the icon that identifies the beginning of the orchestration (▶) and selecting **Add CatchAll Branch menu**.

Error handling is covered in more detail in Chapter 9, “Common error handlers” on page 349.

11.6 Publishing and deploying the project to the development environment

You can now publish the project to the development environment. After you publish a project, you can configure the project, for example with error logging levels, and then deploy the project. The project then waits for a starter activity to occur. This orchestration can be started with an HTTP request or through a schedule. Before the project is published to the development environment, the schedule will be changed to every hour.

11.6.1 Changing the job schedule time

To change the job schedule time:

1. Click the **Schedule Job** activity.
2. Select **Configure** from the Checklist.
3. Set the orchestration to start every hour.
4. Save the project.

11.6.2 Publishing the project

To publish the project, in the Cast Iron Live Modify tab, click the Publish icon, and then select **Development**.

Environment name: The choice of environments is shown only if you have more than one environment. You might not have an environment called Development, in which case you must select an appropriate environment for testing this scenario.

11.6.3 Deploying the project

To deploy the project, follow these steps:

1. Click the Development environment tab. Click the project configuration (circled in Figure 11-44).

Configuration	Last Published	Runni...	Compl...
▶ NetSuite_Google_syncCalendar (Undeployed)	10/20/2011 03:27	0	0

Figure 11-44 Project configurations in Cast Iron Live

2. Click **Edit**, as shown in Figure 11-45.

Name	Status	Logging Level	Log Synchronously	Max Simu...
GoogleCreate	Enabled	Error Values	Disabled	1
GoogleLogin	Enabled	Error Values	Disabled	1
GoogleUpdate	Enabled	Error Values	Disabled	1
NetSuite_Google_syr...	Enabled	Error Values	Disabled	1

Figure 11-45 Project Configuration Details

- Select **All** for the Logging Level for the orchestration NetSuite_Google_syncCalendar, as shown in Figure 11-46.

Name	Enabled	Logging Level	Log Synchrono
	<input type="checkbox"/>	None	<input type="checkbox"/>
GoogleCreate	<input checked="" type="checkbox"/>	Error Values	<input type="checkbox"/>
GoogleLogin	<input checked="" type="checkbox"/>	Error Values	<input type="checkbox"/>
GoogleUpdate	<input checked="" type="checkbox"/>	Error Values	<input type="checkbox"/>
NetSuite_Google_syncCalendar	<input checked="" type="checkbox"/>	All	<input type="checkbox"/>

Figure 11-46 Project configuration settings

- Click **Save** (located at the bottom left corner). Click the Run Configuration icon ().

11.6.4 Testing the project

After you deploy the project configuration, you can either wait for the schedule start or send an HTTP Request. This section shows how to send an HTTP request:

- In a web browser, enter the following web address, where *<environment>* is the name of your environment, and *<receive request listen at URL>* is the URL that is set in the HTTP Receive Request activity:

`https://provide.castiron.com/env/<environment>/<receive request listen at URL>`

This scenario uses the following web address:

`https://provide.castiron.com/env/Development/syncNetSuiteGoogleCalendar`

Note: If you are using the evaluation cloud, use the following web address:

`https://eval-provide.castiron.com/env/<environment>/<receive request listen at URL>`

- When prompted, enter your Cast Iron Live login credentials, as shown in Figure 11-47. (The dialog box that displays varies according to the browser that you use to log in.) These credentials determine the tenant to which to send the request. If your login is successful, you receive a message.

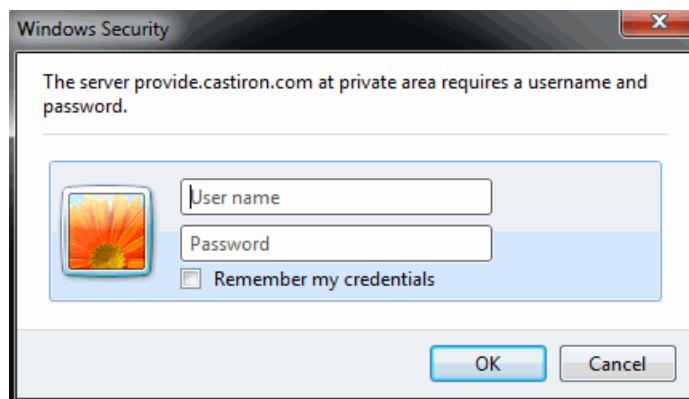


Figure 11-47 Cast Iron Live request for credentials

11.6.5 Connecting to Google

This scenario uses a Cast Iron TIP that consists of the following orchestrations that connect to Google:

- ▶ The GoogleLogin orchestration takes the Google login details and posts a request to Google to receive an authentication token.
- ▶ The GoogleCreate orchestration takes calendar data as its input and forms the XML string that is required to post to Google. If the HTTP Post receives a 302 redirect code, it tries a new post. You can find further details about the Google API at:
http://code.google.com/apis/calendar/data/2.0/developers_guide_protocol.html
- ▶ The GoogleUpdate orchestration is similar to the GoogleCreate orchestration, but it connects to just the Google Calendar to update events.

11.6.6 Google Connector

A development of this scenario is to take the Google orchestrations in the project and, using the Cast Iron Connector Developer Kit (CDK), create a connector. This process creates a series of activities that are accessible on the Studio Activity tab that can simplify orchestration development and make resulting projects more robust.

You can find information about the CDK in Chapter 8, “Building custom plugin connectors” on page 321.



Scenario: Data enrichment and aggregation

This chapter describes a scenario for reading a file using FTP, enriching and aggregating the data extracted from that file with data pulled from a database, and then writing the data to Lotus® Domino®.

This chapter includes the following topics:

- ▶ Cast Iron concepts demonstrated in this scenario
- ▶ Scenario overview
- ▶ The order file structure
- ▶ The item structure
- ▶ The product database structure
- ▶ The Domino document structure
- ▶ The scenario orchestration
- ▶ Preparing the scenario
- ▶ Creating entities for the scenario
- ▶ Building the orchestration

12.1 Cast Iron concepts demonstrated in this scenario

The scenario demonstrates the use of the following Cast Iron concepts:

- ▶ FTP Polling
- ▶ Flat File Schemas
- ▶ Applying XSLTs
- ▶ Database Lookup
- ▶ Data Quality Merge
- ▶ Lookup Tables
- ▶ Custom Functions
- ▶ Lotus Domino Create Documents
- ▶ Job Keys
- ▶ Error Handling

12.2 Scenario overview

In this scenario, a company receives order files from its partners over FTP. The company needs to poll the FTP server to get the files.

When each order file is received, the company needs to extract a product ID from the order file and use the product ID to retrieve product data from a database. This product data is aggregated with the order data and used to create a Lotus Domino document, which represents an invoice for the order.

A Domino application then takes this invoice data and uses it to send an invoice to the customer who made the order. The Domino application that sends the invoice is outside of the scope of this scenario.

Figure 12-1 illustrates the flow of this scenario.

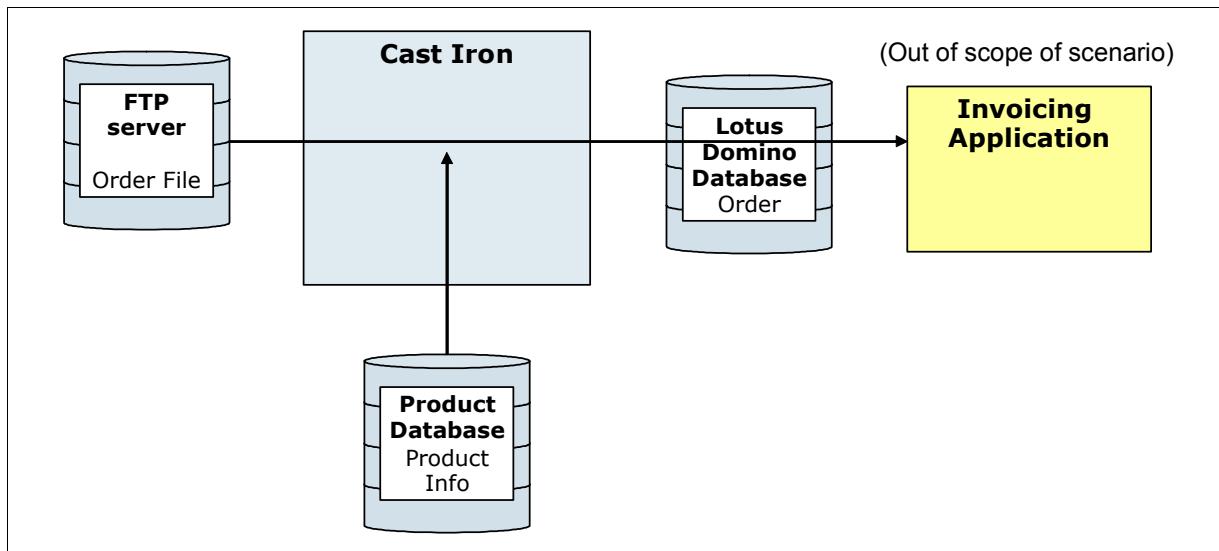


Figure 12-1 Scenario overview

Each order file that is sent from the partner can contain multiple orders from different customers. Multiple products can be ordered within each customer order. Each product order is held as an item within the order. Figure 12-2 shows a representation of an order file.

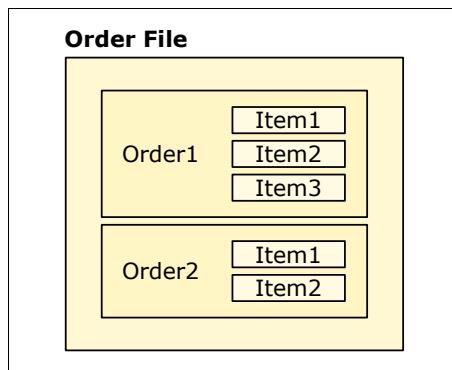


Figure 12-2 Order file structure

The invoicing system needs one document for every item received, not for every order. Thus, each order must be transformed into a set of items, as shown in Figure 12-3.

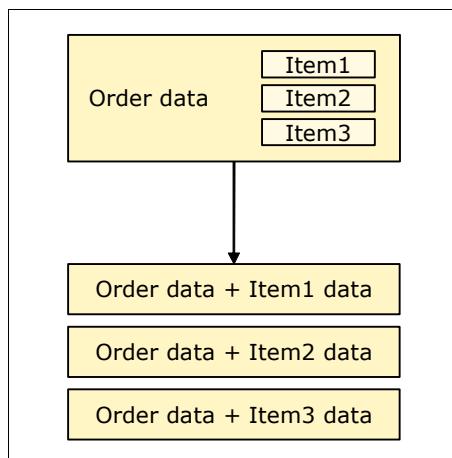


Figure 12-3 Transforming orders to items

The order file contains the product ID and quantity for each product that is ordered. It does not contain the product description or the price of each product. This data is retrieved from the product database.

In addition, products are different sizes and weights. Thus, when they are delivered to the customer, products with smaller sizes and weights can be transported using Air Freight, which can be faster but also more expensive. Products with larger sizes and weights can use slower transport methods, which can be less expensive. The product database contains a transport type for each product that shows which transport method can be used for that product. This data is also retrieved from the product database.

The information retrieved from the product database is combined with each item, as shown in Figure 12-4 on page 468.

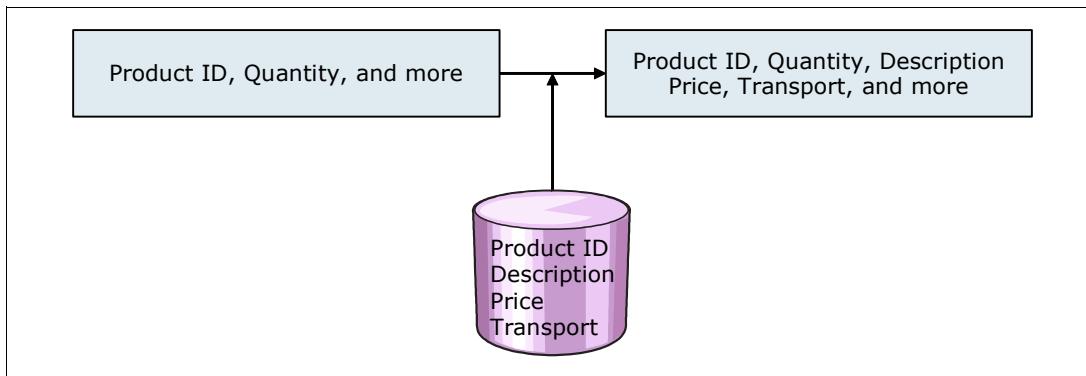


Figure 12-4 Aggregating item data with data from database

An order date and a total price for each item is added to the data. A delivery date is calculated using the transport for the product, and a message giving the estimated delivery date is added to the data, as shown in Figure 12-5. This field is a message string and not just the delivery date because it includes extra text besides the date.

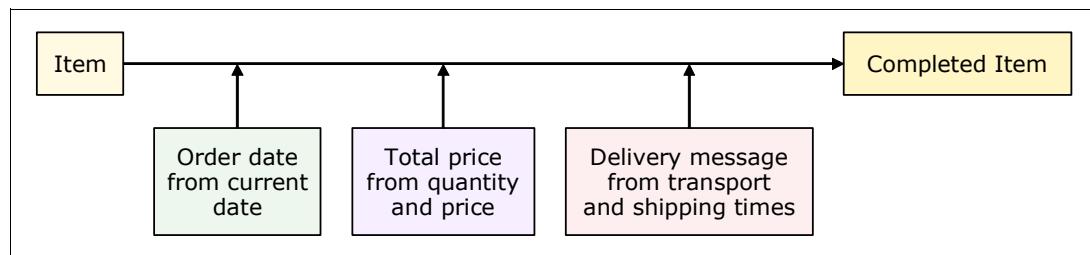


Figure 12-5 Enriching order data

Finally, a Lotus Domino document is created for each item, which is then used to send an invoice to the customer.

12.3 The order file structure

Each order record held within the file has the structure shown in Figure 12-6.

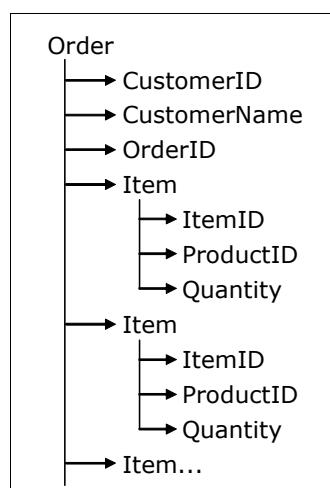


Figure 12-6 Order record structure

Figure 12-7 shows an example of the structure of the incoming order file.

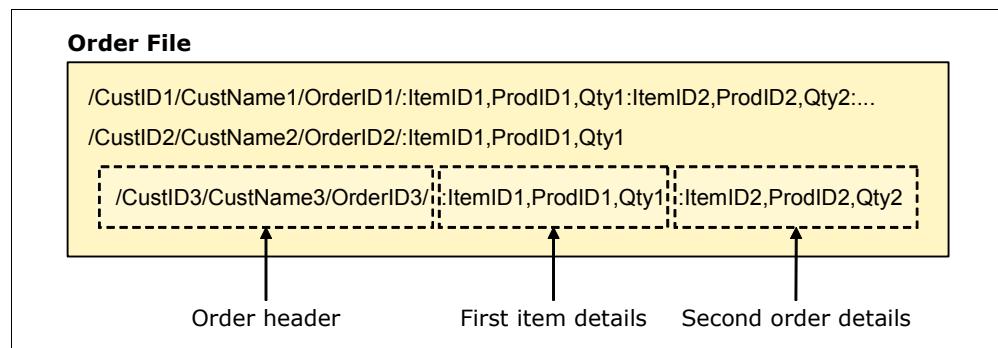


Figure 12-7 Detailed order file structure

Each order file contains multiple orders, with one order per line. Each order is separated by an end-of-line character.

The data in each order uses the delimiters shown in Table 12-1 to separate it.

Table 12-1 The delimiters used in the order file

Delimiter	What it is used for
Slash (/)	Indicates the start of each top level piece of data in the order, such as CustomerID, CustomerName, OrderID, and the items
Colon (:)	Indicates the start of each item
Comma (,)	Separates each piece of data in an item, for example ItemID, ProductID, and Quantity

The invoice system requires that a Domino document be created for every item in the order file.

12.4 The item structure

After each order file is parsed, it is transformed into a set of items with the structure shown in Figure 12-8.

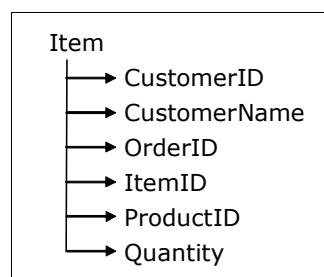


Figure 12-8 Item record structure

12.5 The product database structure

The product database table contains the structure shown in Table 12-2.

Table 12-2 The PRODUCTS database table structure

Column	Data Type
ID	VARCHAR(10)
DESC	VARCHAR(200)
PRICE	DECIMAL(8,2)
TRANSPORT	VARCHAR(10)

12.6 The Domino document structure

The Domino document that is created has the structure shown in Table 12-3.

Table 12-3 The Domino document structure

Field	Data Type
CUSTID	Number
CUSTNAME	Text
ORDERID	Number
ORDERDATE	Date/Time
DELIVERYMSG	Text
ITEMID	Number
DESC	Text
QTY	Number
PRICE	Number
TOTAL	Number

12.7 The scenario orchestration

Figure 12-9 shows an overview of the orchestration that will be created in this scenario.

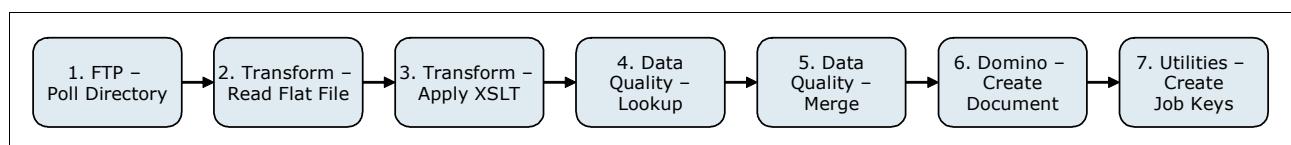


Figure 12-9 Overview of the scenario orchestration

The flow of the processing in this scenario is:

1. FTP – Poll Directory:
 - a. An FTP Poll Directory activity polls an FTP server every 20 seconds for any files in a directory. When a file is detected, the data in the file is read in to the orchestration.
 - b. The name of the file is stored in a variable.
2. Transform – Read Flat File:
 - a. The data from the file is read in as a single string of data. This data is then parsed using a Flat File Schema to turn it in to a set of orders with the correct structure. Flat File Schemas are explained in 3.10.4, “Flat File Schema parsing” on page 151.
 - b. The total number of orders that are read is stored in a variable.
3. Transform – Apply XSLT
4. Data Quality – Lookup

A database Lookup is performed to read the product description, price, and transport for each item from the products database. The Lookup activity combines the functionality of a For Each loop with a Database Execute Query.

The activity returns two sets of data:

- The first set contains a list of the aggregations of the original items and the fields returned from the products database, as shown in Figure 12-10.

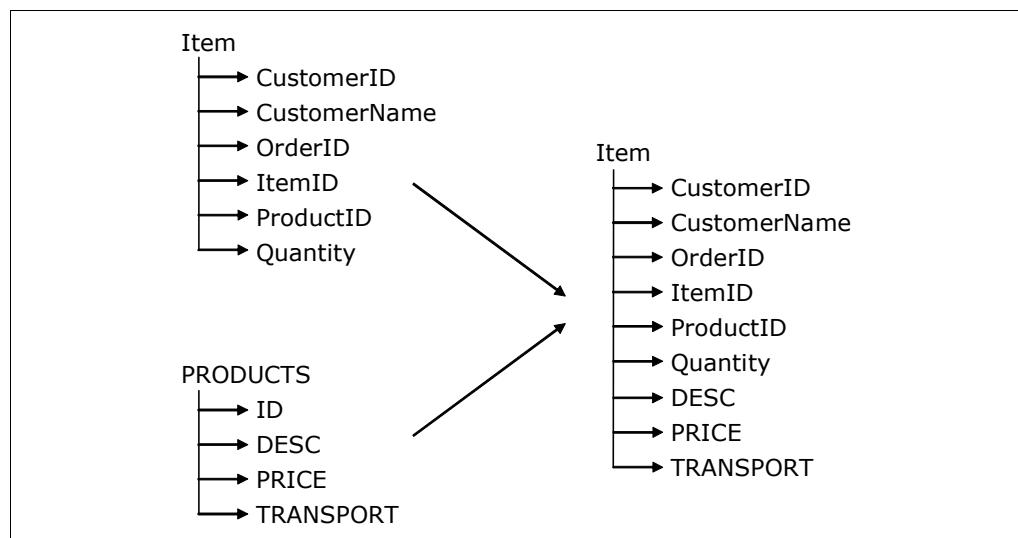


Figure 12-10 Aggregating item record structure and database record structure

- The activity also returns a list of all of the items with products that cannot be found on the products database. This list of items is given default values for the missing fields, as follows:

DESC	ERROR: PRODUCT ORDERED WAS INVALID
PRICE	0
TRANSPORT	ERROR

5. Data Quality – Merge

The two sets of data that are returned from the Lookup activity are then merged to produce one list of items.

Hint: If two items have the same Customer ID, Order ID, and Item ID, the item from the Lookup Success list is used. This occurrence is an error condition and must not happen if the order file is created correctly.

6. Domino – Create Document

A data structure is created for each item in the file. This data structure maps the relevant fields to those fields that are needed for the Domino document.

The following fields are mapped directly to their equivalents in the Domino document:

- CustomerID
- CustomerName
- OrderID
- ItemID
- Description
- Quantity

The following fields are mapped as indicated:

- The OrderDate field is populated with the current date.
- The Price field is returned as a String from the Lookup activity. The Number function is used to convert this to a number.
- The Total field is calculated from the Quantity and Price.
- The DeliveryMessage field is populated using a Lookup Table and a Custom Function:
 - The Lookup Table is used to look up the transport for the item and to return a number of shipping days.
 - The Custom Function takes this number of days and adds it to the current date. The function then returns a delivery message of the form:

Your expected delivery date is <Delivery Date>

The Domino document is then created in the Domino database.

7. Utilities – Create Job Keys

To allow a Cast Iron administrator to view the status of order files that have been processed by the orchestration in the Web Management Console, two Job Keys are created. The two Job Keys hold the name of the file that was processed and the total number of orders in that file. These Job Keys are populated using variables set in earlier activities.

An incoming order file using the format shown in Example 12-1 produces four Domino documents as shown in Figure 12-11 on page 473.

Note: The Order and Delivery dates depend on the date on which the orchestration ran.

Example 12-1 A sample incoming order file

```
/8660001/Customer1/1/:1001,PROD24,3:1002,BadProduct,1:1003,PROD43,1  
/8970001/Customer2/1/:1001,PROD35,1
```

Order Form	
Customer ID	8660001
Customer name	Customer1
Order ID	1
Order Date	08/10/2011 16
Delivery Message	Your expected delivery date is Sun Oct 09 2011
Item ID	1001
Description	Product 24
Quantity	3
Price	\$ 56.55
Total	\$169.65

Order Form	
Customer ID	8660001
Customer name	Customer1
Order ID	1
Order Date	08/10/2011 16
Delivery Message	NO DELIVERY DATE SET
Item ID	1002
Description	ERROR: PRODUCT ORDERED WAS INVALID
Quantity	1
Price	\$0.00
Total	\$0.00

Order Form	
Customer ID	8660001
Customer name	Customer1
Order ID	1
Order Date	08/10/2011 16
Delivery Message	Your expected delivery date is Sun Oct 09 2011
Item ID	1003
Description	Product 43
Quantity	1
Price	\$32.01
Total	\$32.01

Order Form	
Customer ID	8970001
Customer name	Customer2
Order ID	1
Order Date	08/10/2011 16
Delivery Message	Your expected delivery date is Tue Oct 11 2011
Item ID	1001
Description	Product 35
Quantity	1
Price	\$ 97.41
Total	\$ 97.41

Figure 12-11 The Domino documents produced from the sample incoming order file

Every orchestration must include error handling. For this scenario, error handling is performed by the common error handler described in Chapter 9, “Common error handlers” on page 349.

In the completed scenario, the Lookup, Merge, and Create Documents activities are run inside a Try activity. The orchestration also includes a global error handler. Both the Try activity and the global error handler use Web Services Invoke Service activities to call the common error handler.

12.8 Preparing the scenario

An FTP server, a DB2® database, and a Domino database must be created before the scenario can be built.

Additional material: Files that you can use to set up this environment were made available for download. See Appendix A, “Additional material” on page 517 for information about how to get these materials. The files are in the Scenario3 folder. Unzip this folder into a new folder on your system, and open scenario3.sp3 from WebSphere Cast Iron Studio.

12.8.1 Preparing the FTP server

An FTP server is needed for this scenario. The orchestration described in this chapter polls the FTP server for the orders file. For this scenario you can install a new FTP server or use an FTP server already installed in your environment. Create a new user ID and password, and give that user read access to a specified directory of your choice. You will use the user name that you create later in this scenario.

12.8.2 Preparing the DB2 database

A database server is needed for this scenario. This chapter builds a scenario using IBM DB2; however, with a few changes, you can adapt the script that is provided with this book to use a database of your preference. Two database scripts files are provided in the download materials that create the database used in this scenario:

- ▶ CreateCastIronDatabase.bat initializes a DB2 command environment and then executes the second script
- ▶ c_table.bat does the following:
 - Drops a DB2 database called SCENARIO.
 - Creates a DB2 database called SCENARIO.
 - Creates a table called PRODUCTS with the columns needed for the scenario.
 - Inserts 50 rows of product data of the form needed for the scenario.

The database user must have read access to the database and the products table that this scenario uses.

For information about supported databases, refer to:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/db_creating_editing_a_database_activity_endpoint.html

12.8.3 Preparing the Domino database

The Notes database template for the database used in this scenario is available for download with this book. The template is called CastFE.ntf.

1. Download the additional material files, and extract the CastFE.ntf database template. Store it in your Notes/Data directory.
2. In your IBM Lotus Notes® workspace, select **File** → **Application** → **New**, as shown in Figure 12-12 on page 475. On the window:
 - a. Select the Server where you want to install the database.
 - b. Enter a title: CastFE
 - c. Enter the file name to use: CastFE.nsf
 - d. In the template section, select your local server and the CastFE template.
 - e. Click **OK**.

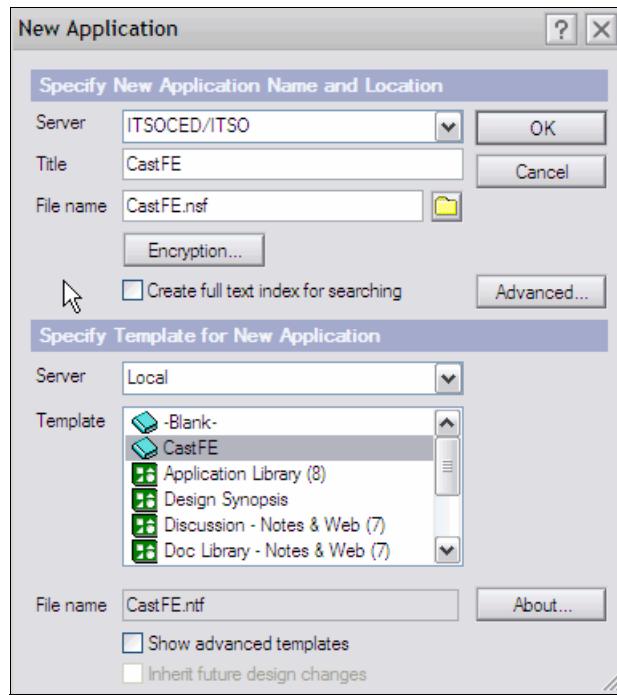


Figure 12-12 Creating a new Notes database from a template

12.9 Creating entities for the scenario

Before you develop the business logic in the orchestration, you must create entities, such as endpoints, flat file schemas, custom functions, and lookup tables. For this scenario, use the following steps to build the entities:

1. Open WebSphere Cast Iron Studio (referred to in this document as *Studio*). Select **File** → **New** → **New Project** to create a new project. If the Save current project window opens, click **Yes**. Give the project a name of Scenario3, and save the project to a suitable Project Directory. Click **OK**.
2. On the Project tab, rename the orchestration that was created for you, which is currently called Orchestration. Right-click the orchestration, and select **Rename**. Rename the orchestration `FTP_DB2_to_Domino_sync_Orders`.

12.9.1 Creating and testing an FTP endpoint

To create and test an FTP endpoint:

1. In the Project tab, create the FTP Endpoint by right-clicking **Endpoints**, and then selecting **Create Endpoint** → **FTP**.
2. Enter the IP address for the FTP server. Click the green dot that displays at the bottom right of this field to create a configuration property for the field. Enter `FTPServer` as the name of the configuration property, and click **Create**.
3. In the Login section of the FTP endpoint, select the **Log into the Server with User Name and Password** option.
4. Enter the user name and password that you set up for the FTP server. Create a configuration property for the User Name field called `FTPUserId`, and create a configuration property for the Password field called `FTPPassword`.

Refer to Figure 12-13. Leave any settings that are not shown here at the default values.

The screenshot shows the 'Location' and 'Login' sections of the FTP endpoint configuration. In the 'Location' section, 'Host Name' is set to 'ftplib://FTPServer' and 'Port' is set to '21'. A checked checkbox 'Use default port' is also present. In the 'Login' section, a radio button 'Log into the Server with User Name and Password' is selected. The 'User Name' field contains 'FTPUserId' and the 'Password' field contains 'FTPPassword'.

Figure 12-13 The completed FTP endpoint

5. Click **Test Connection** at the bottom of the FTP endpoint pane to confirm that the connection values entered for the endpoint are correct.
6. If the connection test fails, change the values for the configuration properties by selecting **Project → Configuration Properties** and then testing again.

12.9.2 Creating and testing the DB2 endpoint

To create and test the DB2 endpoint:

1. In the Project tab, create the Database Endpoint by right-clicking **Endpoints** and then selecting **Create Endpoint → Database**.
2. For the database type, select **DB2 UDB** from the drop-down list.
3. Enter the database name. Click the green dot that displays at the bottom right of this field to create a configuration property for the field. Enter **DB2Name** as the name of the configuration property, and click **Create**.
4. Enter the database server IP address (or host name). Create a configuration property for the field. Enter **DB2Server** as the name of the configuration property.
5. Enter the user name and password for the database. Create one configuration property for each field, and name these properties **DB2Username** and **DB2Password**, respectively.
6. In the Additional Parameters section, enter **NULIID** as a parameter value for the **PackageCollection Parameter Name**.

Figure 12-14 on page 477 shows the completed database endpoint. Leave any settings not shown here at the default values.

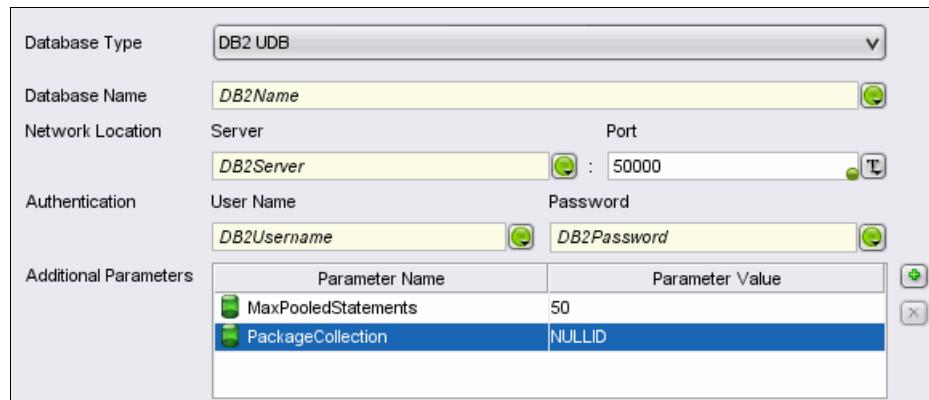


Figure 12-14 Database endpoint settings

7. Click **Test Connection** at the bottom of the Database endpoint pane to confirm that the connection values entered for the endpoint are correct.
8. If the connection test fails, change the values for the configuration properties by selecting **Project → Configuration Properties** and then testing again.
9. On the Project tab, rename the Database endpoint that you just created. Right-click the Database endpoint, and then select **Rename**. Rename the Database endpoint DB2.

12.9.3 Creating and testing the Domino endpoint

To create and test the Domino endpoint:

1. In the Project tab, create the Domino Endpoint by right-clicking **Endpoints**, and then selecting **Create Endpoint → Domino**.
2. Enter the host name that you set up in 12.8.3, “Preparing the Domino database” on page 474. Click the green dot that displays at the bottom right of this field to create a configuration property for the field. Enter DominoServer as the name of the configuration property, and then click **Create**.
3. Enter the user name and password. Create one configuration property for each field, and name these properties DominoUsername and DominoPassword respectively.

Figure 12-15 shows the completed Domino endpoint. Leave any settings not shown here at the default values.

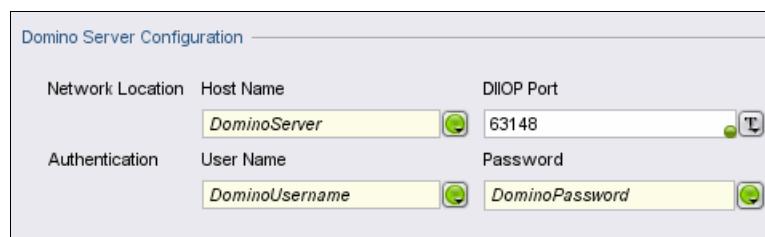


Figure 12-15 Domino endpoints settings

4. Click **Test Connection** at the bottom of the Database endpoint pane to confirm that the connection values entered for the endpoint are correct.
5. If the connection test fails, change the values for the configuration properties by selecting **Project → Configuration Properties** and then test again.

12.9.4 Creating the Orders flat file schema

The Orders flat file schema parses the input file and represents the data. It provides an organized data structure inside the orchestration.

Important: Field names are case sensitive. Entering names and types incorrectly can affect final orchestration results and generate errors.

To create the Orders flat file schema:

1. In the Studio, create a new flat file schema by clicking **Project → New Flat File Schema**. Name the new schema Orders, and then click **OK**.

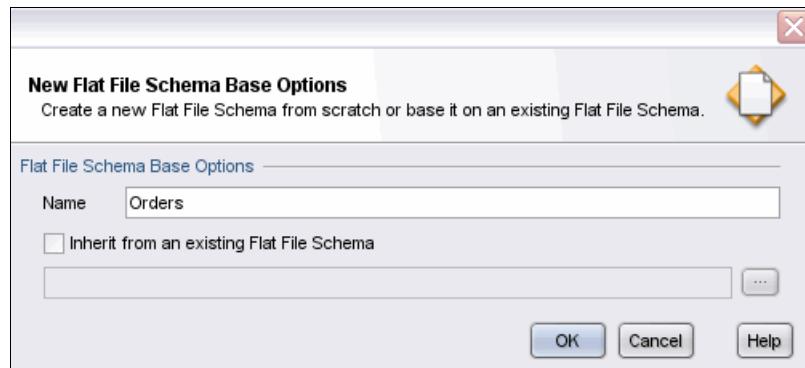


Figure 12-16 Creating a flat file schema

2. In the Orders entity tab, select the **Document Ends with EOS (end of stream)** option. In the Delimiters section, set **Line End** for both Child and Child Delimiter Position options. The flat file schema uses new line as the record delimiter. Refer to Figure 12-17.

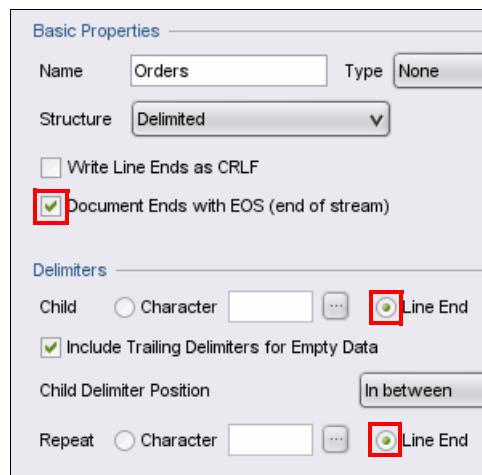


Figure 12-17 Basic Properties and Delimiters for the Orders flat file schema

3. With the Orders entity editor opened and the Schema Layout tab selected, design the file structure. Right-click **Orders** → **New Child** → **Record**, as shown in Figure 12-7 on page 469.

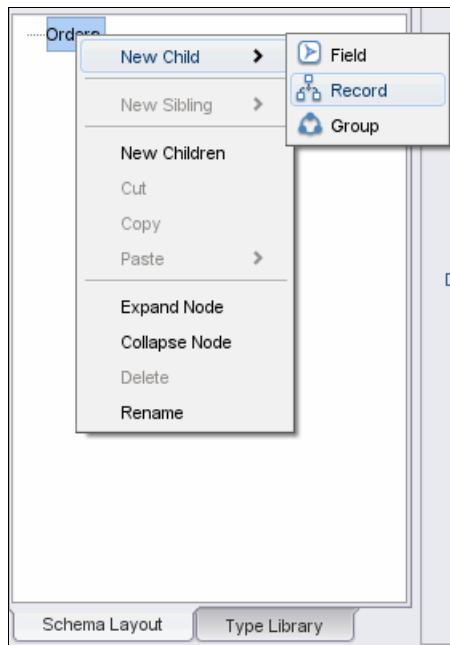


Figure 12-18 Creating a new record for the Orders flat file schema

4. Name the new record Order, and then click **OK**, as shown in Figure 12-19.



Figure 12-19 Creating a new record in the Order flat file schema

5. Select the Order child record that you just created, and complete the following configuration, as shown in Figure 12-20 on page 480:
 - In the Basic Properties panel, select **Unbounded** for Occurrences.
 - In the Delimiters panel, enter a slash (/) character for Child.
 - In the Child Delimiter Position drop-down menu, select **Before**.



Figure 12-20 Order record settings

6. In the Schema Layout tab, right-click **Orders** → **Order** → **New Child** → **Field** (Figure 12-21):
 - a. Name the new child CustomerID.
 - b. In the Basic Properties panel, change the Type field (below the Name field) to Integer in the **Type** drop-down menu.



Figure 12-21 Creating a new field within the record

7. In the Schema Layout tab, right-click **Orders** → **Order** → **New Child** → **Field**, and name the new child CustomerName.
8. In the Schema Layout tab, right-click **Orders** → **Order** → **New Child** → **Field**:
 - a. Name the new child OrderID.
 - b. In the Basic Properties panel, change the field type to Integer in the **Type** drop-down menu.
9. In the Schema Layout tab, right-click **Orders** → **Order** → **New Child** → **Record**. In the New Record window, name the new child Items, and click **OK**.
Select the Items child record that you just created, and complete the following configuration, as shown in Figure 12-22 on page 481:
 - a. In the Delimiters panel, enter a colon (:) character for Child.
 - b. In the Child Delimiter Position drop-down menu, select **Before**.
 - c. Enter a colon (:) character for the Child Delimiter Position option.

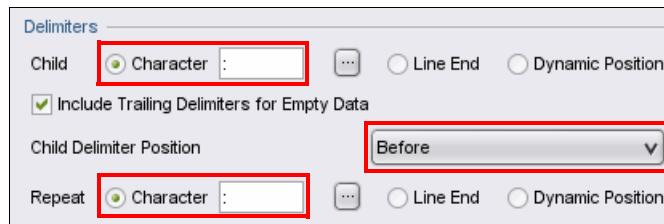


Figure 12-22 Settings for items record

At this point, the layout looks like Figure 12-23.

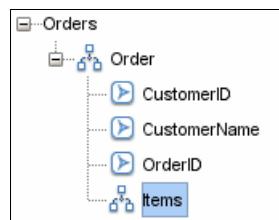


Figure 12-23 Orders layout

10. In the Schema Layout tab, right-click **Orders** → **Order** → **Items** → **New Child** → **Record**, and then:

- In the New Record window, name the new child Item, and click **OK**. Complete the configuration, as shown in Figure 12-24.
- In the Basic Properties panel, select **Unbounded** for the Occurrences setting.
- In the Delimiters panel, enter a comma (,) character for the Child option.

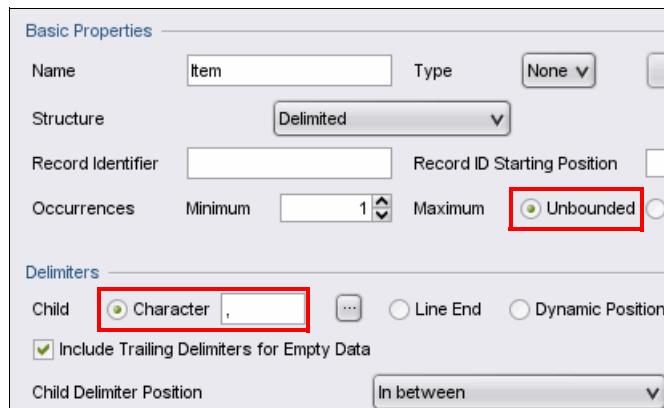


Figure 12-24 Settings for item

11. In the Schema Layout tab, right-click **Orders** → **Order** → **Items** → **Item** → **New Child** → **Field**, and name the new child ItemID. In the Basic Properties panel, change the field type to Integer in the Type drop-down menu.

12. In the Schema Layout tab, right-click **Orders** → **Order** → **Items** → **Item** → **New Child** → **Field**, and name the new child ProductID. In the Basic Properties panel, change the field type to string in the Type drop-down menu.

13. In the Schema Layout tab, right-click **Orders** → **Order** → **Items** → **Item** → **New Child** → **Field**, and name the new child Quantity. In the Basic Properties panel, change the field type to Integer in the Type drop-down menu.

The Orders flat file schema tab in which you are working now looks like Figure 12-25.

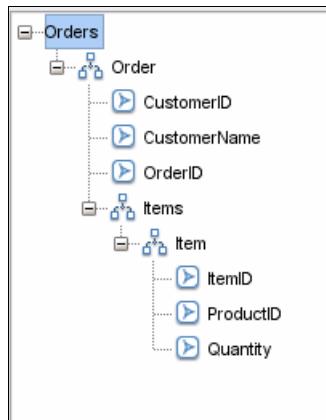


Figure 12-25 The Orders flat file schema for the input file

14. Test the Orders flat file schema. In the Orders entity tab, enter the sample data shown in Example 12-2 into the test panel, and click the Test icon. Refer to Figure 12-26.

This test parses the sample data into an XML structure, which displays on the right side of the test pane. If the test is successful, the XML structure displays and no error message opens.

Example 12-2 Sample data to run a flat file schema test

```
/8660001/Customer1/1/:1001,PROD24,3:1002,BadProduct,1:1003,PROD43,1  
/8970001/Customer2/1/:1001,PROD35,1
```

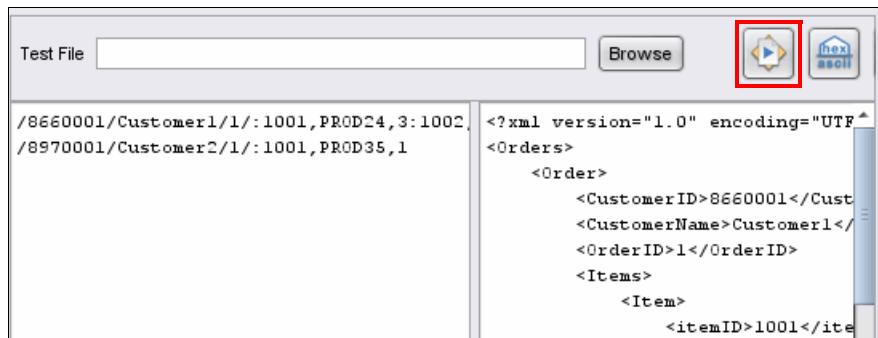


Figure 12-26 Test panel showing the result on the right side

12.9.5 Creating the Items flat file Schema

The next step is to create a flat file schema to hold data about the items within the flow. This Items schema is not used to transform data. Instead, it defines a complex data type that creates a variable in the orchestration. This concept will be clearer when you build the actual orchestration.

You can create the Items flat file schema using the Flat File Wizard:

1. Create a file to be used as a template for the wizard. Create a text file on the local machine with the contents shown in Example 12-3.

Example 12-3 Template data for the Items flat file wizard

```
customerid,customername,orderid,itemid,productid,quantity
```

`customerid,customernumber,orderid,itemid,productid,quantity`

2. In Studio, start the wizard by selecting **Project → Flat File Wizard**.
3. For the sample data, click **Browse**, and then select the template file that you just created. This step loads the file content into the Sample Data container. You can also type the sample data without creating a template file. Refer to Figure 12-27. Click **Next**.

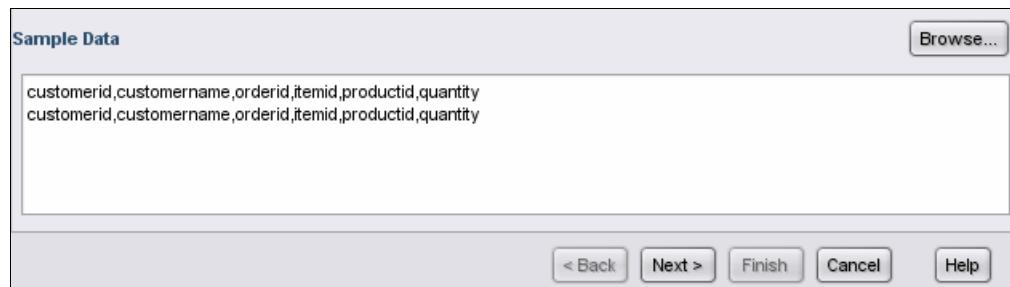


Figure 12-27 Providing sample data for the flat file wizard

4. The template that you provided is a comma-separated value (CSV) format. So, you can accept the defaults for the file format, which is the Character-delimited format option. Click **Next**.
5. Accept the defaults for the Field Delimiters page:
 - Fields separated by comma
 - Fields enclosed by double quoteClick **Next**.
6. In the Header and Trailer pane, select the **Treat first row as header for the field names** option. This option uses the values that you provided in the sample data as field names for the Items flat file schema. Click **Next**.
7. Name the schema that you are creating **Items**, and click **Finish**. Refer to Figure 12-28.

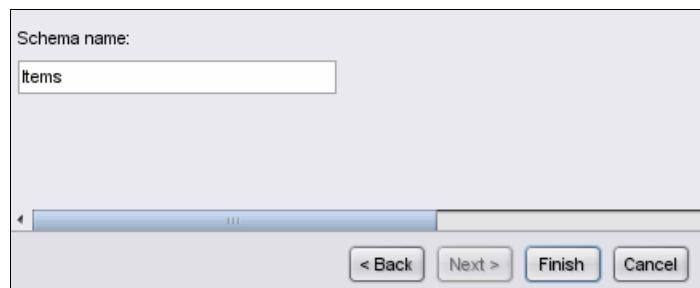


Figure 12-28 Naming the items flat file schema

8. Adjustments are still needed in the Items flat file schema that you just created. In the Project tab, double-click the Items flat file schema in the Flat File Schema section, as shown in Figure 12-29 on page 484.

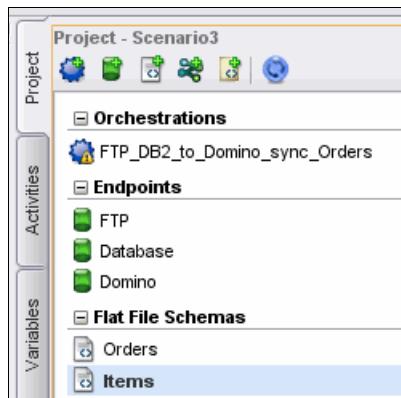


Figure 12-29 Opening the Items flat file schema for edition

9. After the entity tab Items is opened, delete the header node because it is not going to be used. Right-click the header, and select **Delete**, as shown in Figure 12-30.

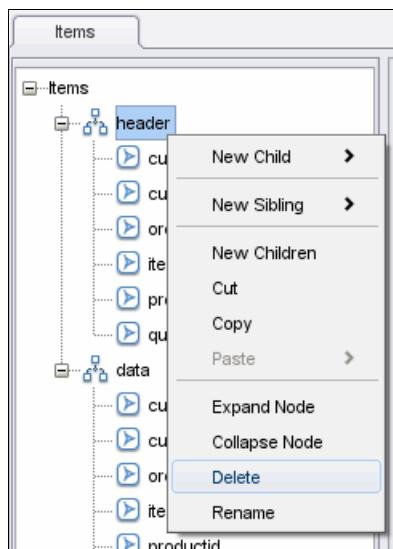


Figure 12-30 Deleting the header node from the Items flat file schema

10. Rename the data node to Item:

- a. Select the data node.
- b. In the Basic Properties panel, select Name, and replace data with Item. Refer to Figure 12-31.

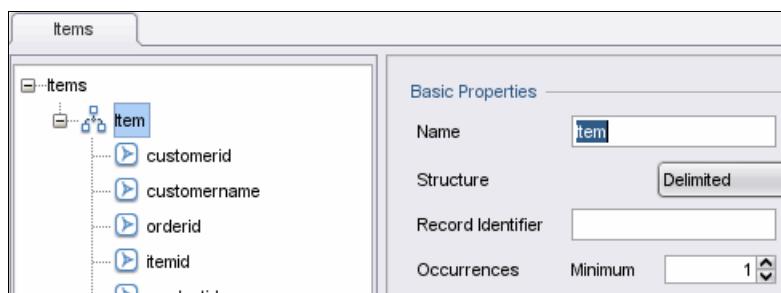


Figure 12-31 Renaming data record structure to Item

11. For each field in the Item structure, update the name and type so that it matches the details shown in Table 12-4.

Table 12-4 Item fields details

Name	Type
CustomerID	integer
CustomerName	string
OrderID	integer
ItemID	integer
ProductID	string
Quantity	integer

12. You can test the Items flat file schema with the template data used within the wizard in the same manner that you tested the Orders flat file schema.
13. At this point, you might want to close any open entity tabs, and save your project.

12.9.6 Creating a custom function

The orchestration uses a function to calculate the expected delivery date. Because this function is not one of the built-in functions in Cast Iron, create a custom function:

1. Select the Functions tab, and then scroll down to the bottom of the functions list. Right-click **Custom Functions**, and then select **Add a new Custom Function**, as shown in Figure 12-32.

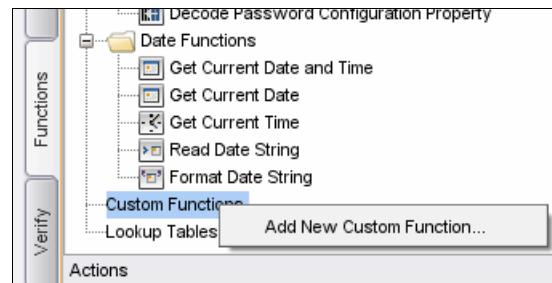


Figure 12-32 Adding a new custom function

2. In the Custom Function window, enter a function name of `createDeliveryDateMessage`. Click **Add** to create a parameter. Call the parameter `numberOfDays`, and give it a type of `number`, as shown in Figure 12-33. Click **Next**.

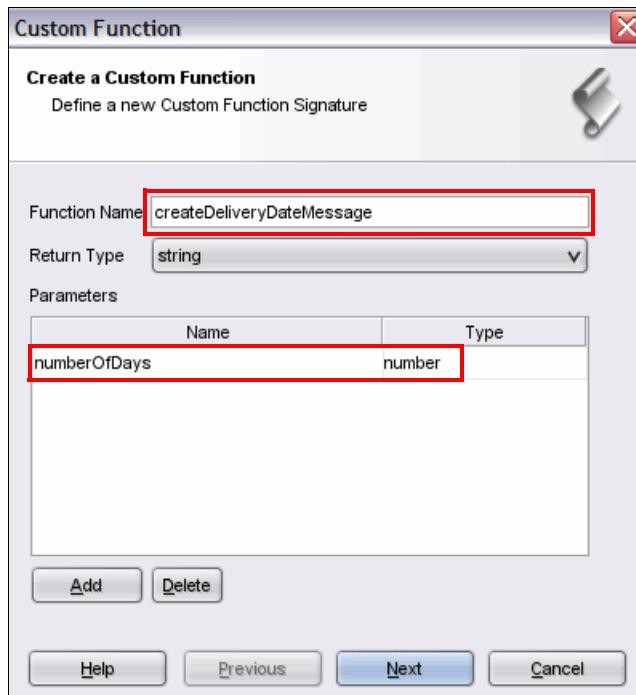


Figure 12-33 Creating a custom function

3. In the Create a Custom Function window, enter the JavaScript function shown in Example 12-4.

This JavaScript function calculates a delivery date based on the current date and the number of days that the delivery takes. A Lookup Table is created in 12.9.7, “Creating a lookup table” on page 487 to select the number of days each product takes to be delivered. This function returns a string that includes the delivery date.

Example 12-4 JavaScript function to calculate a date in future

```
if (numberOfDays == -1){
    return "NO DELIVERY DATE SET"
}
else {
    var today = new Date();
    var calculatedDate = today.getTime() + (numberOfDays*24*60*60*1000);
    var newDate = new Date(calculatedDate);
    return "Your expected delivery date is " + newDate.toDateString();
}
```

- Click **Compile**. When the success message displays, as shown in Figure 12-34. Click **OK**. Click **Finish**.

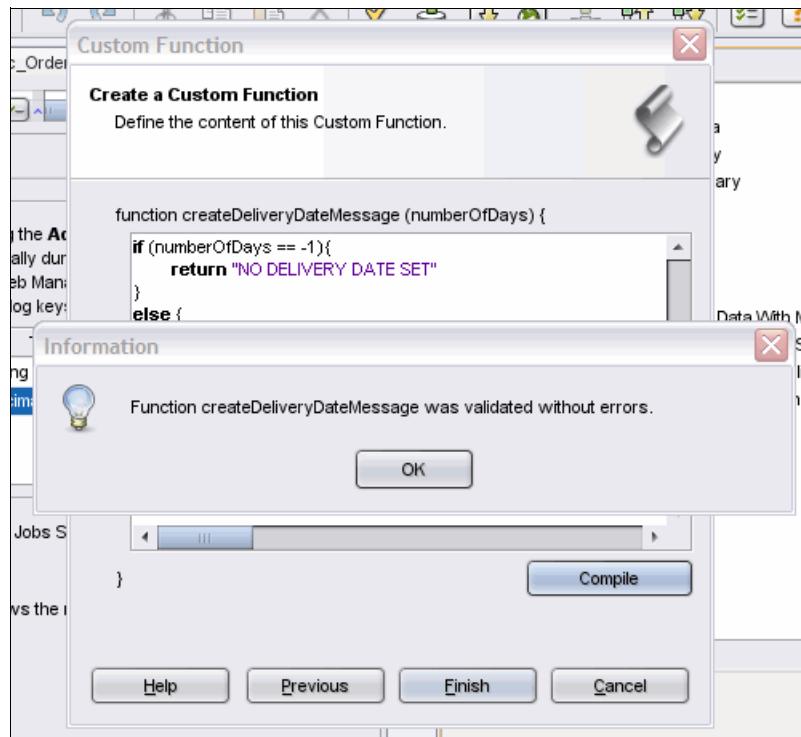


Figure 12-34 Compiling the custom function

12.9.7 Creating a lookup table

Next you must create a lookup table that contains the transport delivery methods and the number of days for each one:

- With the Functions tab selected, click the **Add New Lookup Table** link that is located at the bottom of the Actions panel, as shown in Figure 12-35. You also can create a lookup table by right-clicking the **Lookup Tables** node.

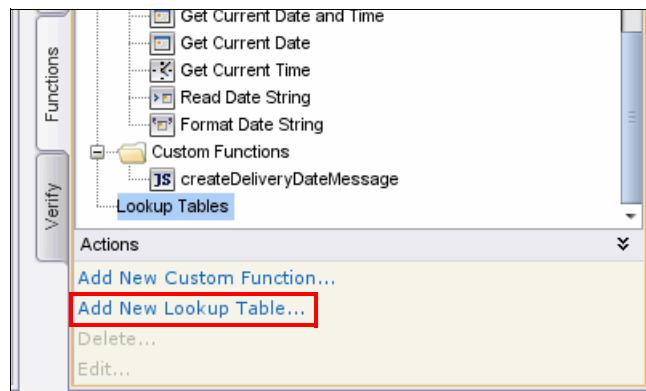


Figure 12-35 Creating a lookup table

- Enter **ShippingTimes** for the Table Name. Use the **Add** button to enter the values listed in Table 12-5 on page 488.

Table 12-5 Lookup Table rows

Key	Value
AIR	1
ROAD	3
SEA	10
SKATEBOARD	30

3. In the Create Lookup Table window, enter -1 for both the Default Value and Value if “nil” fields, as shown in Figure 12-36.

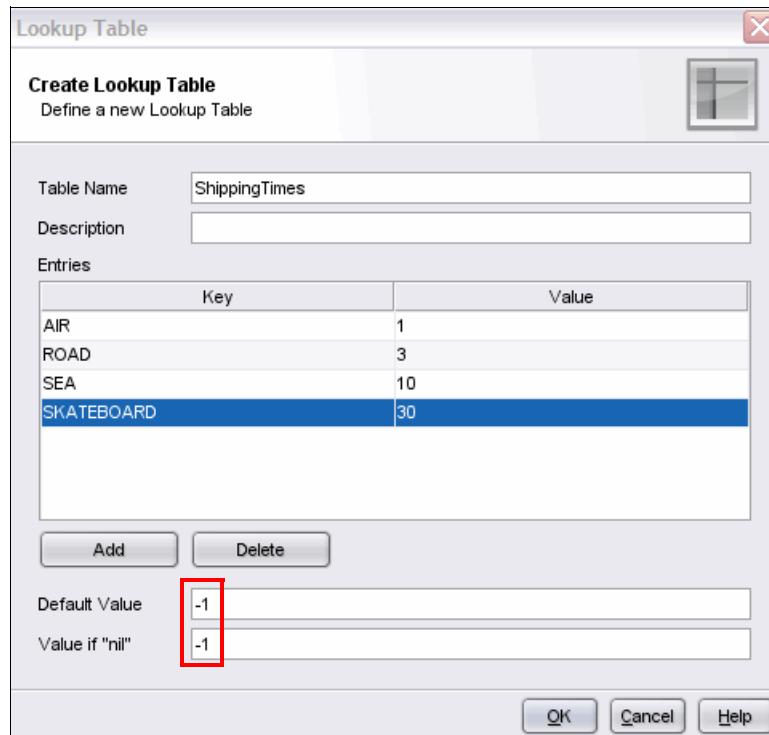


Figure 12-36 Setting up the lookup table

4. Click **OK** to save the lookup table.

12.10 Building the orchestration

Because you now created all the entities that are used by the orchestration, you can now build the `FTP_DB2_to_Domino_sync_Orders` orchestration. To build the orchestration use the following steps.

1. 12.10.1, “Creating job keys” on page 489
2. 12.10.2, “Polling the FTP directory” on page 490
3. 12.10.3, “Creating and renaming orchestration variables” on page 491
4. 12.10.4, “Parsing the input data” on page 493
5. 12.10.5, “Transforming the data with an XML stylesheet” on page 495
6. 12.10.6, “Adding a Lookup activity” on page 497
7. 12.10.7, “Mapping outputs for the Lookup activity” on page 499

8. 12.10.8, “Adding a Merge activity” on page 502
9. 12.10.9, “Adding a For Each activity” on page 504
10. 12.10.10, “Adding the Create Documents activity” on page 505
11. 12.10.11, “Logging the job keys using the Create Job Keys activity” on page 511
12. 12.10.12, “Testing the orchestration” on page 513
13. 12.10.13, “Adding error handling” on page 513

12.10.1 Creating job keys

For this scenario, job keys log the file name that was processed and how many orders each file contained for every job instance. To create the job keys:

1. Select the Project tab, and double-click the **FTP_DB2_to_Domino_sync_Orders** orchestration, if it is not already open.

The Orchestration Properties window allows you to create job keys. You can set orchestration properties in the Configuration pane at the bottom of Studio. Because you are editing a new orchestration, the orchestration properties are already displayed. If the orchestration properties are not visible, click the green circle that starts the orchestration to make them visible.

2. Click **Add** twice to add two job keys in the Job Identification section.
3. Update the name and type of the job keys so that they match the details shown in Table 12-6.

Table 12-6 Job keys

Name	Type
fileNameProcessed	string
totalOrders	decimal

The completed Job Identification section is shown in Figure 12-37.

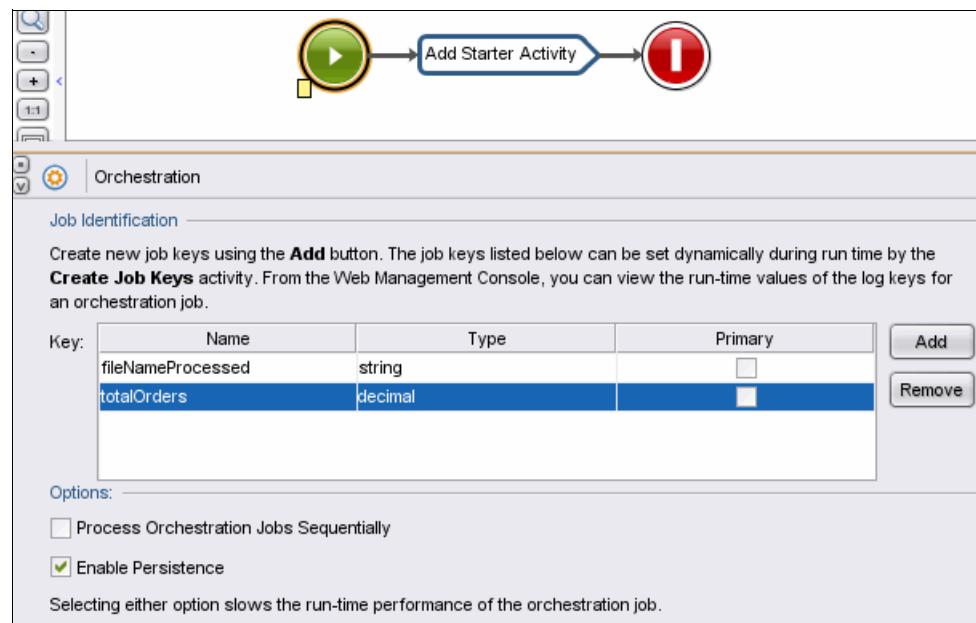


Figure 12-37 Entering job keys

12.10.2 Polling the FTP directory

This orchestration starts when a file is detected in a directory on the FTP server. Thus, because the orchestration polls an FTP server, you must include the FTP Poll Directory as a starter activity. To include an FTP Poll Directory activity:

1. Select **FTP → Poll Directory** from the Activities tab, and drag the activity into the orchestration, as shown in Figure 12-38.

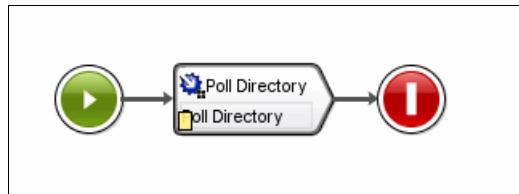


Figure 12-38 Started activity added to the *FTP_DB2_to_Domino_sync_Orders* orchestration

2. From the Checklist pane, click **Summary**, and rename this activity to **Read Orders File over FTP**.
3. From the Checklist pane, click **Pick Endpoint**. Click **Browse**, and select **FTP**, and then click **OK** to bind the FTP endpoint that you created earlier in 12.9.1, “Creating and testing an FTP endpoint” on page 475.
4. From the Checklist pane, click **Configure**:
 - a. Enter ***.txt** for the **File Named** field.
 - b. Create a new configuration property for this field named **FTPPfileName** in the same way you did earlier in 12.9.1, “Creating and testing an FTP endpoint” on page 475.
 - c. For the **In Directory**, click **Browse**, and select the slash (/) field. At this point a connection to the FTP server must be working.
 - d. Create a new configuration property, and name it as **FTPDdirectory**.
 - e. Select **Keep in place** for the **After reading the file** option. With this option selected, the Poll Directory activity will not move or delete the file.

At this point, the Configuration pane looks similar to Figure 12-39 on page 491.

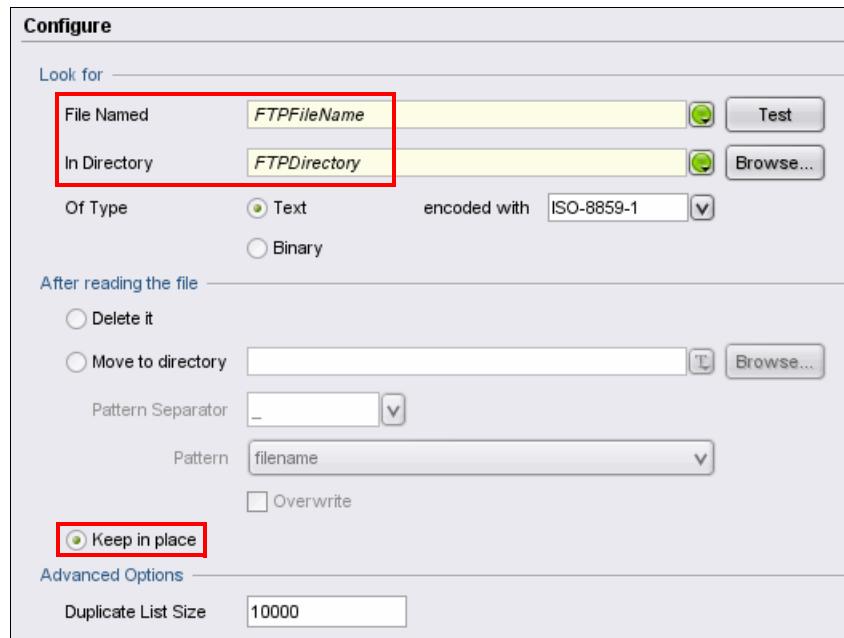


Figure 12-39 Configuring Poll Directory activity

5. In the checklist, select **Configure** → **Delivery Rules**, and change the Polling Interval to 20 seconds.
6. Click **Map Outputs**:
 - a. In the From Activity panel (left side), select **filename**, and click **Copy**.
 - b. In the Copy Parameters window that opens, select **filename** → **Create**.
 - c. Repeat the same steps for the data field.

The Map Outputs pane looks like Figure 12-40.

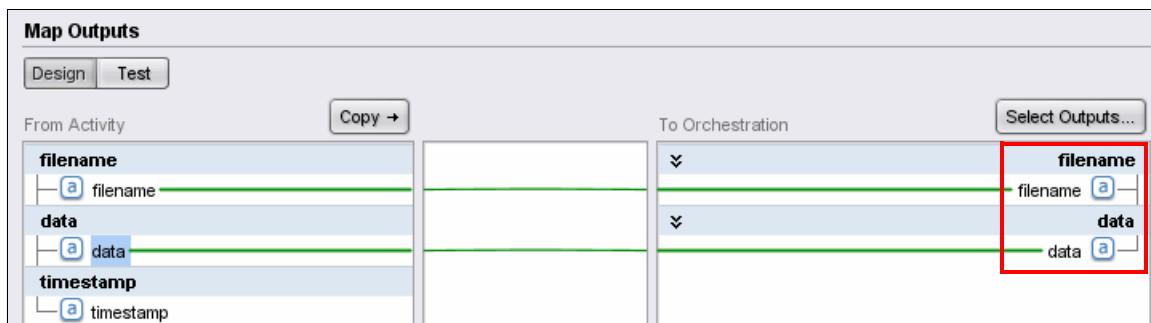


Figure 12-40 Map Outputs from Poll Directory activity

12.10.3 Creating and renaming orchestration variables

The two fields highlighted in Figure 12-40 were created as variables with default names. You must rename them to indicate what they are used for. The orchestration also needs other variables to hold the order data. To create and rename variables:

1. Select the **Variables** tab, and change the variables named **filename** and **data** to **fileNameProcessed** and **ordersFileData** respectively in the Properties panel. Refer to Figure 12-41 on page 492.

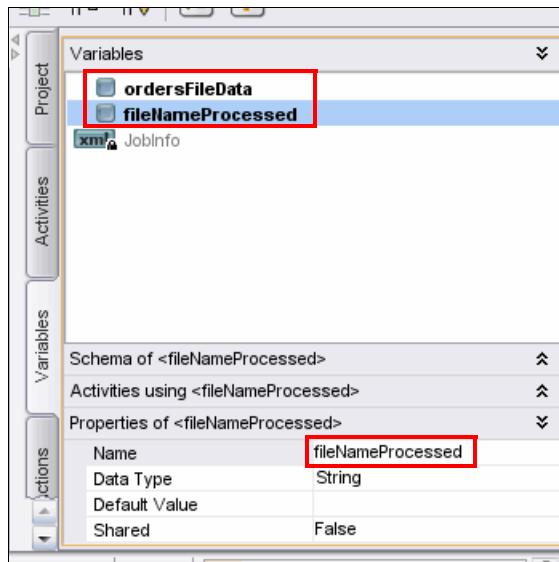


Figure 12-41 Renaming variables

2. While you are working in the Variables tab, create a new variable named totalOrders of type int:
 - a. Right-click anywhere in the Variables panel, and select **Create New Variable**.
 - b. In the window that opens (Figure 12-42), select **int** as the data type, and click **Next**.
 - c. Enter **totalOrders** for the name, and click **Finish**.

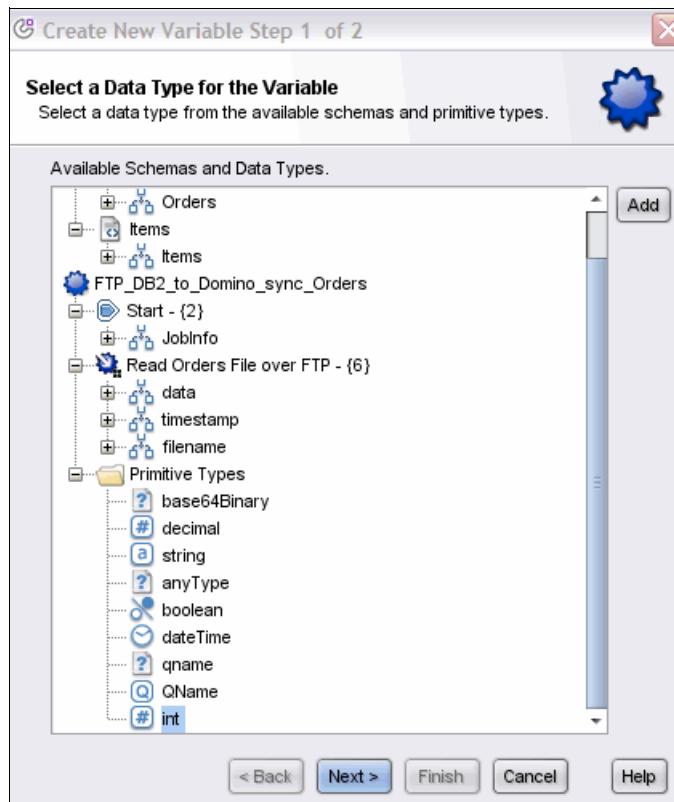


Figure 12-42 Selecting a type for a variable

3. Repeat the previous step to create another variable named `ordersParsedData`. For the data type, select **Flat File Schemas** → **Orders** → **Orders**, as shown in Figure 12-43.

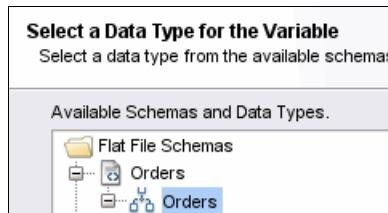


Figure 12-43 Selecting a variable type from a flat file schema type

4. Create another variable of type **Flat File Schemas** → **Items** → **Items**, and name it `allItems`. This variable will be used in a further XSLT step in the orchestration.

Figure 12-44 shows the variables panel at this point.

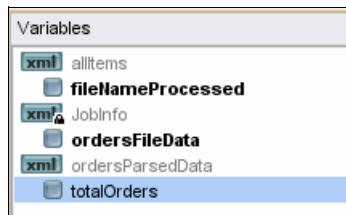


Figure 12-44 Variables panel

5. Save your project.

12.10.4 Parsing the input data

After the Poll Directory identifies and picks the file, the next activity in the orchestration needs to parse the input data. To parse the input data:

1. From the Activities tab, select **Transform** → **Read Flat File**. Drag it to the right of the Poll Directory activity. Right-click the activity, and select **Rename** to rename it to `Parse Orders File`. Refer to Figure 12-45.

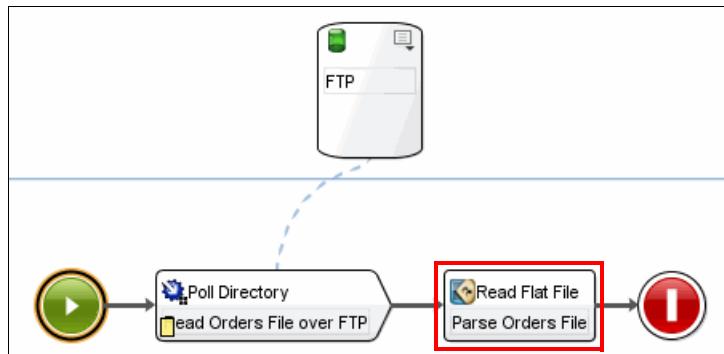


Figure 12-45 Read Flat File activity

2. In the Checklist, select **Configure**, and click **Browse** for Select a Flat File Schema. Select **Orders** from the Project Explorer window.
3. In the Checklist, select **Map Inputs**, click **Select Inputs**, and select `ordersFileData` from the Select Inputs window.

4. Drag **orderFileData** from the left panel to Data on the right panel, as shown in Figure 12-46.

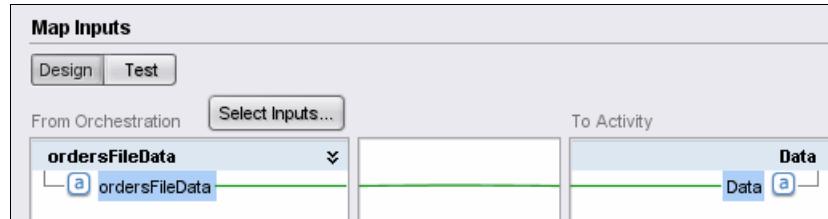


Figure 12-46 Mapping data in the Map Inputs

5. In the Checklist, select **Map Outputs**.
6. Click **Select Outputs**, and select the **totalOrders** variable.
7. Repeat the previous step, this time selecting **ordersParsedData**.
8. Right-click anywhere inside the From Activity panel, and select **Show Additional Property Nodes**, as shown in Figure 12-47:
 - In the From Activity panel on the left, drag occurrence to totalOrders in the To Orchestration panel on the right. This step updates the totalOrders job key with the number of total orders read from the input file.
 - Right-click anywhere inside the From Activity panel, and select **Hide Additional Property Nodes**.



Figure 12-47 Enabling detailed view

9. Drag **Orders** → **Order** on the left (recurring element) to **ordersParsedData** → **Orders** → **Order** (recurring element) on the right side, as illustrated in Figure 12-48 on page 495.

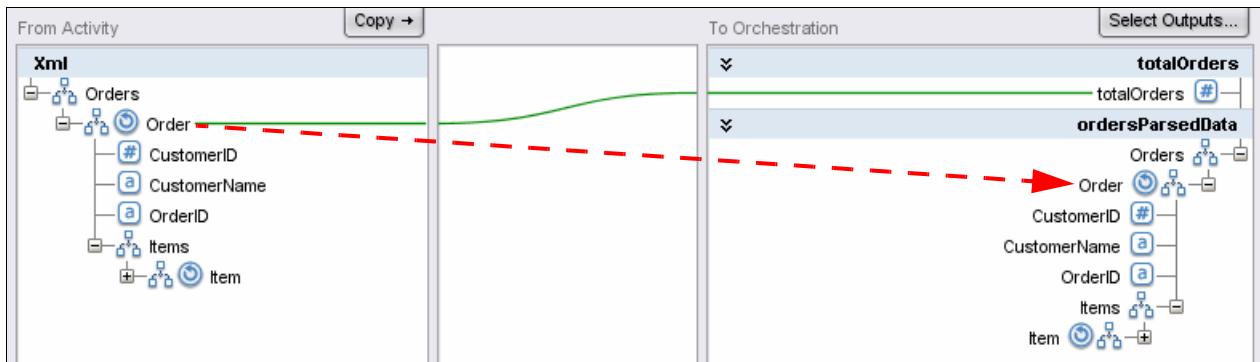


Figure 12-48 Dragging Order to ordersParsedData

The Map Outputs pane now looks like Figure 12-49.

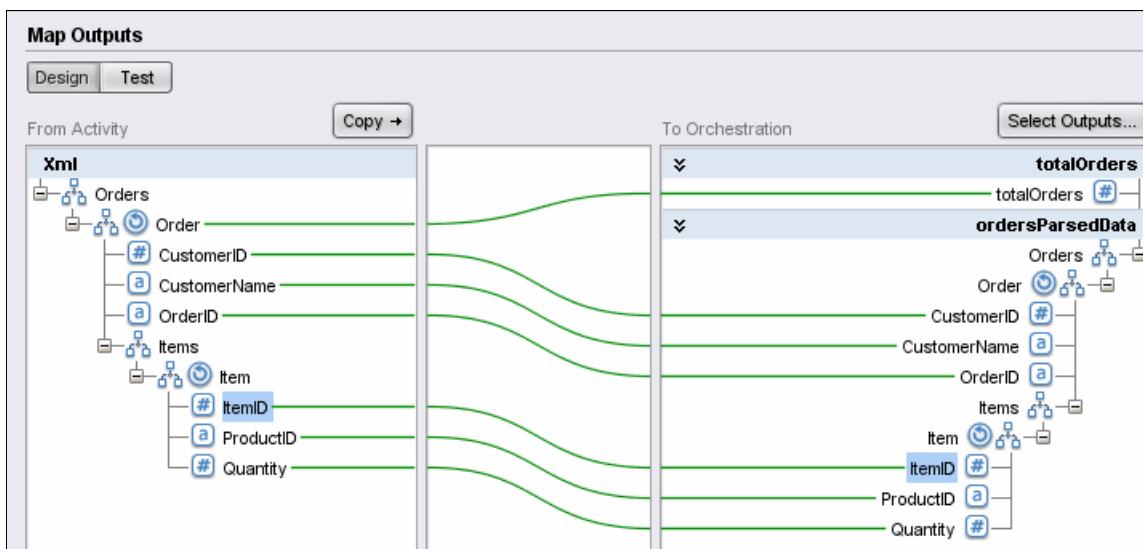


Figure 12-49 Map Outputs from Read Flat File

10. Save your project.

12.10.5 Transforming the data with an XML stylesheet

In this step, you add the Apply XSLT activity to the orchestration. This activity gets the orders that are read in the previous step and transforms the orders into a list of items. This transformation simplifies much of the work that must be done for this orchestration. You will load the scenario3.xslt file, shown in Example 12-5, into your project.

Example 12-5 The scenario3.xslt file

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
<xsl:template match="/">
<Items>
    <xsl:apply-templates select="/Orders/Order/Items/Item"/>
</Items>
</xsl:template>

```

```

<xsl:template match="/Orders/Order/Items/Item">
  <Item>
    <CustomerID><xsl:value-of select="../../CustomerID"/></CustomerID>
    <CustomerName><xsl:value-of select="../../CustomerName"/></CustomerName>
    <OrderID><xsl:value-of select="../../OrderID"/></OrderID>
    <ItemID><xsl:value-of select="ItemID"/></ItemID>
    <ProductID><xsl:value-of select="ProductID"/></ProductID>
    <Quantity><xsl:value-of select="Quantity"/></Quantity>
  </Item>
</xsl:template>
</xsl:stylesheet>

```

To add the Apply XSLT activity:

1. From the Projects tab, right-click **Stylesheets** → **Add Document**. Browse and select the scenario3.xslt file.
2. Select the Activities tab, and then drag **Transform** → **Apply XSLT** to the right of the Read Flat File activity.
3. Rename this activity to Transform Orders to List of Items.
4. In the Checklist, select **Pick Stylesheet**. Click **Browse**, and select the scenario3.xslt file that you added earlier.
5. In the Checklist, select **Set Input & Output**. For the input panel, click **Browse**, and select **ordersParsedData**.
6. For the output panel on the right, click **Browse**, and select **allItems**.

The Apply XSLT settings now looks like Figure 12-50.

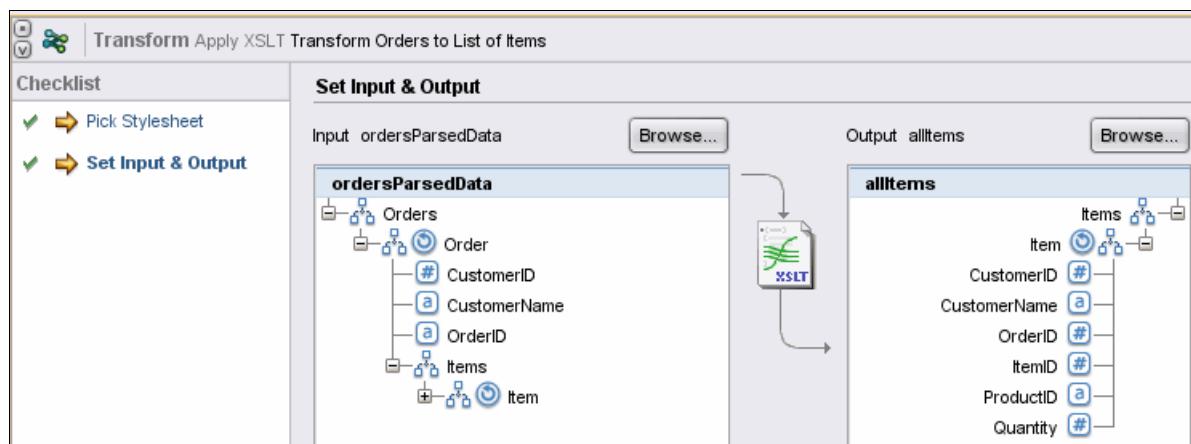


Figure 12-50 Apply XSLT settings

At this point, the orchestration looks similar to Figure 12-51 on page 497.

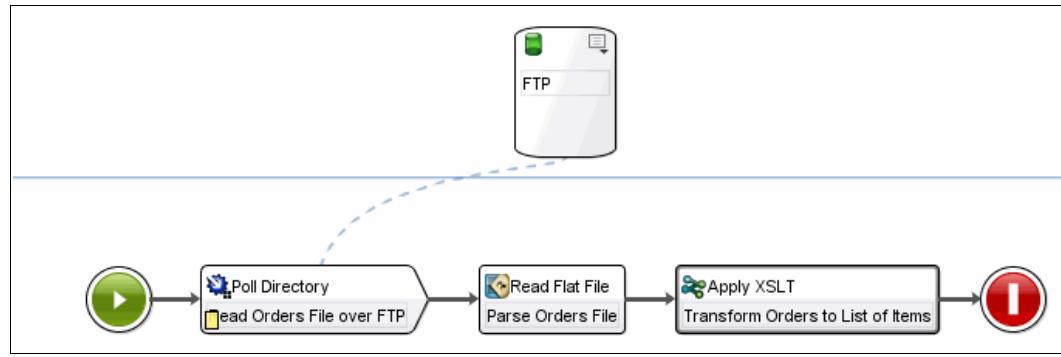


Figure 12-51 Orchestration with the Apply XSLT activity

7. Save your project.

12.10.6 Adding a Lookup activity

The Apply XSLT activity populated a list of all items from all orders. Now, you must add a Lookup activity that searches for complementary information for each item in the product database (in DB2). To add a Lookup activity:

1. Drag **Data Quality** → **Lookup** from the Activities tab to the right of the Apply XSLT activity. Rename it to **Lookup Products Data in Database**. Refer to Figure 12-52.

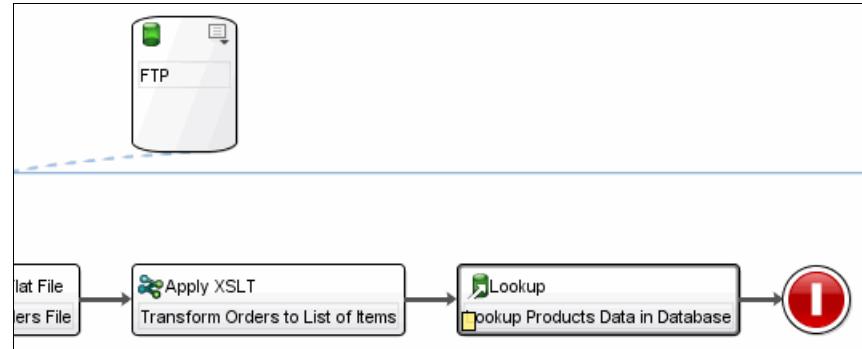


Figure 12-52 Lookup activity

2. From the Checklist pane on the bottom left, click **Pick Endpoint** → **Browse**. Select the endpoint named DB2 that you created earlier.
3. Click **Configure**:
 - a. From the Variable Name drop-down menu, select **allItems**.
 - b. For the Element Name field, click the ellipsis, and select **Items** → **item**. Refer to Figure 12-53 on page 498.

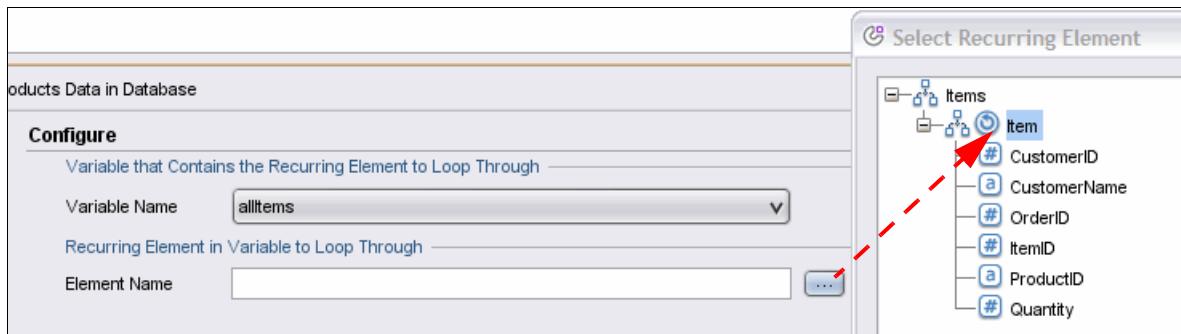


Figure 12-53 Configure settings

4. Each item contains a product ID that is used as a key to find a description, price and transport method for the product in the database. Select **Configure** → **Enter Query**, and type the SQL query from Example 12-6.

Example 12-6 Query products table

```
SELECT DESC, PRICE, TRANSPORT FROM PRODUCTS WHERE ID=?
```

5. Validate the SQL query by clicking the **Validate Query** button, shown in Figure 12-54.

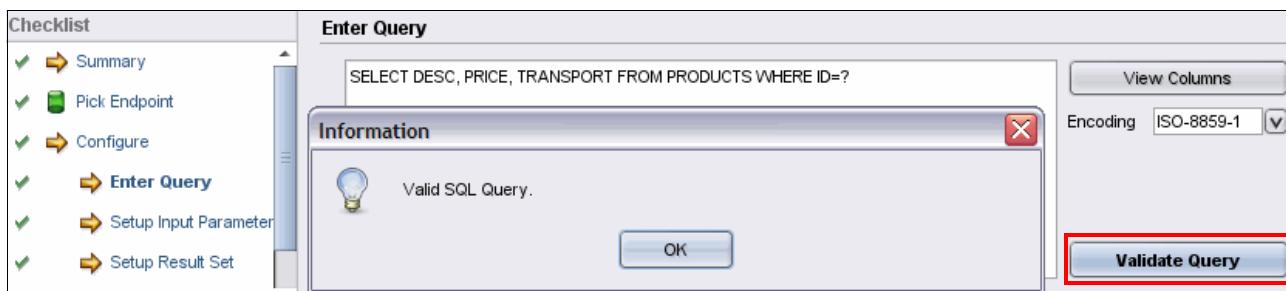


Figure 12-54 Validating the SQL query

6. Still in the Checklist, select **Setup Input Parameters**:

- a. Scroll to the right in the table until you see a column named XPath.
- b. Click the cell to select it, and then click the ellipsis button.
- c. A Select XPath window opens, as shown in Figure 12-55.
- d. Select **item** → **ProductID**, and click **OK**.

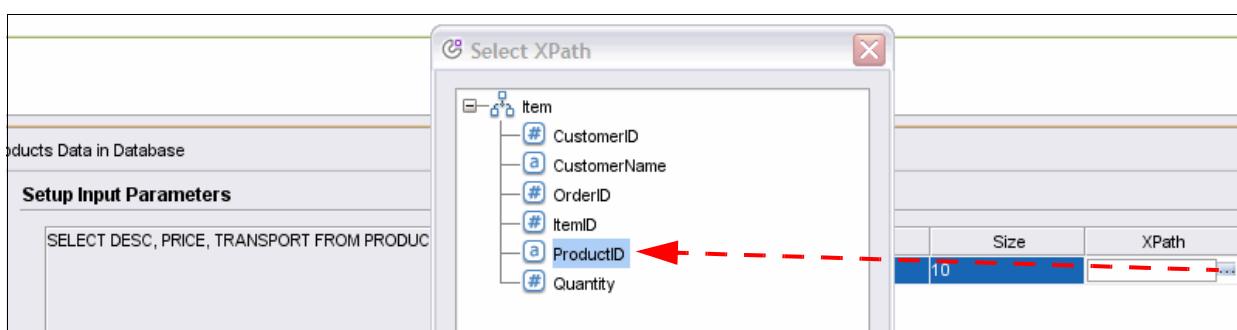


Figure 12-55 Selecting ProductID from a XPath selection

Notice that after you validate the query, selecting **Setup Result Set** in the checklist shows the database details of the following fields that are part of query:

- ▶ Description
- ▶ Price
- ▶ Transport

The Lookup activity reads data from the database intelligently. Thus, it batches and caches data for optimization. Selecting **Delivery Rules** in the checklist shows advanced configurations for this activity, such as the Preload cache and Cache at most options. To learn more about these configurations, visit the information center at:

http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/topic/com.ibm.websphere.cast_iron.doc/use_the_lookup_activity.html

12.10.7 Mapping outputs for the Lookup activity

The Lookup activity returns the following data structures:

- ▶ goodXML
- ▶ badXML

The goodXML structure contains data that was retrieved successfully from the database and includes the original fields for each item and the product description, price, and transport type.

The badXML structure lists the items whose product was not found in the database and does not include the product data. These two data structures are quite useful because you can use them to identify easily which product searches failed and which succeeded.

Next map these output parameters to two new variables that are used by the rest of the orchestration. To map outputs:

1. Before you define the mapping for the Lookup activity, create two new variables. They are itemsFound and itemsNotFound both of type Items. Go to the Variables tab, and right-click to Create a New Variable.
2. Select the **Items** structure under Lookup Products Data in Database node (see Figure 12-56 on page 500). Click **Next**, and name it itemsFound, and then click **Finish**.
3. Repeat the previous step, but this time create a variable named itemsNotFound.

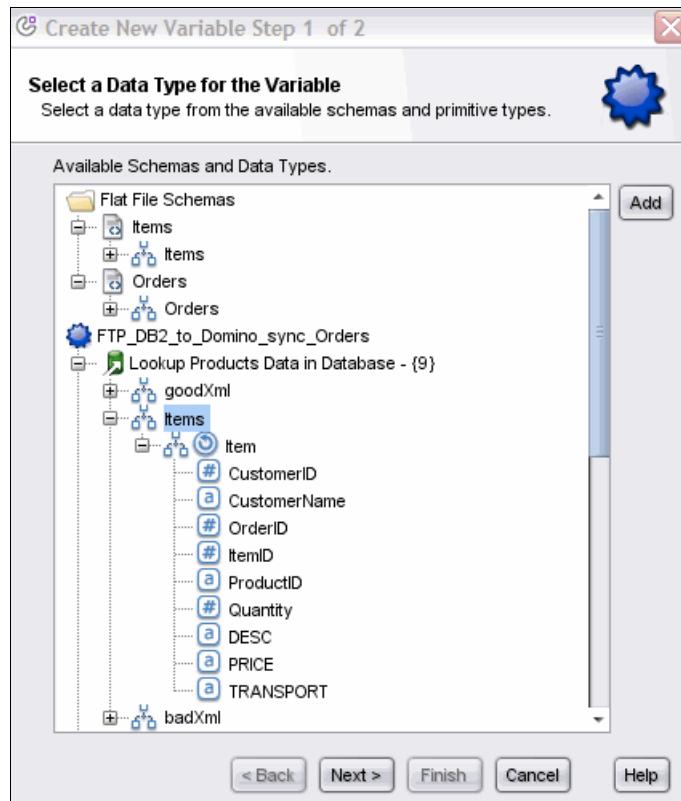


Figure 12-56 Creating itemsFound variable

4. Click the **Lookup Products Data in Database** activity in the orchestration.
 5. In the Map Outputs panel, click **Select Outputs**. Holding the Control key, select both **itemsFound** and **itemsNotFound**. Click **OK**. Now these two variables are loaded into the To Orchestration column.
- Tip:** Sometimes, the From Activity pane of the Lookup activity Map Outputs task does not populate properly. If this happens, restart Studio.
6. Drag **goodXML** → **items** → **item** onto **itemsFound** → **items** → **item**. Drag **badXML** → **items** → **item** onto **itemsNotFound** → **items** → **item**. Refer to Figure 12-57 on page 501.



Figure 12-57 Map outputs for Lookup activity

Because **itemsNotFound** was mapped from **badXML**, which contains items where no data was found in the database, you must provide default values for the **DESC**, **PRICE**, and **TRANSPORT** fields in the **itemsNotFound** variable.

7. In the To Orchestration pane on the right, right-click **itemsNotFound** → **DESC**, and select **Define Default Value** from the menu. Type **ERROR: PRODUCT ORDERED WAS INVALID** as the default value.
8. Repeat this step to add default values for **itemsNotFound** → **PRICE** and **itemsNotFound** → **TRANSPORT**. For **PRICE** set **0** (zero) as default and for **TRANSPORT** set **ERROR** for the default value.

In the mapping pane, these three fields now are presented with a big D (default) on the left side. See Figure 12-58.

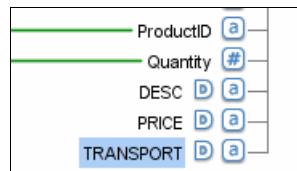


Figure 12-58 Default values assigned for desc, price, and transport

In this scenario, we want to create invoices even if something is wrong with the input data, more specifically if product details cannot be found in the database. This is the reason you used default values for items with products not found in the Lookup activity.

12.10.8 Adding a Merge activity

Because you must create invoices with both complete items and items that returned errors, add a Merge activity at this point to merge both items found and items not found lists. Follow these steps:

1. From the Activities tab drag **Data Quality** → **Merge** to the right of the Lookup activity. Rename it to Merge Good Data and Error Data. Refer to Figure 12-59.

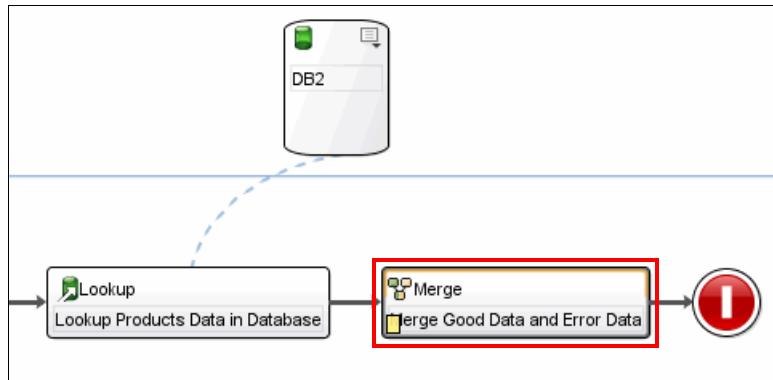


Figure 12-59 Merge activity

2. This activity merges itemsFound and itemsNotFound variables. In the Checklist pane, on the left, click **Configure**:
 - a. In the Left Variable that Contains the Recurring Element to Loop Through:
 - i. Click the drop-down in the **Variable Name** field, and select **itemsFound**.
 - ii. Click the ellipsis to the right of the **Element Name** field, and select **Items** → **Item**.
 - b. In the Right Variable that Contains the Recurring Element to Loop Through:
 - i. Click the drop-down in the **Variable Name** field, and select **itemsNotFound**.
 - ii. Click the ellipsis to the right of the **Element Name** field, and select **Items** → **Item**.

The Configure panel should look like Figure 12-60.

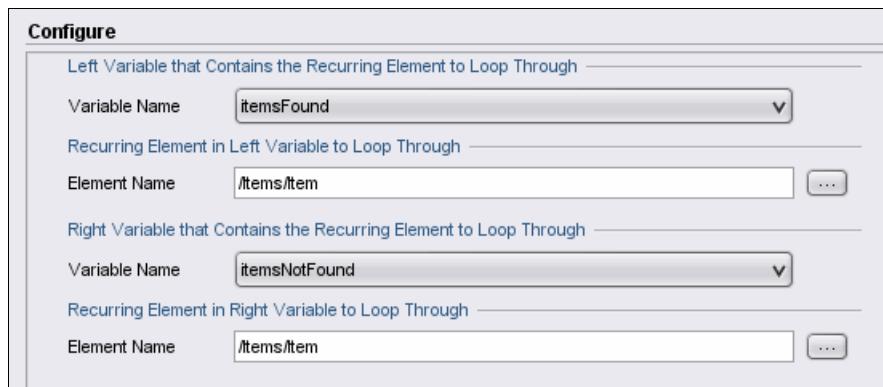


Figure 12-60 Merge activity configure panel

3. Select **Configure** → **Merge Properties**:
 - a. In the drop-down menu for the Merge Type field, select **Merge and keep left duplicates**.
 - b. Include three **Left Sort Keys**. Click **Add** three times.

- c. Change the **XPath** value for each row you just added. Click the ellipsis button inside the first cell in the XPath column, and select CustomerID. For the XPath values for the other two lines, select OrderID and ItemID respectively.
- d. Repeat the same steps for the Right Sort Keys table. The final result is presented in Figure 12-61.

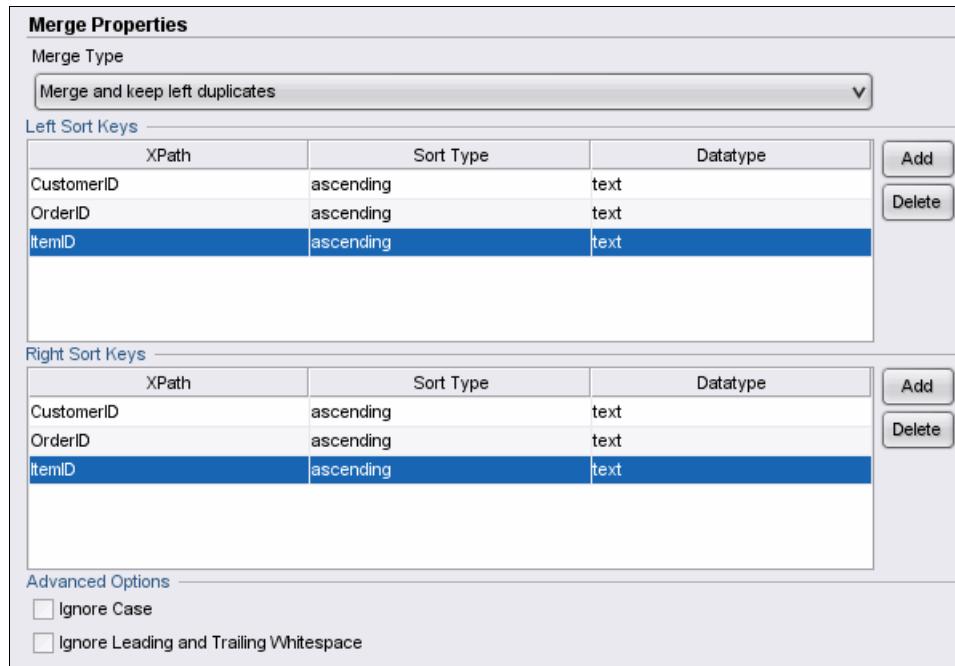


Figure 12-61 Merge Properties configured with three keys

4. Select **Map Outputs**. Click **Copy**, and select **mergedXML** in the Copy Parameters window. Click **OK**, as shown in Figure 12-62.

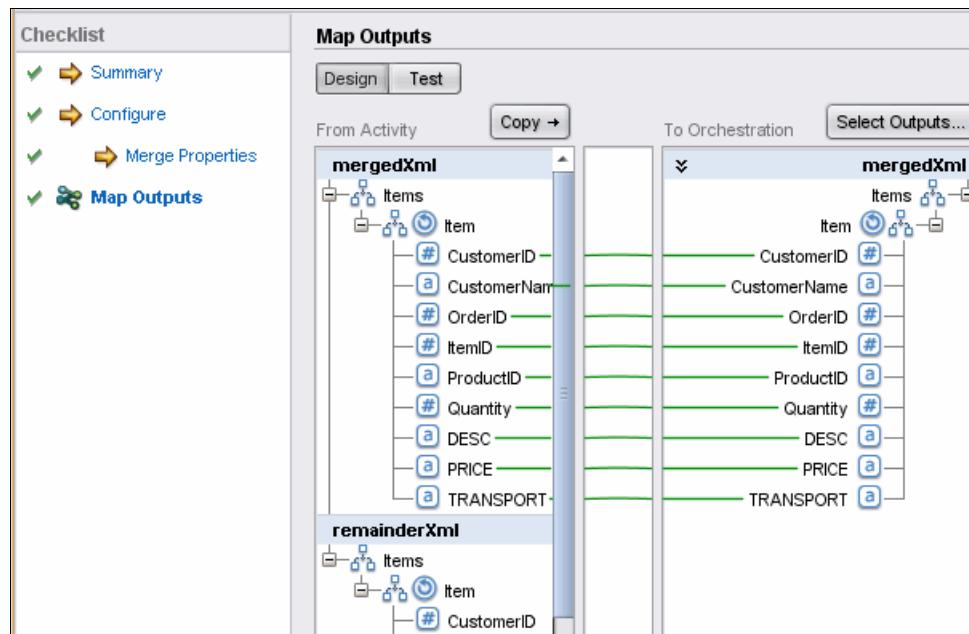


Figure 12-62 Map Outputs panel from Merge activity

5. Now, give a proper name to the variable copied in the Map Outputs panel. Select the Variables tab. Find and select the variable named **mergedXML**. Rename it to **itemsMergedData** in the properties panel located at the bottom, as shown in Figure 12-63.

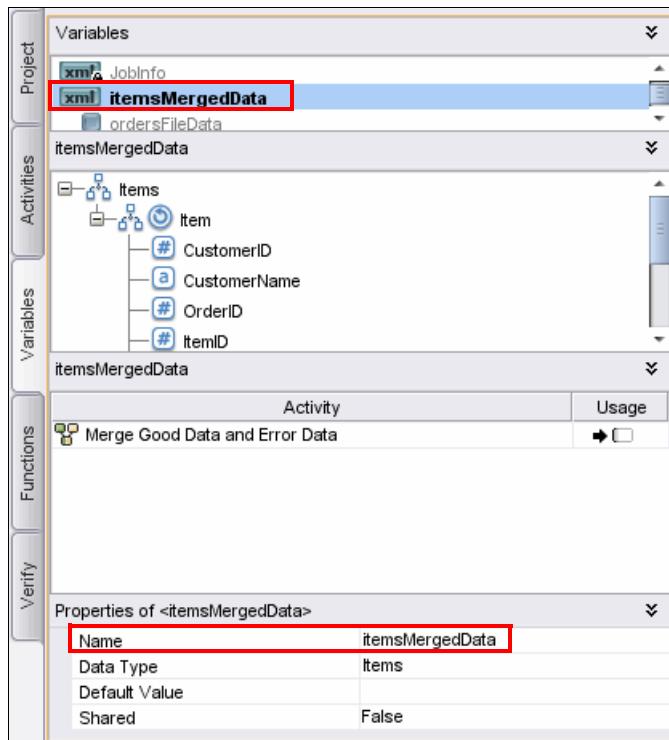


Figure 12-63 Renaming the mergeXML variable to itemsMergedData

All the activities using this variable are updated automatically.

6. Save your project.

12.10.9 Adding a For Each activity

Next, you must add a For Each activity, which iterates over each item element from the itemsMergedData variable that was set in the Merge activity. To add a For Each activity:

1. Select the Activities tab, and drag **Logic → For Each** to the right side of the Merge activity. Make sure that you rename it to Loop through items, as shown in Figure 12-64:
 - a. In the drop-down menu for the Variable that Contains the Recurring Element to Loop through Variable Name field, select **itemsMergedData**.

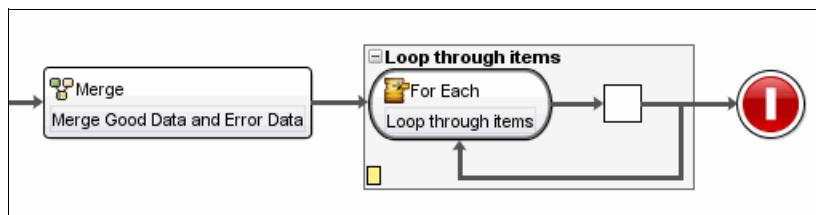


Figure 12-64 For Each activity

2. Click the ellipsis button to select a value for the Element Name field. In the Select Recurring Element window, select **Items** → **Item**, and then click **OK**.

3. Enter **item** for the Variable Name field in the Single Element Returned by the Loop section. Figure 12-65 shows the For Each settings.

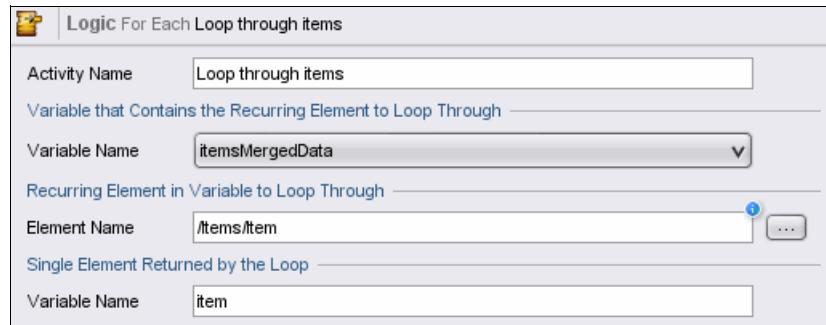


Figure 12-65 For Each settings

4. Save your project.

12.10.10 Adding the Create Documents activity

After the For Each activity is set, add the Create Documents activity inside the For Each loop. These two activities combined insert new documents into the Domino database that you configured in 12.8.3, “Preparing the Domino database” on page 474.

To add this activity:

1. From the Activities tab, drag **Domino → Create Documents** inside the For Each block in the small empty square space. Rename it to **Create Invoice Documents in Domino**, as shown in Figure 12-66.

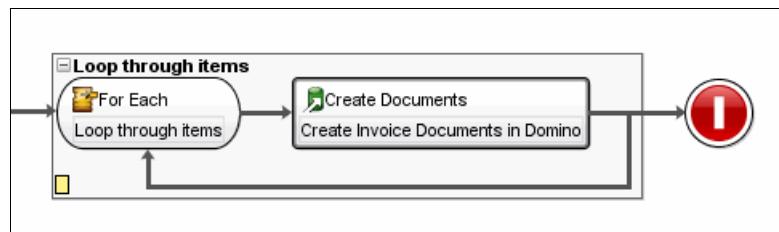


Figure 12-66 Inserting Create Documents activity

2. In the Checklist pane, select **Pick Endpoints** → **Browse**. Select the Domino endpoint you created earlier.
3. Select **Configure** → **Browse** to select the Domino database where this orchestration will insert the documents. After the Domino database is selected, you must also select the Order form. Click **Finish**.

Figure 12-67 on page 506 presents the result with these two fields selected. Notice that the two fields, Database and Form, are pulled from the Domino server, so when you click **Browse**, a list of databases and forms are displayed that are pulled from the Domino Server.



Figure 12-67 Selecting Domino database and form

4. Select **Map Inputs** in the Checklist pane. Click **Select Inputs**. Select the variable item from the Select Inputs window, and click **OK**, as shown in Figure 12-68.

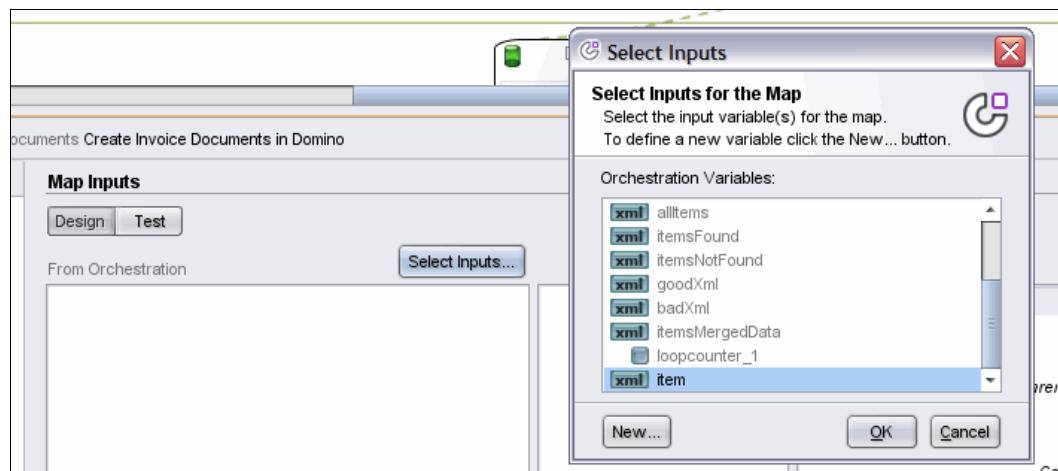


Figure 12-68 Selecting input variable

The data structure on the right side panel (To Activity) was defined when you selected the form in the previous step.

5. You will now start an extensive mapping. The target fields are highlighted on the bottom of the right panel (To Activity), shown in Figure 12-69.

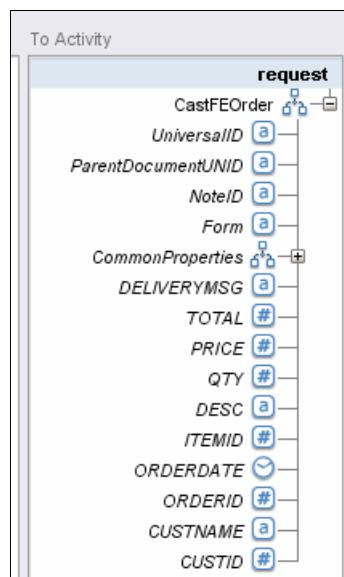


Figure 12-69 Target field from Domino's database form

Hint: Collapse the CommonProperties node to simplify the mapping view. See Figure 12-69.

6. Drag **CustomerID** on the left to CUSTID on the right. Click **Yes** in the warning message window. This message warns of data type mismatch. It does not affect your project. See Figure 12-70. From now on in this Map Inputs panel, click **Yes** for all the warning messages that pops up. Complete these steps on this window:
 - a. Drag **CustomerName** on the left to CUSTNAME on the right.
 - b. Drag **OrderID** on the left to ORDERID on the right.
 - c. Drag **ItemID** on the left to ITEMID on the right.
 - d. Drag **Quantity** on the left to QTY on the right.
 - e. Drag **DESC** on the left to DESC on the right.

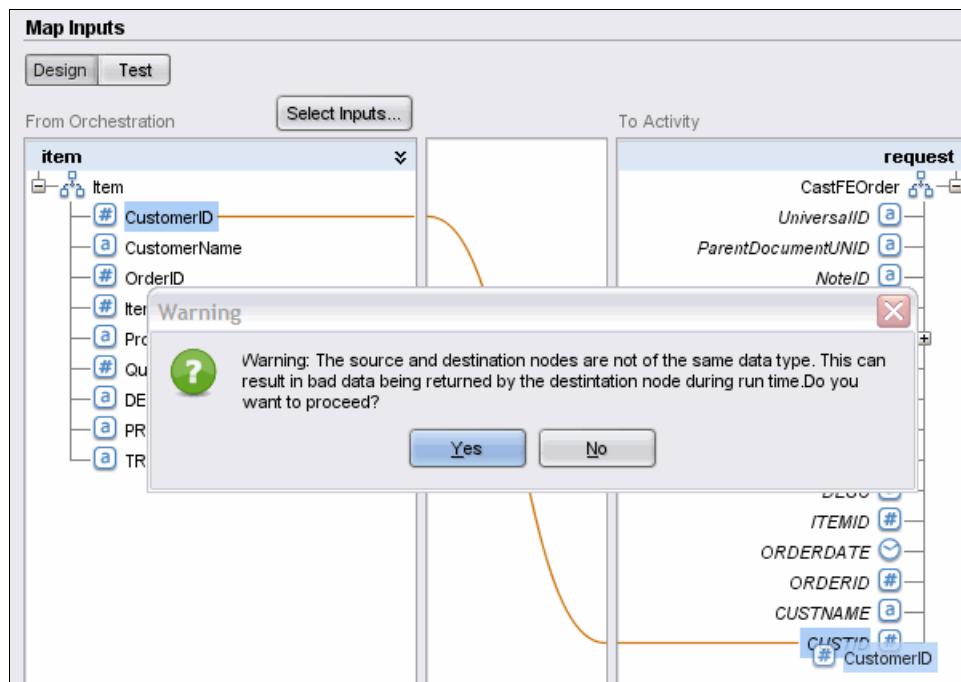


Figure 12-70 The warning message when you map CustomerID to CUSTID

7. Go to the Functions tab, and drag **DateTime** → **Get Current Date** to the central mapping pane. See Figure 12-71 on page 508.

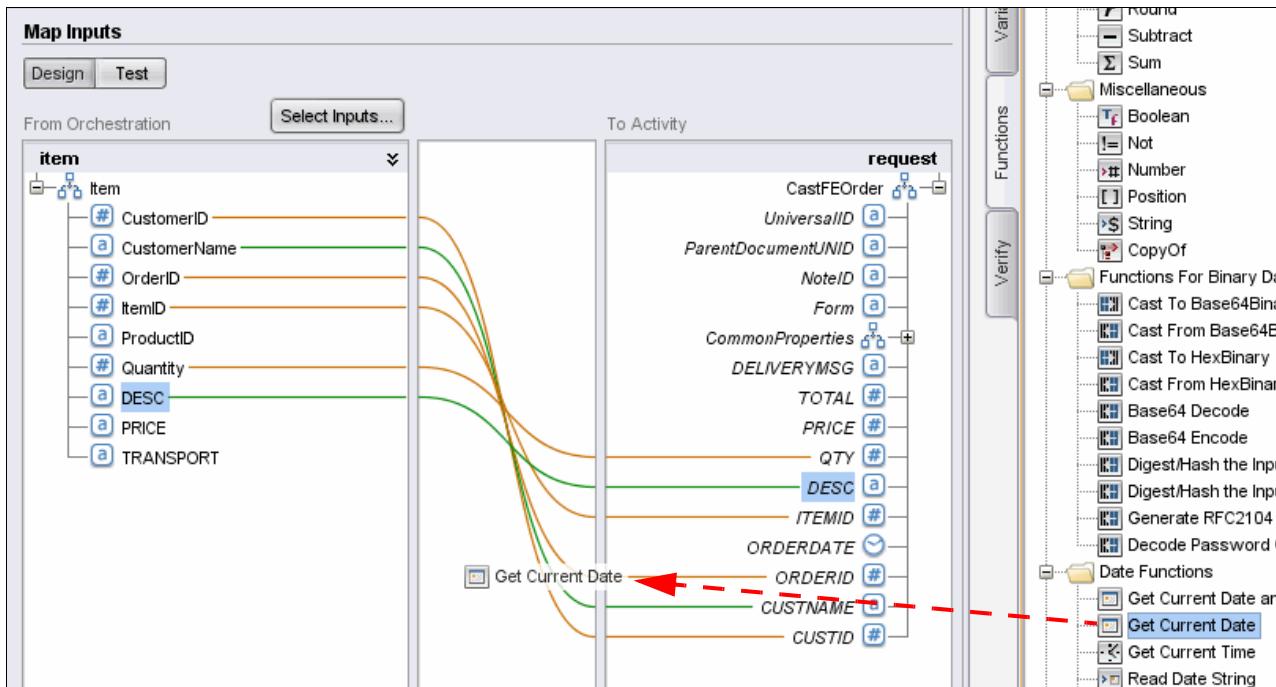


Figure 12-71 Adding Get Current Date to the mapping panel

8. Drag the Get Current Date function to the ORDERDATE in the To Activity column.
9. Right-click the Get Current Date function in the center pane, and select **Apply Function Graph**, as shown in Figure 12-72.

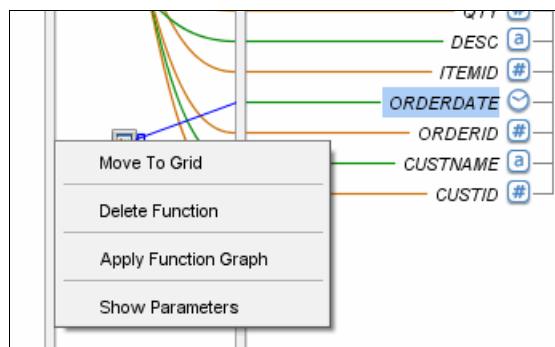


Figure 12-72 Mapping Get Current Date to ORDERDATE

10. The PRICE field you are going to map now is represented as a String. We must convert it to the appropriate data type (Number) before pushing it to the Domino database because in Domino this field is represented as a number:
 - a. Go to the Functions tab, and drag **Miscellaneous** → **Number** to the central mapping pane.
 - b. Drag **PRICE** on the left to the Number function you just added.
 - c. Drag the **Number** function to PRICE on the right. Right-click the **Number** function, and select **Apply Function Graph**. See Figure 12-73 on page 509.

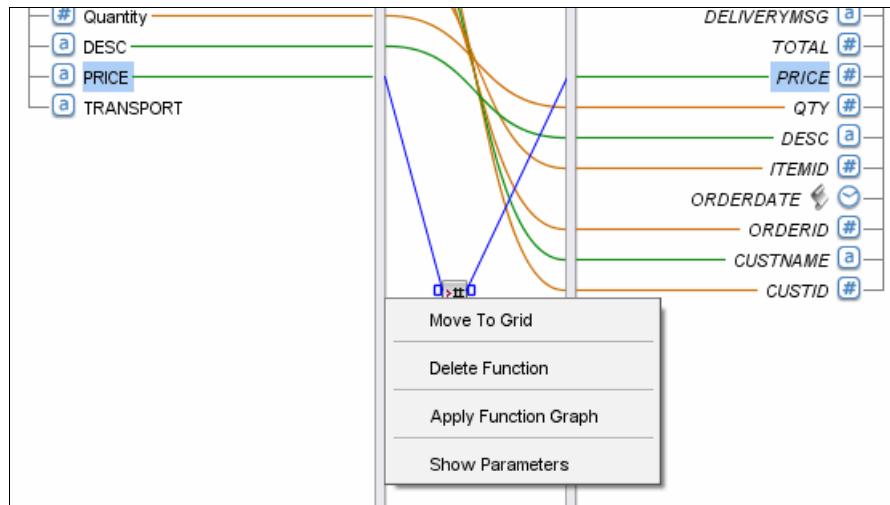


Figure 12-73 Mapping PRICE using Number function.

11. The Total is calculated based on the price of the item times the quantity:
- From the Functions tab, drag **Miscellaneous** → **Number** to the central pane.
 - From the Functions tab, drag **Mathematical** → **Multiply** to the central pane.
 - Drag **PRICE** on the left to the Number function in the central pane.
 - Drag the **Number** function to the Multiply function in the central pane.
 - Drag **Quantity** on the left to the Multiply function you just added.
 - Drag the **Multiply** function to **TOTAL** on the right.
 - Right-click the **Number** function, and select **Apply Function Graph**. See Figure 12-74.

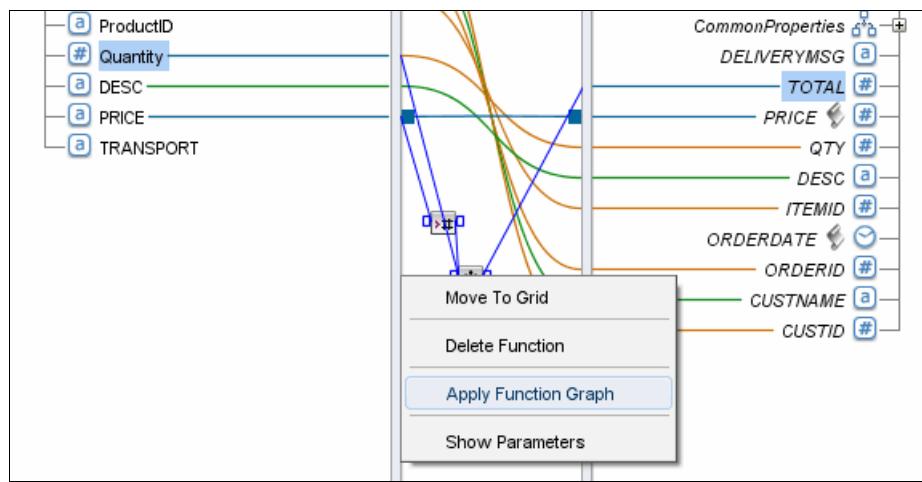


Figure 12-74 Calculating Total

12. The next step is to calculate the estimated delivery time. Remember that the products table contains a transport column that identifies the transportation method for each product. You also created a custom function (createDeliveryDateMessage) in 12.9.6, “Creating a custom function” on page 485 and a Lookup table (ShippingTimes) in 12.9.7, “Creating a lookup table” on page 487.

You use both artifacts to calculate the delivery date message. The lookup table contains the number of days that each transport method will take and the custom function calculates the delivery date based on the current date plus the number of days for the selected transport.

From the Functions tab, drag **Lookup Tables** → **ShippingTimes** to the central pane.

13. From the Functions tab, drag **Custom Functions** → **createDeliveryDateMessage** to the central pane, as shown in Figure 12-75.

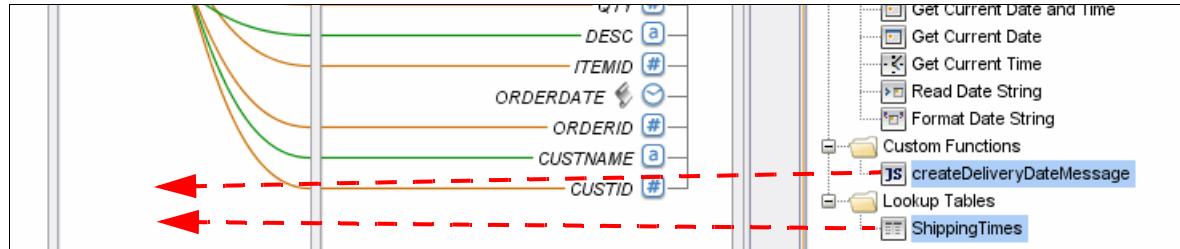


Figure 12-75 Dragging ShippingTimes and createDeliveryMessage

14. Drag **TRANSPORT** on the left to the ShippingTimes lookup table you just added.
15. On the central pane, drag **ShippingTimes** to the **createDeliveryDateMessage** custom function you just added. Click **YES** on the warning message.
16. Drag the **custom function** to **DELIVERYMSG** on the right.
17. Right-click the **custom function**, and select **Apply Function Graph**, as shown in Figure 12-76.

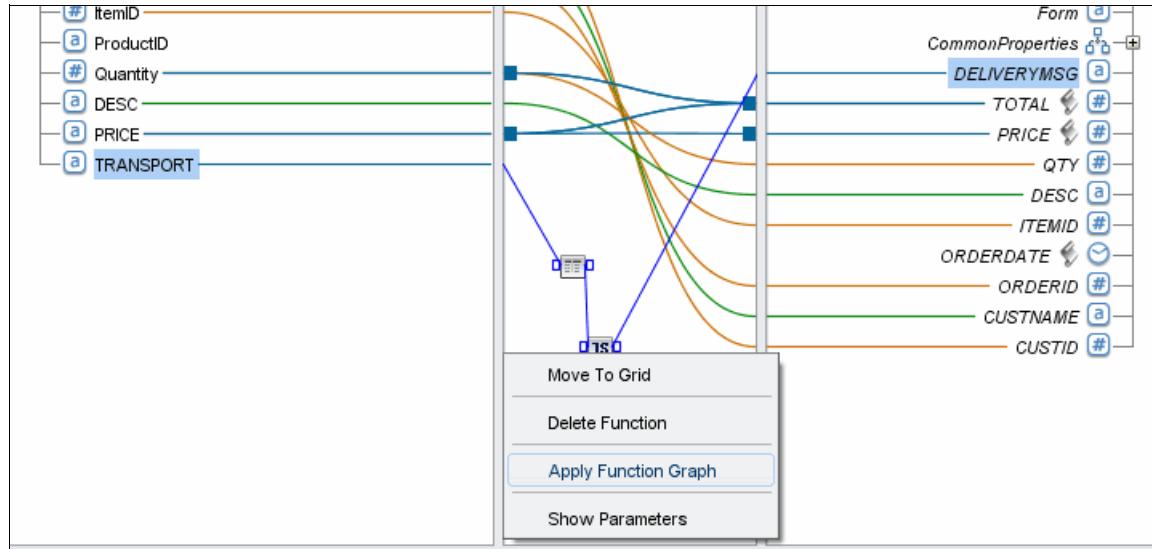


Figure 12-76 Calculating delivery date message

Figure 12-77 shows the final Map Inputs.

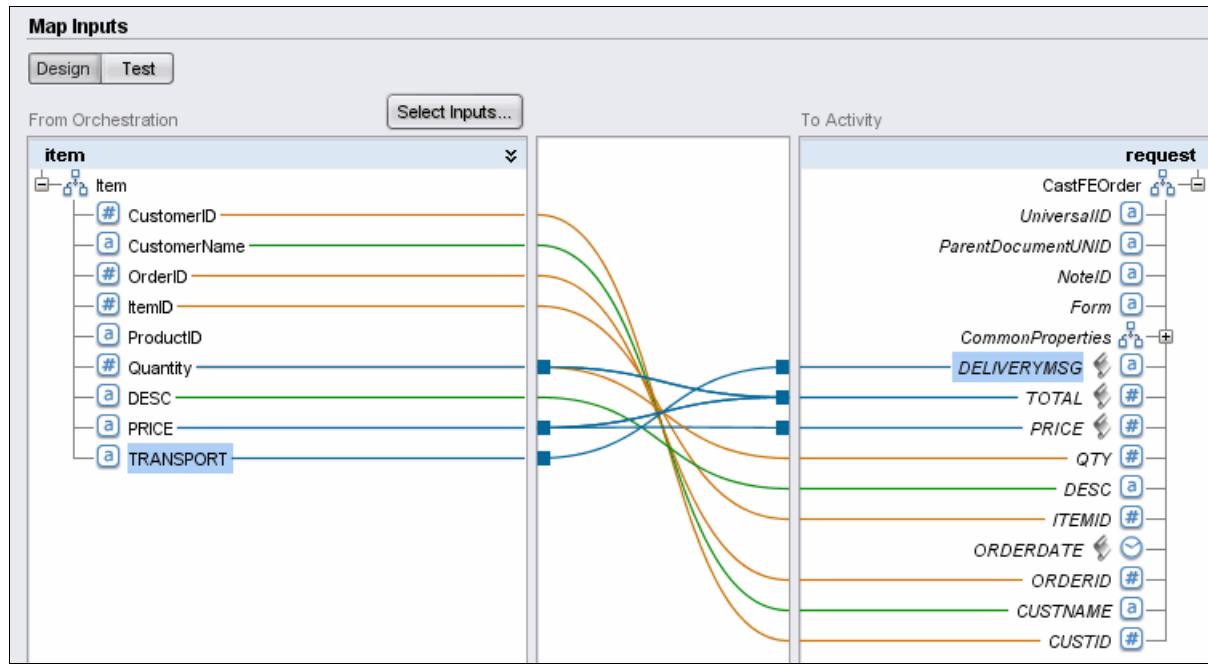


Figure 12-77 Map Inputs

Note: You might notice the mapping lines have different colors. For more information about mapping, see 3.9, “Maps” on page 128.

18. Save your project.

12.10.11 Logging the job keys using the Create Job Keys activity

In the initial steps of your orchestration (12.10.1, “Creating job keys” on page 489), you prepared two orchestration job keys that hold the input file name and the total number of orders. Now it is time to log these two job keys using the Create Job Keys activity.

To log these two job keys using the Create Job Keys activity:

- From the Activities tab, drag **Utilities → Create Job Keys** to the right of and outside the For Each block. Rename it to Record filename and total orders. See Figure 12-78.

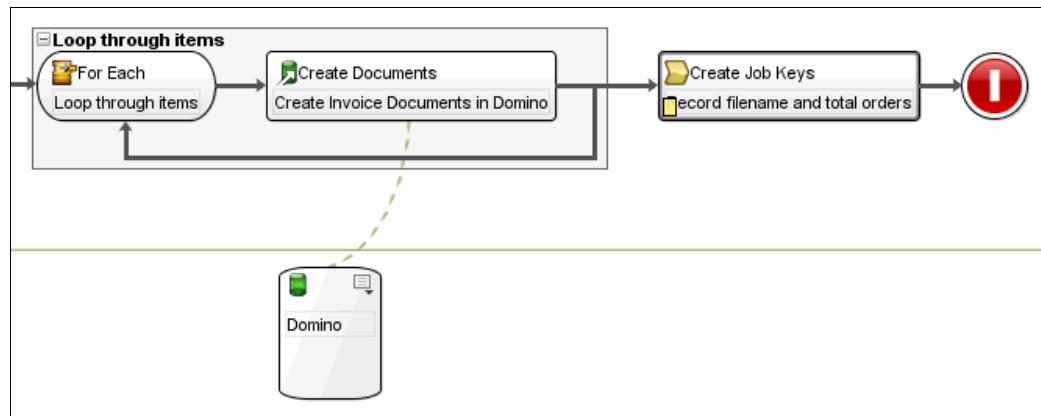


Figure 12-78 Create Job Keys activity

- In the Checklist pane, select **Map Inputs**:

- Click **Select Inputs**. In the Select Inputs window, hold the Control key, and select both **fileNameProcessed** and **totalOrders**. Click **OK**.
- Drag **fileNameProcessed** on the left to **fileNameProcessed** on the right.
- Drag **totalOrders** on the left to **totalOrders** on the right.

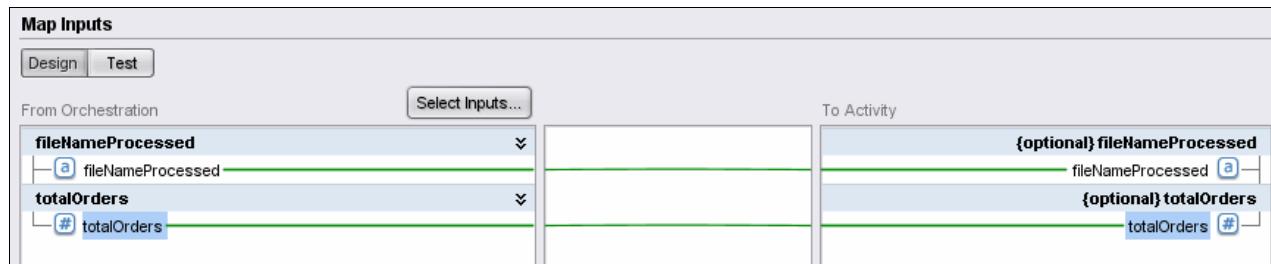


Figure 12-79 Map Inputs for Job Keys

- Save your project.

Figure 12-80 illustrates the orchestration at this point.

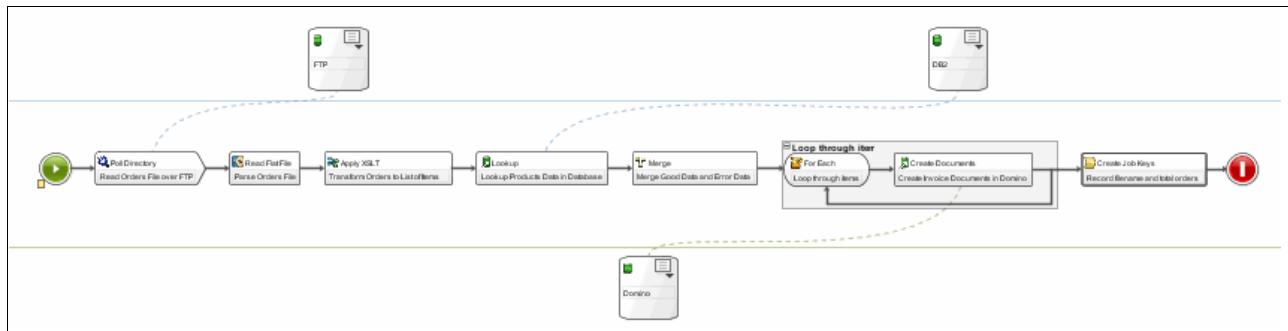


Figure 12-80 Orchestration without error handling

12.10.12 Testing the orchestration

Before you continue developing the orchestration, test the orchestration within Studio.

Important: Remember that you must have a correctly formatted test file in the directory on the FTP server before you can test the orchestration.

To test the orchestration at this point, select the Verify tab, and click the green arrow on the top. The orchestration will run and load detailed information for all activities and the completion status too. See Figure 12-81.

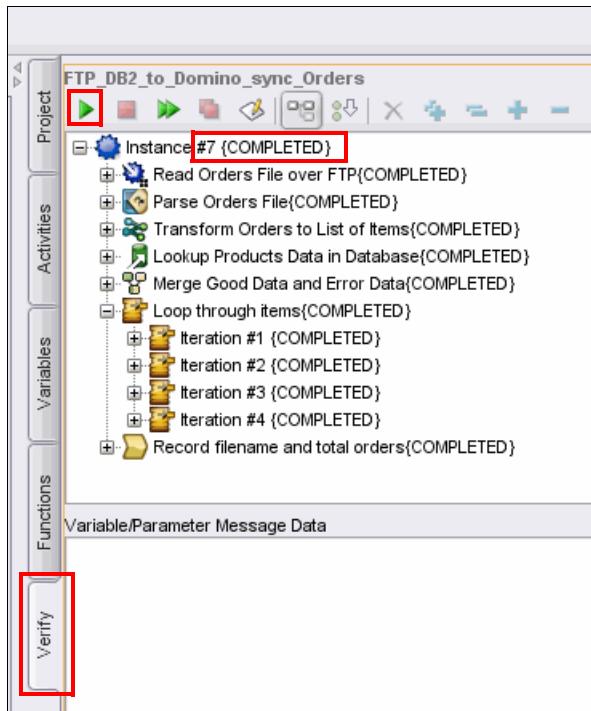


Figure 12-81 Testing the orchestration inside Studio

12.10.13 Adding error handling

9.4, “Using the common error handler TIP” on page 353 explains how to deploy and use the Common Error Handler. 10.5.7, “Using the Common Error Handler” on page 397 also provides an example of how to prepare and use the CommonErrorHandler. This section provides an overview of how the error handler can be added to this orchestration.

This process assumes that the common error handler was deployed and is running on a Cast Iron runtime.

This is a high-level view of how the common error handler is added to this project.

1. The WSDL file for the common error handler web service is downloaded from the Cast Iron runtime and added into the current project.
2. A web services endpoint is created in the project, using the WSDL file.
3. The web services endpoint is used to create an Invoke Service activity.
4. A Try activity is added that encapsulates and handles errors for the activities shown in Figure 12-82 on page 514. The Invoke Service is added to the CatchAll branch.

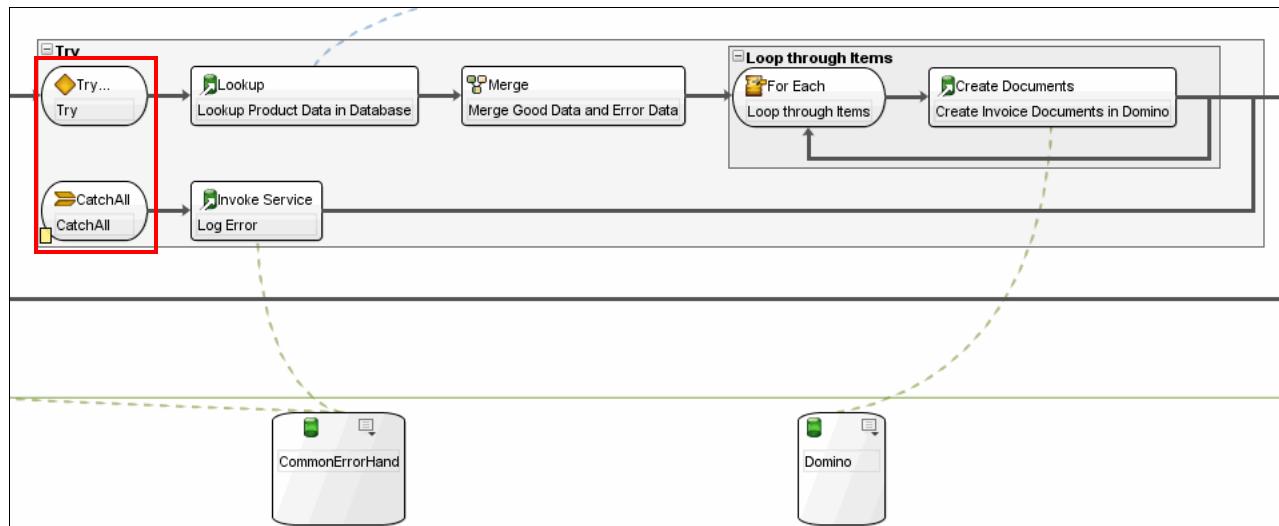


Figure 12-82 Try catch block

5. A general purpose Catch All activity is added for the activities that are not handled by the Try activity. Refer to Figure 12-83.

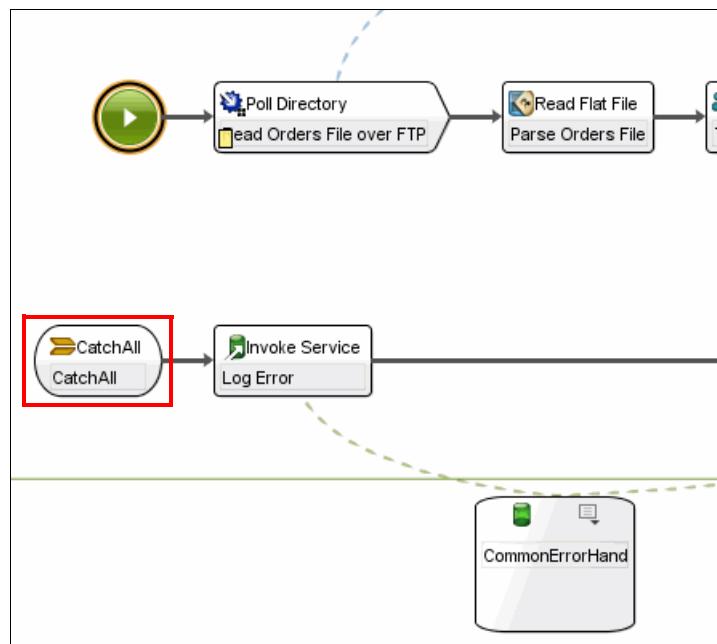


Figure 12-83 Catch All block

Figure 12-84 on page 515 illustrates the complete orchestration.

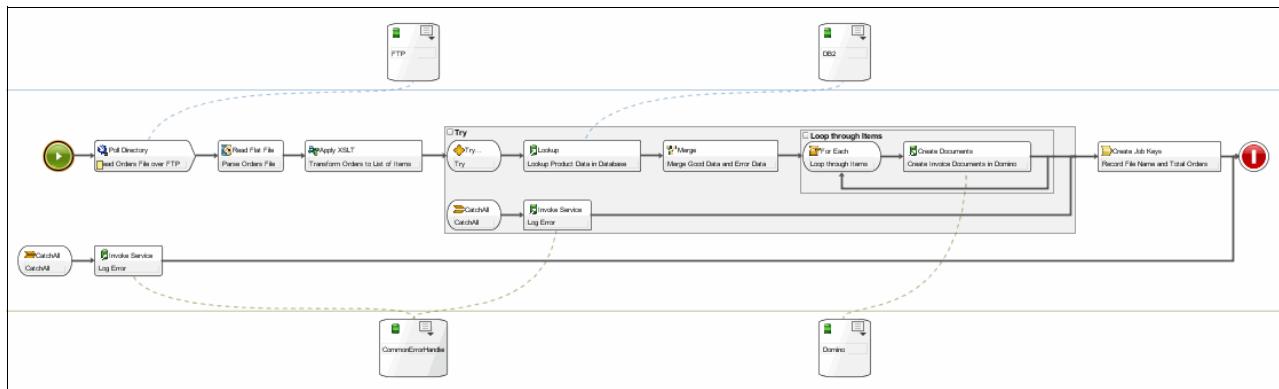


Figure 12-84 The complete orchestration

It is a good idea to test the orchestration again in Studio, forcing error situations to check whether the orchestration and the CommonErrorHandler are working together as expected. In this scenario, an error can be forced by changing the input file data to a different format, or by shutting down the FTP server or the database.

After you test the orchestration in Studio, you can publish and deploy it to a Cast Iron runtime.

Deployment is discussed in 3.13, “Exporting and publishing” on page 170. You can publish the project from Studio directly on the appliance. Alternatively, you can export it as a .par file and upload it from the WMC, as described in 4.2.13, “Uploading and downloading a project” on page 197.

After the project is published, go to the WMC, validate the configuration properties, and start the configuration. The orchestration immediately polls the FTP server and executes against any files that it finds there.

Tip: The FTP Poll activity remembers the files that it already received. There is a property in the activity called “Duplicate List Size” that has a minimum of 10,000 files. So, it remembers at least 10,000 files based on the name and time stamp that it has already processed. To retest the orchestration with the same file, you must stop, undeploy the orchestration, and then restart it. Alternatively, you can change the file content, which updates the time stamp automatically.



A

Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG248004>

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248004.

Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>Folder name</i>	<i>Description</i>
Scenario3	Contains files required to run the scenario in Chapter 12, “Scenario: Data enrichment and aggregation” on page 465
GoogleCalendar	Contains the WSDL and schema files used to build the Google Calendar connector example.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- ▶ *Connect Cloud and On-premise Applications Using IBM WebSphere Cast Iron Integration, REDP-4674*

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Online resources

These websites are also relevant as further information sources:

- ▶ IBM WebSphere Cast Iron Version 6.1 Information Center
<http://publib.boulder.ibm.com/infocenter/wci/v6r1m0/index.jsp>
- ▶ WebSphere Cast Iron Cloud Integration
<http://www-01.ibm.com/software/integration/cast-iron-cloud-integration/>
- ▶ Cast Iron Express
<http://express.castiron.com/express/>
- ▶ WebSphere Cast Iron Cloud Integration prerequisites
<http://www-01.ibm.com/software/integration/cast-iron-cloud-integration/reqs/>
- ▶ IBM Support Portal
<http://www-947.ibm.com/support/entry/portal/Overview>
- ▶ IBM Fix Central
<http://www.ibm.com/support/fixcentral>
- ▶ WebSphere Cast Iron Cloud integration Fixpacks
<http://www-933.ibm.com/support/fixcentral/swg/selectFixes?parent=ibm~WebSphere&product=ibm/WebSphere/WebSphere+Cast+Iron+Cloud+Integration&release>All&platform>All&function=all>
- ▶ WebSphere Cast Iron Cloud integration Community page
<http://community.castiron.com>
- ▶ Salesforce Implementation Considerations

- http://www.salesforce.com/us/developer/docs/api/Content/implementation_considerations.htm
- ▶ *WebSphere Cast Iron Live Security,*
<http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=WH&htmlfid=WSW14139USEN>
- ▶ WebSphere DataPower Service Gateway XG45
<http://www-01.ibm.com/software/integration/datapower/XG45/>
- ▶ VMware vSphere vMotion
<http://www.vmware.com/products/vmotion/overview.html>
- ▶ Google Calendar APIs and Tools
http://code.google.com/apis/calendar/data/2.0/developers_guide_protocol.html

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

IBM



Redbooks

Getting Started with IBM WebSphere Cast Iron Cloud Integration

(1.0" spine)
0.875" <-> 1.498"
460 <-> 788 pages



Getting Started with IBM WebSphere Cast Iron Cloud Integration



Learn how to integrate applications, both on-premise and in the cloud

Learn about WebSphere Cast Iron Integration Solution technology

Design, build, and manage integration solutions

Cloud computing provides companies with many capabilities to meet their business needs but can also mean that a hybrid architecture is created that includes on-premise systems and the cloud. Integration is needed to bridge the gap between the on-premise existing systems and the new cloud applications, platform, and infrastructure.

IBM® WebSphere® Cast Iron® meets the challenge of integrating cloud applications with on-premise systems, cloud applications-to-cloud applications, and on-premise to on-premise applications. It contains a graphical development environment that provides built-in connectivity to many cloud and on-premise applications and reusable solution templates that can be downloaded from a solution repository. The integration solutions that are created can then run on either an on-premise integration appliance or the multi-tenant WebSphere Cast Iron Live cloud service.

This IBM Redbooks® publication is intended for application integrators, integration designers, and administrators evaluating or already using IBM WebSphere Cast Iron. Executives, leaders, and architects who are looking for a way to integrate cloud applications with their on-premise applications are also shown how WebSphere Cast Iron can help to resolve their integration challenges.

The book helps you gain an understanding of Cast Iron and explains how to integrate cloud and on-premise applications quickly and simply. It gives a detailed introduction to the development tool and the administration interfaces and how they are used. It also discusses security, high availability, and re-usability. The book also includes three detailed scenarios covering real-world implementations of a Cast Iron Integration Solution.

**INTERNATIONAL
TECHNICAL
SUPPORT
ORGANIZATION**

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks