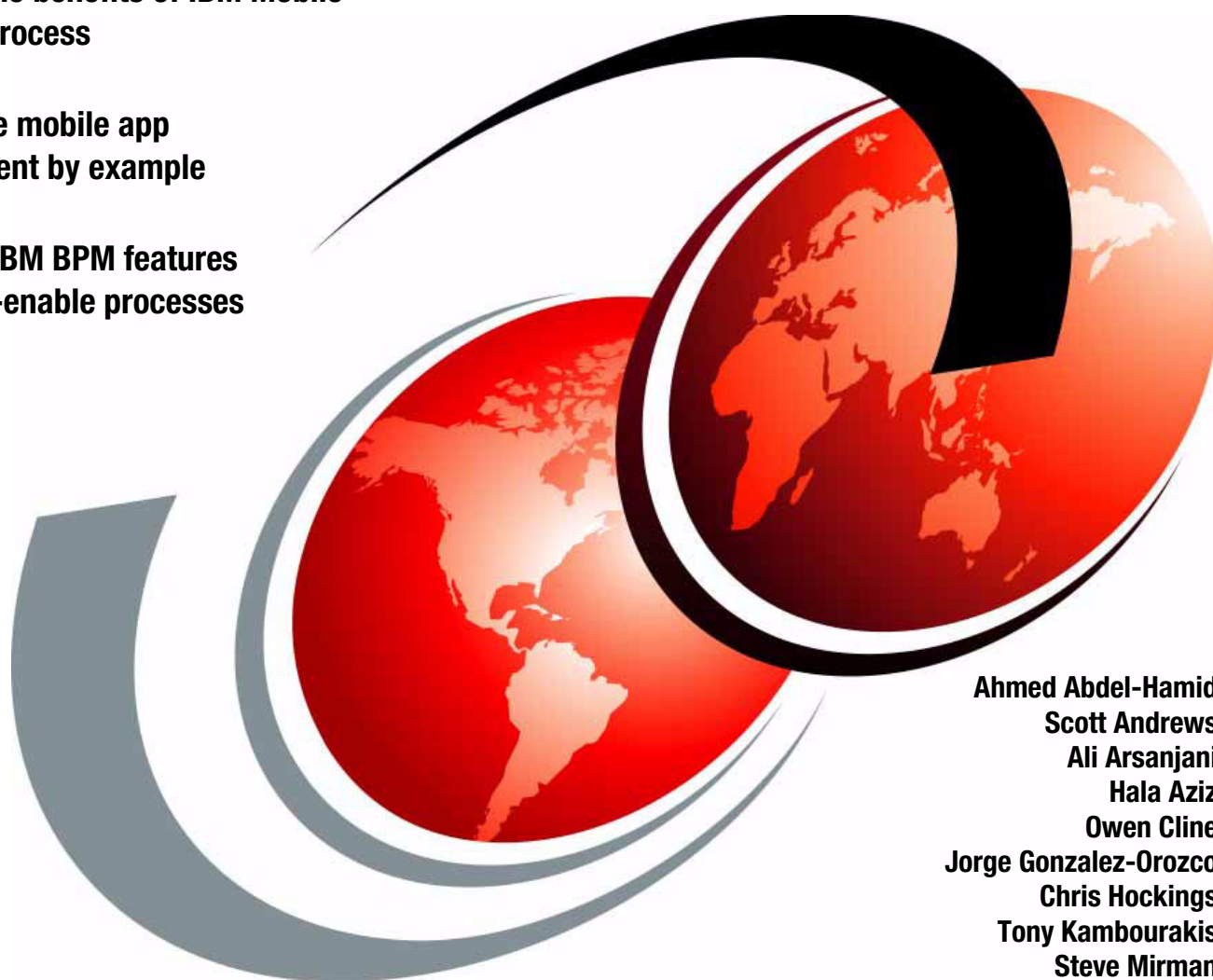


Extending IBM Business Process Manager to the Mobile Enterprise with IBM Worklight

Unleash the benefits of IBM Mobile Smarter Process

Accelerate mobile app development by example

Discover IBM BPM features to mobile-enable processes



Ahmed Abdel-Hamid
Scott Andrews
Ali Arsanjani
Hala Aziz
Owen Cline
Jorge Gonzalez-Orozco
Chris Hockings
Tony Kambourakis
Steve Mirman

Redbooks



International Technical Support Organization

**Extending IBM Business Process Manager to the
Mobile Enterprise with IBM Worklight**

February 2015

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

First Edition (February 2015)

This edition applies to Version 6, Release 2, Modification 0 of IBM Worklight (product number 5725-I43) and Version 8, Release 2, Modification 5 of IBM Business Process Manager (product number I5725-C94).

© Copyright International Business Machines Corporation 2015. 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	ix
Trademarks	x
IBM Redbooks promotions	xi
Preface	xiii
Authors	xiii
Now you can become a published author, too!	xvii
Comments welcome	xvii
Stay connected to IBM Redbooks	xvii
Chapter 1. Mobile Smarter Process overview	1
1.1 Drivers for change: Market forces in the digital economy	2
1.2 What is IBM Mobile Smarter Process?	3
1.2.1 IBM Smarter Process	3
1.2.2 IBM MobileFirst	4
1.2.3 IBM Mobile Smarter Process	4
1.2.4 Benefits and value of Mobile Smarter Process	8
1.3 Defining a mobile strategy	11
1.3.1 Elements of a leading mobile strategy	11
1.3.2 Common mistakes	11
1.3.3 Identifying mobile business usage patterns	12
1.3.4 Identifying use cases that deliver highest value	13
1.3.5 Mobile Smarter Process transformation: Lessons learned in the trenches	14
1.3.6 Mobile technology adoption patterns	17
1.4 IBM BPM features to mobile-enable processes	19
1.4.1 IBM BPM mobile app for iOS	19
1.4.2 IBM Business Process Manager REST Interface	20
1.4.3 Non-production entitlement to Worklight Enterprise Edition with IBM BPM V8.5 and later	20
1.4.4 Mobile app development accelerators	21
1.4.5 Client-side human services	22
1.4.6 Responsive coach design	23
1.5 IBM Worklight features to mobile-enable processes	26
1.5.1 Worklight adapters	26
1.5.2 IBM Worklight Studio	27
1.5.3 Command-line interface for IBM Worklight developers	27
1.5.4 Worklight APIs	27
1.5.5 Location services (geolocation)	28
1.5.6 Push notification	28
1.5.7 Worklight security and LTPA	28
1.5.8 Offline-enabled mobile applications	29
1.5.9 Operational analytics	29
1.6 Conclusion	29
Chapter 2. IBM Worklight Foundation V6.2 overview	31
2.1 Description of Worklight	32
2.1.1 Worklight Studio	35
2.1.2 Worklight Server	37

2.1.3 Worklight device runtime components	38
2.1.4 Worklight Application Center	38
2.1.5 Worklight Console	39
2.2 Worklight in IBM MobileFirst	39
2.3 Business benefits	40
2.4 System topology	41
2.5 Components	43
2.5.1 Worklight Studio and command-line interface for IBM Worklight Developers	44
2.5.2 Client-side runtime environment	45
2.5.3 Worklight Server runtime environment	47
2.5.4 Worklight Console	50
2.5.5 Worklight Application Center	50
Chapter 3. IBM Business Process Manager V8.5.5 overview	53
3.1 Understanding business process management	54
3.2 Lifecycle of a business process	55
3.3 IBM Business Process Manager	56
3.3.1 Product overview	57
3.3.2 Product editions	58
3.4 What's new in IBM BPM V8.5.5	60
3.4.1 IBM Business Monitor V8.5.5	62
Chapter 4. Architecture patterns for mobile-enabled processes	65
4.1 IBM MobileFirst reference architecture	66
4.2 Enterprise mobile architecture	67
4.2.1 Mobile devices and applications	67
4.2.2 Mobile channel services	68
4.2.3 Business process management and enterprise integration	72
4.3 Deployment topology	74
4.4 IBM BPM interaction patterns	75
4.4.1 Pattern 1: IBM BPM Process Portal	76
4.4.2 Pattern 2: IBM BPM dashboard	77
4.4.3 Pattern 3: IBM BPM portlet and iFrame	77
4.4.4 Pattern 4: IBM BPM REST API	77
4.4.5 Pattern 5: API facade	78
4.4.6 Pattern 6: Message or web service start	79
4.4.7 Pattern 7: Service orchestration and advanced integration	80
4.4.8 Pattern selection guide	80
Chapter 5. Enterprise mobile security with IBM Security Access Manager and IBM Worklight	83
5.1 Security principals	83
5.1.1 Authentication	83
5.1.2 Access control	84
5.1.3 Server side single sign-on	85
5.2 Mobile security principals	86
5.2.1 Staff (internal and external)	86
5.2.2 Customer (external)	86
5.3 IBM Worklight security overview	87
5.3.1 IBM Worklight security capabilities	87
5.3.2 What is new in IBM Worklight v6.2 security	90
5.4 Extending Worklight security with IBM Security Access Manager	91
5.5 IBM Security Access Manager for Mobile overview	91
5.6 Planning for OAuth using IBM Security Access Manager for Mobile	92

5.6.1 OAuth patterns	92
5.6.2 PIN protection on refresh tokens	95
5.7 Integrating Mobile and self-service requirements into IBM Security Access Manager for Mobile	96
5.7.1 Create API protection definition	96
5.7.2 Create OAuth client.	98
5.7.3 Attach API protection policy to endpoints	100
5.8 Example OAuth flow with PIN validation using IBM Security Access Manager for Mobile.	101
5.8.1 Application registration	101
5.8.2 Two factor authentication step up	104
5.8.3 Business API using access token/refresh token	104
5.8.4 Administrator device management	105
5.8.5 User device management	106
5.9 Conclusion	109
Chapter 6. Scenario 1: Getting started	111
6.1 Scenario 1 requirements and use case description	112
6.1.1 Customer use case description.	112
6.1.2 Field technician use case	112
6.1.3 Overall architecture.	113
6.1.4 Additional considerations	114
6.2 Field service mobile application	115
6.3 Worklight features	121
6.3.1 Adapter-based authentication	121
6.3.2 Geolocation.	123
6.3.3 Offline storage.	124
6.3.4 Push notifications	126
6.3.5 IBM BPM integration using adapters.	127
6.4 Worklight implementation details	128
6.4.1 Worklight project	128
6.4.2 Creating the adapters	130
6.4.3 User Interface	137
6.4.4 Security and authentication.	140
6.4.5 Calling adapters directly	144
6.4.6 Supporting offline storage.	147
6.4.7 Push notifications support.	154
6.4.8 Geolocation support	157
6.4.9 Storing location information in IBM Bluemix	160
6.5 IBM BPM processes supporting this scenario.	162
6.6 Building the business process.	167
6.6.1 Creating the business process definition	167
6.6.2 Define the business objects and variables	168
6.6.3 Implementing the Capture Customer Order coach forms	170
6.6.4 Implementing the Installation Work Order linked process.	172
6.6.5 Implementing the Create Customer Profile system activity.	175
6.6.6 Implementing the Schedule Work Order system activity.	184
6.6.7 Implementing the Send Notification to Field Staff activity.	195
6.7 IBM BPM REST API	197
Chapter 7. Scenario 2: Advanced features	207
7.1 Scenario 2 overview	208
7.1.1 Scenario 2 requirements.	208

7.1.2 Customer use case	208
7.1.3 Field technician use case	209
7.1.4 Overall architecture	209
7.2 Parts app	210
7.3 IBM Worklight features in this scenario	214
7.3.1 LTPA authentication and token propagation	214
7.3.2 Device single sign-on	216
7.3.3 Data sharing across mobile applications	216
7.3.4 Integration with IBM Mobile Data for Bluemix database	217
7.4 SSO configuration with LTPA implementation	217
7.4.1 Configuring LTPA on WebSphere Application Server	218
7.4.2 Configuring the Worklight Server to authenticate using LTPA	220
7.4.3 Configuring the Worklight mobile app to authenticate using LTPA	224
7.4.4 Testing the SSO configuration	228
7.5 Data sharing across apps implementation	228
7.5.1 Enabling Simple Data Sharing in the Field Service app	228
7.5.2 Saving Order ID in the Field Service app	229
7.6 Parts app implementation	230
7.6.1 Creating the Parts database in Bluemix	230
7.6.2 Creating the Parts app	231
7.6.3 Enabling Simple Data Sharing in the Parts app	232
7.6.4 Creating the Parts DB Bluemix adapter	232
7.6.5 Creating the Parts Order adapter	233
7.7 Extending scenario 1 with IBM BPM case management	237
7.7.1 NewCustomerOrderCase case type overview	240
7.7.2 Activities in NewCustomerOrderCase	246
7.7.3 User tasks (human services) implementation	252
7.8 Responsive coaches	261
7.8.1 IBM Process Designer web editor	261
7.8.2 Modifying a coach form to be responsive	268
7.8.3 Accessing responsive coaches from mobile devices	280
 Chapter 8. Deeper insight through	
IBM Business Process Manager and IBM Worklight analytics	283
8.1 IBM BPM business analytics	284
8.1.1 Business Space widgets for business monitoring	286
8.1.2 IBM Business Monitor topologies	290
8.1.3 What is new in IBM Business Monitor v8.5.5	293
8.2 IBM BPM operational analytics	293
8.2.1 Process Monitor Summary page	294
8.2.2 Process Monitor Process page	294
8.2.3 Process Monitor Services page	295
8.3 Worklight operational analytics	295
8.3.1 Data capture	296
8.3.2 Analytics views	298
 Appendix A. Samples included with this book	307
Samples compressed file content	308
Scenario 1 sample instructions	310
IBM BPM scenario 1 sample artifacts	310
IBM BPM integration services and IBM Bluemix	312
IBM Worklight scenario 1 sample artifacts	314
Scenario 2 sample instructions	315
IBM BPM scenario 2 sample artifacts	316

IBM Bluemix	316
IBM Worklight scenario 2 sample artifacts	316
Testing on an Android device.	317
Software version reference	318
Appendix B. Additional material	319
Locating the Web material	319
Using the Web material	319
System requirements for downloading the Web material	319
Downloading and extracting the Web material	320
Related publications	321
IBM Redbooks	321
Online resources	321
Help from IBM	321

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 grant 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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:

Bluemix™
Blueworks Live™
Cast Iron®
CICS®
Cognos®
DataPower®
DB2®
developerWorks®

FileNet®
Global Business Services®
IBM®
IBM SmartCloud®
IBM Watson™
IMS™
Rational®
Redbooks®

Redbooks (logo) ®
Tealeaf®
Tivoli®
Trusteer®
Watson™
WebSphere®
Worklight®

The following terms are trademarks of other companies:

MaaS360, and We do IT in the Cloud. device are trademarks or registered trademarks of Fiberlink Communications Corporation, an IBM Company.

SoftLayer, and SoftLayer device are trademarks or registered trademarks of SoftLayer, Inc., an IBM Company.

Microsoft, 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.

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

Find and read thousands of IBM Redbooks publications

- ▶ Search, bookmark, save and organize favorites
- ▶ Get up-to-the-minute Redbooks news and announcements
- ▶ Link to the latest Redbooks blogs and videos

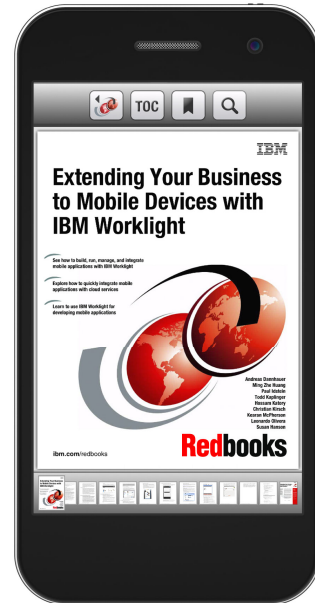
Get the latest version of the Redbooks Mobile App



iOS

Download
Now

Android



Promote your business in an IBM Redbooks publication

Place a Sponsorship Promotion in an IBM® Redbooks® publication, featuring your business or solution with a link to your web site.

Qualified IBM Business Partners may place a full page promotion in the most popular Redbooks publications. Imagine the power of being seen by users who download millions of Redbooks publications each year!



ibm.com/Redbooks

About Redbooks → Business Partner Programs

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

In today's business in motion environments, workers expect to be connected to their critical business processes while on-the-go. It is imperative to deliver more meaningful user engagements by extending business processes to the mobile working environments.

This IBM® Redbooks® publication provides an overview of the market forces that push organizations to reinvent their process with Mobile in mind. It describes IBM Mobile Smarter Process and explains how the capabilities provided by the offering help organizations to mobile-enable their processes.

This book outlines an approach that organizations can use to identify where within the organization mobile technologies can offer the greatest benefits. It provides a high-level overview of the IBM Business Process Manager and IBM Worklight® features that can be leveraged to mobile-enable processes and accelerate the adoption of mobile technologies, improving time-to-value. Key IBM Worklight and IBM Business Process Manager capabilities are showcased in the examples included in this book. The examples show how to integrate with IBM Bluemix™ as the platform to implement various supporting processes.

This IBM Redbooks publication discusses architectural patterns for exposing business processes to mobile environments. It includes an overview of the IBM MobileFirst reference architecture and deployment considerations.

Through use cases and usage scenarios, this book explains how to build and deliver a business process using IBM Business Process Manager and how to develop a mobile app that enables remote users to interact with the business process while on-the-go, using the IBM Worklight Platform.

The target audience for this book consists of solution architects, developers, and technical consultants who will learn the following information:

- ▶ What is IBM Mobile Smarter Process
- ▶ Patterns and benefits of a mobile-enabled Smarter Process
- ▶ IBM BPM features to mobile-enable processes
- ▶ IBM Worklight features to mobile-enable processes
- ▶ Mobile architecture and deployment topology
- ▶ IBM BPM interaction patterns
- ▶ Enterprise mobile security with IBM Security Access Manager and IBM Worklight
- ▶ Implementing mobile apps to mobile-enabled business processes

Authors

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



Ahmed Abdel-Hamid is a Certified Expert IT Specialist in Software Group Services in IBM Egypt. Ahmed has over 12 years experience in services projects in Mobile, IBM Business Process Manager, and IBM Web Content Management. Throughout his career, Ahmed took on many roles, including software developer, solution architect, and consultant. In his current role, Ahmed is a developer and architect in the IBM Mobile Center of Competence team.



Scott Andrews is an Advisory Software Engineer in the IBM Australia Development Lab in Gold Coast, Australia. He is the team leader for the IBM Security Access Manager Integration Factory, specializing in IBM Security Access Manager for Web, IBM Security Access Manager for Mobile, IBM Tivoli® Federated Identity Manager, and mobile security. Scott has designed and implemented security integrations for many applications and products, including IBM MobileFirst Platform Foundation, IBM Security Trusteer®, IBM MaaS360®, JBoss, and Microsoft SharePoint.



Ali Arsanjani is an IBM Distinguished Engineer and member of IBM Academy of Technology. He is the CTO of Smarter Process, service-oriented architecture (SOA), and API Management for Software Services.



Hala Aziz is an IT Specialist in the Cairo Technology Development Center (CTDC) in IBM Egypt. She has 10 years of experience in IBM Application and Integration Middleware Software, such as IBM WebSphere® Application Server, IBM WebSphere Portal, IBM Worklight, and IBM Endpoint Manager. She worked as a consultant on eGovernment and banking solutions for clients in Egypt, Dubai, Oman, and Switzerland. Hala has several technical professional certifications, such as Certified Application Developer for IBM Web Content Manager and IBM Worklight and she has delivered IBM internal education and client enablement training workshops around the world.



Owen Cline is a member of the IBM Software Services for WebSphere team based in San Diego, CA. He earned a BS in Computer Science from the University of Pittsburgh and an MS from San Diego State University. He has over 20 years of experience in software development. Owen holds four software patents, has written several IBM Redbooks publications, and he is a frequent speaker in technical conferences. In recent years, Owen has specialized in IBM Business Process Manager; IBM Operational Decision Manager; and Java Platform, Enterprise Edition application development and deployment (with a special emphasis on the WebSphere platform). Owen worked as IBM BPM architect for many clients in North and Latin America.



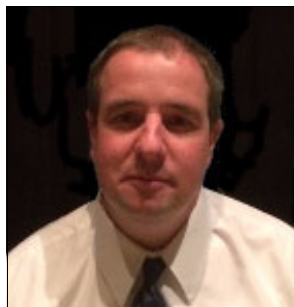
Jorge Gonzalez-Orozco is a senior Mobile Solution Architect in IBM. He has experience in Mobile, e-commerce, enterprise integration, and software development lifecycle. Jorge is a hands-on IT Architect who has successfully led multiple teams in the role of lead architect and project manager. Jorge has led the strategy, design, and delivery of IT solutions for the automotive, retail, insurance, and energy industries by managing large global delivery teams. Jorge is also an IBM Redbooks thought leader and he holds the IBM IT Architect and SOA Solution Designer certifications. Jorge is based in Raleigh, NC US.



Chris Hockings is an IBM Master Inventor and Executive IT Specialist. He leads a team of security technology experts in the Australia Development Lab and across the worldwide IBM Security Systems SWAT team.



Tony Kambourakis is an IBM Certified IT Architect. Tony is highly experienced in the design and delivery of cross-industry solutions that include multi-channel digital platforms, field force automation, business process management, SOA, electronic design automation (EDA), and enterprise integration. Tony has a broad range of experience in business analysis, technical architecture, development, and applied methodology, which provides him the breadth of skills required to successfully lead large delivery teams. Tony is the Technology and Offerings Lead in the IBM Business Process Manager and Integration practice and he is a technical subject matter expert (SME) in the IBM Mobility Community of Practice. He has worked at IBM for 18 years specializing in IBM BPM, integration, and mobile solutions.



Steve Mirman is a Senior Certified IT Architect focusing on IBM Worklight and the MobileFirst portfolio. He has over 15 years of IT experience as an application developer, solution architect, and product specialist. In his current role, Steve leads proof of concept implementations and consults with customers throughout North America on mobile strategy, architecture, and integration. He holds numerous IT certifications and has authored several technical articles.

The project that produced this publication was managed by **Marcela Adan**, IBM Redbooks Project Leader - IBM International Technical Support Organization, Global Content Services.

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

John Alcorn
Mobile Smarter Process CTO office, IBM Software Group

Dustin Amrhein
IBM MobileFirst Solution Architect, IBM Software Group

Mangalaganesh Balasubramanian
IBM BPM and Integration, IBM Global Business Services® Australia

Sagy Bar
IBM MobileFirst Platform Product Management, IBM Software Group

Karl Bishop
IBM MobileFirst Platform Product Management, IBM Software Group

Alessandro Campioli
Smarter Process Center of Competence, IBM Software Group

Deepak Elias
IBM Watson™ Product Management, IBM Software Group

Scott Glen
Smarter Process Center of Competence, IBM Software Group, IBM United Kingdom

Christopher Hockings
Security Systems, IBM Software Group Australia

Guy Lipof
Smarter Process Center of Competency, IBM Software Group

Michael Morris
Mobile Smarter Process & SaaS Offerings, IBM Software Group, Worldwide Sales

Shishir Narain
Mobility and business solution services, India Software Lab, IBM India

Christopher Schmitt
IBM Application and Integration Middleware Software Marketing, IBM Software Group

Greg Truty
IBM MobileFirst Platform Chief Architect, IBM Software Group

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:

<https://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>



Mobile Smarter Process overview

This chapter¹ provides an overview of the market forces that push organizations to reinvent their process with Mobile in mind. It describes IBM Mobile Smarter Process and explains how the capabilities provided by the offering help organizations to mobile-enable their processes.

This chapter outlines an approach that organizations can use to identify where within the organization mobile technologies can offer the greatest benefits.

This chapter includes an overview of the typical adoption patterns followed by organizations in their journey to mobile-enable their processes. It provides an overview of effective procedures that organizations should consider when designing their Mobile strategy.

Finally, this chapter provides a high-level overview of the IBM Business Process Manager and IBM Worklight features that can be leveraged to mobile-enable processes and accelerate the adoption of mobile technologies improving time-to-value.

This chapter includes the following topics:

- ▶ Drivers for change: Market forces in the digital economy
- ▶ What is IBM Mobile Smarter Process?
- ▶ Patterns and benefits of a mobile-enabled Smarter Process
- ▶ Mobile Smarter Process adoption patterns
- ▶ IBM BPM features to mobile-enable processes
- ▶ IBM Worklight features to mobile-enable processes

This IBM Redbooks publication includes scenarios, use cases, and implementation examples that illustrate how IBM BPM processes can be extended into the mobile space with IBM Worklight mobile apps and mobile devices.

¹ This chapter is based on original materials provided by Ali Arsanjani, Deepak Elias, Michael Morris, and Christopher Schmitt

1.1 Drivers for change: Market forces in the digital economy

The digital economy has already driven and continues to drive radical changes in the market place. The study, *Digital reinvention: Preparing for a very different tomorrow* conducted in 2013² by the IBM Institute for Business Value, describes and predicts how digital technologies are and will continue driving drastic changes in the economy. Digital technologies such as social media, mobility, analytics, and cloud keep changing how people, businesses, and governments interact. Who, how, and when customers are interacting with organizations is driving radical change. According to the survey conducted by the IBM Institute for Business Value for this study:

- ▶ 63% of leaders expect consumers to gain even more power and influence of their businesses.
- ▶ 58% of business executives expect new technology to reduce barriers to entry.
- ▶ 69% of business leaders expect more cross-industry competition, which is happening already as book sellers are becoming cloud service providers, telecoms are becoming banks, banks are becoming insurance providers, and so on.

In this individual-centered economy, it is easy to see how, in the last few years, each of us personally as customers of businesses and government agencies have changed our interactions with them and also our expectations of the services we expect to receive. Our tolerance for losing a service or convenience if we have to move to a new provider is very low. As customers, we expect to get what we want, based on our current situation, and as soon as we need it. If the provider does not supply exactly what we want, how we want it, and when we want it, we start wondering if this is the best vendor for us. We look around and likely quickly switch to the vendor that meets our expectations.

As employees, we value flexibility at work very highly, in some cases even more than salary. We want to be able to do a good job for our bosses and clients but at the same time we want to have the flexibility to do it from home or while watching our kids playing a soccer game. We want access to our work resources from anywhere, any time, and from any device. Our employers know that they will get better results and better customer service if their employees are mobile-enabled.

To compete for clients, organizations need to not just meet the expectation of their demanding customers but also provide an enjoyable customer experience. Market leaders are using mobile technologies to differentiate themselves to their competitive advantage.

The “*upwardly mobile*” enterprise study conducted by the IBM Institute for Business Value and Oxford Economics offers insight into the practices of mobile leaders. According to the study, organizations that have a clear direction for their mobile efforts and see their mobile strategies as distinguishing them from their peers, outperform their peers across a number of business metrics. Mobile strategy leaders have also seen clear benefits in their mobile investments to date:

- ▶ 73% of leaders have seen measurable ROI from their mobile initiatives.
- ▶ 81% stated that mobile capabilities are fundamentally changing the way that their organizations do business.

Mobile strategy leaders reported that:

- ▶ They are using mobile technologies to fundamentally change the ways they do business.

² Download the IBM Institute for Business Value executive report from
<http://www-935.ibm.com/services/us/gbs/thoughtleadership/digitalreinvention>

- ▶ They are mobilizing their employees and are turning information from mobile into insights. Faster response time to customers is the primary reason for investing in employees use of mobile technologies.
- ▶ They are unlocking core business knowledge for mobile uses. Integrating mobile systems with existing systems is the number one challenge faced by organizations. The majority of mobile strategy leaders indicate that they are effective in integrating mobile technologies and business processes.
- ▶ They are securing the mobile enterprise: They focus on addressing security issues around the protection of data, securing connectivity, device management threat detection, mobile app security, and user security.
- ▶ They are getting results. They have seen measurable ROI from their mobile initiatives.

1.2 What is IBM Mobile Smarter Process?

This section introduces IBM Mobile Smarter Process and its components: IBM Smarter Process and IBM MobileFirst. This section includes a description of the benefits and value of IBM Mobile Smarter Process.

1.2.1 IBM Smarter Process

A *business process* is a series of tasks or activities that are performed to accomplish a specific goal in the organization. A business process *workflow* orchestrates the activities and creates a repeatable pattern that can be automated. Very often processes require human intervention, for example, an approval, the authorization of an exception, data that must be input to the process. IBM BPM includes coach forms, which are a set of web-based user interfaces that guide or *coach* users through their interaction with the process. At some point during execution, a process needs access to systems of record in the organization's back-end. Up to this point, certain capabilities can be identified that make the process run more efficiently, with fewer errors, faster but not necessarily smarter.

A business process becomes a *smarter process* when it can be started in response to an event, for example, someone applying for a loan, or submitting an insurance claim from a mobile device. Once the smarter process starts, there will be points during its execution where decisions must be made. For example, based on the credit rating of the person applying for a loan, a sub-process is triggered with a set of rules that guide specific decisions relative to the interest rates, maximum loan amount, and so on. The direction of the business process is affected by the context provided. The process becomes *smarter* because it can adapt to different context and it enters a different set of options that may be available depending on the context. The customer may be presented with up-sale or cross-sale alternatives based on context, rules, and decisions.

To summarize, the components of a smarter process are:

- ▶ Business process, process steps, workflow.
- ▶ Business rules and decisions.
- ▶ Events that are triggered and cause a process or a branch in the process to start.
- ▶ Service-oriented architecture (SOA) that provides the integration capabilities to access the systems of record in the enterprise back-end.
- ▶ Analytics that enables organizations to add intelligent actions to the process. For example, analytics helps to translate all the information an organization knows about a customer

(within and external to the enterprise) into actions or interactions that make sense to the customer (*next best action*), driving long-term customer loyalty and value.

From an IBM product offering perspective, *IBM Smarter Process* is a set of business process enhancement and optimization tools that work together to improve the business outcomes of business processes. IBM software products for Smarter Process include:

- ▶ IBM Business Process Manager
- ▶ IBM Operational Decision Manager
- ▶ IBM Integration Bus, formerly known as the IBM WebSphere Message Broker family
- ▶ IBM Case Manager software
- ▶ IBM Infosphere product set
- ▶ IBM Business Monitor
- ▶ IBM Big Data and Analytics portfolio

1.2.2 IBM MobileFirst

IBM MobileFirst is a significant mobile strategy from IBM that enables organizations to radically streamline and accelerate mobile adoption. IBM assembled the people, technology, and research and development (R&D) to build the most comprehensive mobile portfolio in the industry. IBM MobileFirst combines deep industry expertise with mobile, big data and analytics, cloud and social technologies to help organizations to implement a mobile strategy to drive the wanted business outcomes, capture new markets, and reach more people.

With this end-to-end solution, IBM enables an enterprise to benefit from mobile interactions with customers, with business partners, and within organizations. There are products available from the IBM MobileFirst solution to support management, security, analytics, and development of the application and data platforms in a mobile environment. Additionally, IBM also provides services for strategy and design, cloud and managed services, and development and integration services to support mobile activities. There are several aspects to consider for a cost-efficient and secure Mobile environment. The IBM MobileFirst portfolio includes products in the following areas:

- ▶ Application and data platform
- ▶ Management
- ▶ Security
- ▶ Customer engagement and analytics

For the application and data platform component, the key capabilities in the platform are oriented to help companies to build and deliver engaging mobile solutions more quickly, with higher quality, and at lower cost. Key assets in this space include IBM Worklight and IBM Rational® tools for building and testing mobile assets.

For more information about IBM MobileFirst, refer to the IBM Redbooks publication *IBM MobileFirst Strategy Software Approach*, SG24-8191.

1.2.3 IBM Mobile Smarter Process

Mobile Smarter Process helps organizations to reinvent how business is performed exploiting mobile technologies. The goal is to fundamentally change how an organization does business by integrating Mobile and processes. It enables radical simplification of every customer interaction and leveraging new mobile contextual opportunities to engage the customer in new ways.

Mobile provides the channel and the functionality. Process provides the rigor and the discipline. Organizations should identify the mobile interactions that add value to their

processes and the process must be designed or reinvented to harness the process for these mobile interactions and avoid chaos. Figure 1-1 shows that Mobile creates numerous timely interactions that can be harnessed for goal-oriented results with Smarter Process.

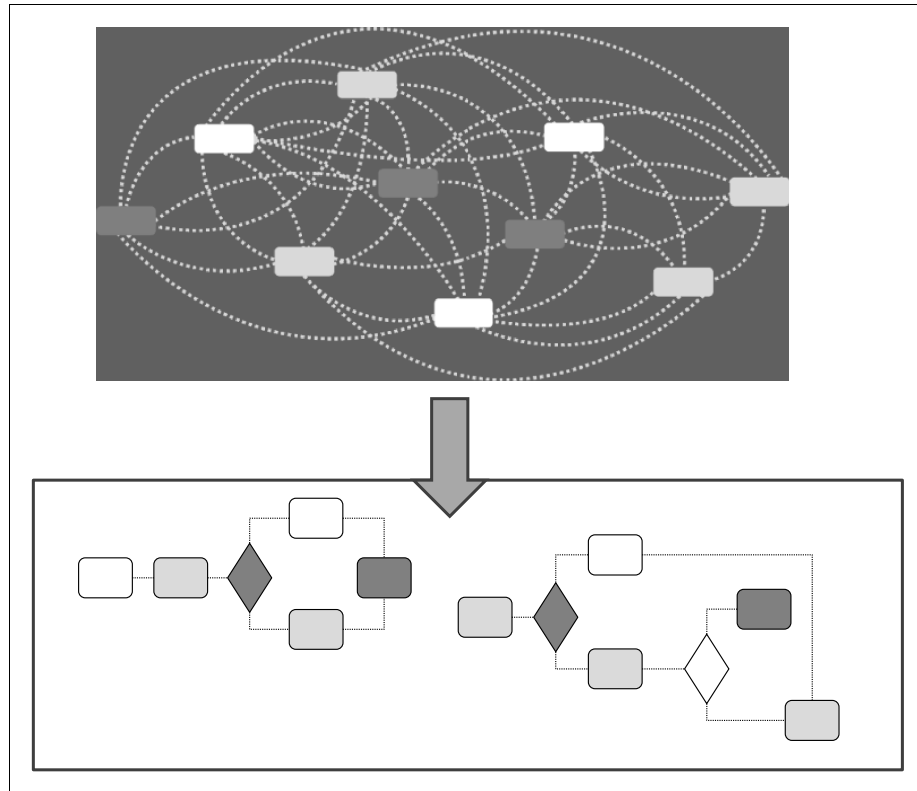


Figure 1-1 Harnessing Mobile interactions with Smarter Process

IBM Mobile Smarter Process combines market leading offerings from IBM Smarter Process suites and IBM MobileFirst offerings.

Figure 1-2 on page 6 provides an example of how Mobile Smarter Process works. This example shows a Smarter Process already in place in the organization that can be exposed through multiple channels, such as different types of mobile devices.



Figure 1-2 Mobile Smarter Process at work

1. Customers, employees, and business partners want to interact with the organization using their device of choice to instantiate a process or check on the status of a process they previously started. For example, in an insurance company scenario, by instantiating a claim, customers start a process that can leverage the geolocation information available on their device, device services, such as the camera to take a picture of a damaged car and upload it to the process, and the information stored in multiple back-end systems.
2. A process may involve many different roles to complete it. Smarter Process uses coaches or forms that streamline the activities of the process performers and work across any mobile devices with different form factors. A Mobile Smarter Process allows for process performers to use social collaboration to deliver a better customer experience. Integration with multiple back-end systems may be necessary to complete the process, which is enabled by SOA.
3. If the process requires human input, it can send a notification to the appropriate people and they can always be reached through their mobile devices. The business process is not going to be hindered by someone not having access to a website or not being logged in.
4. Processes not always occur as planned. As the global economy, the global workforce, and mobility expand, it is even more important to give managers process visibility and agility to make quick changes. Otherwise, organizations risk losing customers. A Mobile Smarter Process gives managers the visibility and agility they need to make fast adjustments to improve performance using their device of choice.

In summary, this simple example shows that:

- Mobile extends the reach of a business process beyond the traditional channels to where the knowledge workers (customers, employees, business partners, managers) are. Knowledge workers have access to the business process at all times and the business process can reach the knowledge workers when it needs them. For example, if it requires an approval, a notification will come on the approver's mobile device to alert them that the process is waiting for their approval.

- By adding mobility to the smarter process, Mobile Smarter Process allows managers to gain instant visibility so they can make adjustments quicker in order to adapt to changes in the market, make faster adjustments, make decisions quicker, and do so on a device of their choice. The mobile smarter process has constant access to decision makers and decision makers can tap into the business process at all times.

Mobile Smarter Process maturity levels

By looking at how organizations can apply a systematic approach to engage the mobile user and achieve a customer-centric Smarter Process with Mobile, it is clear that not every organization, and not every process is at the same stage in the Smarter Process or Mobile journey, and the capabilities and skills needed are different depending on the stage.

The following stages can be identified:

- **Mobile aware**

Organizations that fall under this category are using mobile to enhance the process with intelligence in form of capabilities like geolocation. Organizations are tapping into the core benefits and key features of mobile devices and initiating the mobile experience to enhance the process.

- **Mobile specific**

Organizations that fall under this category have employees and customers that are comfortable initiating the mobile experience to enhance the process. For example, taking pictures, capturing signatures, updating data. These capabilities inject data back into the process.

- **Mobile centric**

Very few organizations are at this stage. This category is where the process is viewed through a mobile lens from the beginning. Not *only* a mobile lens, but Mobile is considered from the very beginning of the process discovery and modeling to achieve a true Smarter Process.

Figure 1-3 on page 8 shows a progression through the seven stages of Mobile Smarter Process maturity.

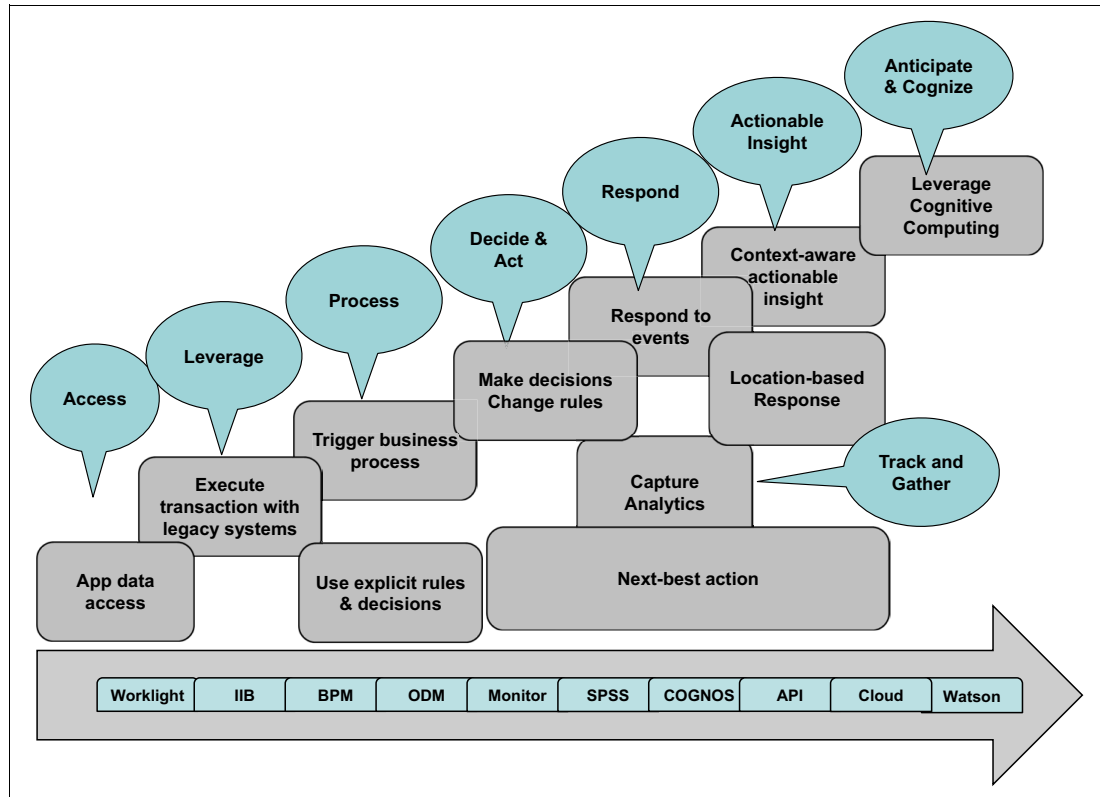


Figure 1-3 Mobile Smarter Process: Maturity levels

- ▶ The first level of maturity is a simple mobile app accessing data, for example, accessing the systems of record at the organization's back-end systems.
- ▶ The second level represents transactions between the mobile app and the legacy systems. In this stage, the mobile app not only reads the data but it also can update and add data in the organization's back-end systems.
- ▶ The third level is where Smarter Process starts. At this level, the process can be triggered by an event, and there is interaction between the business process and the user of the mobile device. For example, a process is running and at some point it needs input from a decision maker at which point the process sends a notification to the user or presents a coach to guide the user's input.
- ▶ Moving to the right, the figure shows that, as more capabilities and IBM solutions and offerings are added to Smarter Process, the organization can do more and attain higher business value.

1.2.4 Benefits and value of Mobile Smarter Process

To survive in the digital economy, organizations need a way to transform how they engage with customers, how they empower their employees, and how they establish their ecosystem with partners. Leading organizations are realizing that mobile is more than just an app. Mobile is not just a technology or a channel to transfer knowledge. It is not just a way to stay in constant contact with individuals. It is a way to gather data and convert it into personalized insight. It is a way to transform how a business operates, and it is a way to reinvent industries.

An effective mobile strategy cannot be a simple add-on to business operations. It cannot even be integrated into business operations. It needs to be *fundamental* to business operations.

Just as today, the notebook computer is ingrained in business processes, mobile must be viewed in the same manner: it is the integral way to engage with customers on their terms; when, where, and how they want.

But organizations will never realize the full business value of mobile without reinventing their business operations. Old, pre-mobile business processes simply will not work. Competitive businesses require a whole new approach to engaging and supporting their customers, supporting their employees, and connecting with their partners.

The way the Internet changed retail provides a good example. There were the companies that put up a website to convey product information, where consumers still had to either visit the store or call to order. And then there were others who revolutionized the shopping experience. But to do that, they could not follow the same business model and the same business processes as brick and mortar retailers. They had to create a whole new way to engage, a new way to shop, and a new way to support their customers.

Mobile Smarter Process enables organizations to take advantage of mobile technologies and embed those technologies directly into the processes of an organization. It promotes radical simplification of every interaction. It allows organizations to use new mobile contextual opportunities to engage customers in innovative ways providing new personalized services and creating a culture of customer centricity that was never available before. And it allows the clients to interact with the organization when, where, how and as much as they want to. It creates a level of personal relationship between the organization and the client that has not been seen before. It allows organizations to empower their employees. No longer are employees tied to their desks in an office. Employees have the ability not only to work more closely with their clients but also they can define how the work is performed, make improvements, and come up with new innovations to drive greater value and efficiencies. And it allows organizations to establish their ecosystem where new partners, new sensors, and new data lead to the establishment of completely new experiences and possibilities.

Mobile Smarter Process establishes interactions that are instant, seamless, and insightful. It provides the opportunity to gather insights that were never before possible and to create a new experience, whether for customers, employees, or partners, that engages at every interaction.

Figure 1-4 on page 10 shows that as organizations move gradually from either end of the spectrum, Smarter Process-centric on the left or Mobile-centric on the right, they reach the apex of the business value and innovation triangle in the middle by combining both in Mobile Smarter Process. The optimum point when combining Smarter Process and Mobile provides the highest business value delivering capabilities, such as context awareness, next best action, geolocation awareness, processes triggered by events, process interacting with user when requiring human intervention, and so on.

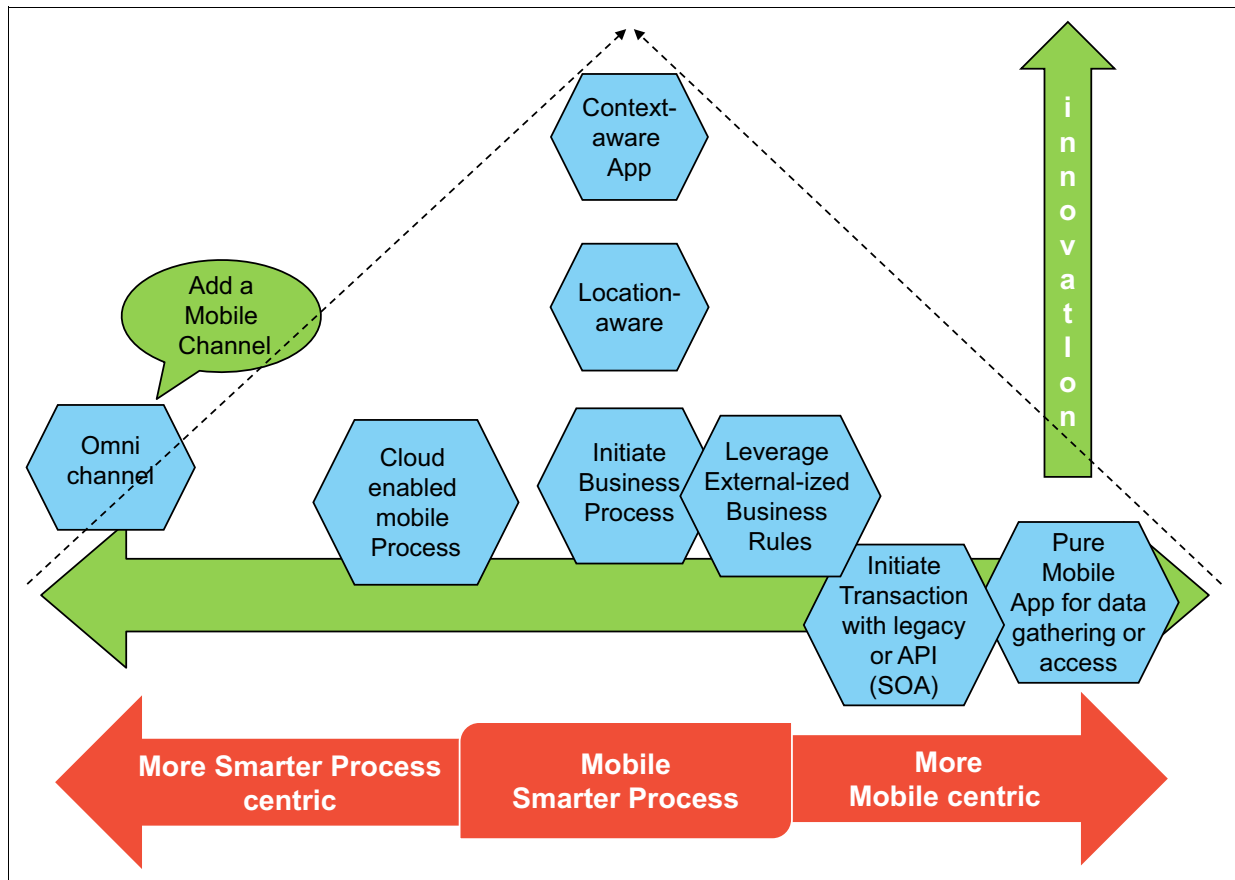


Figure 1-4 Mobile Smarter Process delivers the highest business value

In summary, Mobile Smarter Process enables organizations to realize business value:

- ▶ Increase ROI on mobile projects by:
 - Reducing cost for developing mobile projects.
 - Streamlining processes across multiple systems.
 - Orchestrating manual and automated tasks.
 - Attracting and retaining customers.
- ▶ Improve customer experience by:
 - Engaging the customer using their device of choice.
 - Providing a personalized experience based on customer history, real-time information, and industry trends.
 - Providing social capabilities for better collaboration.
 - Providing real-time performance visibility and process agility to process managers.
- ▶ Improve workforce productivity by:
 - Dynamically assigning tasks based on employee location and availability.
 - Using device services to gather and share all the information related to a task in the right place at the right time.
 - Making employees more responsive to customer needs providing them with the flexibility and information they need, when and where they need it.
 - Sending timely notifications to employees enabling them to handle situations that otherwise would be missed.

- Providing management real-time process visibility to proactively respond and make adjustments.
- ▶ Maximize the value of the partner ecosystem by:
 - Engaging with the appropriate partners through sensors or people when and where an organization needs them.
 - Accelerating time to market, improving partner training and support, shortening the sales cycle.

1.3 Defining a mobile strategy

Without a mobile strategy, organizations pursue individual, often disconnected initiatives, resulting in numerous fragmented and uncoordinated efforts. This lack of integration with existing processes and infrastructure often causes confusion and poor utilization of limited resources. Limited mobile technology skills and resources remain scattered across the enterprise, making it difficult to tackle key initiatives and deliver tangible results to the business. This situation, in turn, makes it more difficult to integrate mobile efforts into a cohesive, enterprise-wide strategy. For mobility to be successful, it requires close partnerships among multiple stakeholders, all owning a valued stake in this rapidly evolving space.

1.3.1 Elements of a leading mobile strategy

A leading mobile strategy:

- ▶ Is aligned to areas where mobile can fundamentally change business processes and models to generate new revenue streams, lower costs, or redefine the organization's role in the value chain.
- ▶ It is enterprise-wide and it is designed to leverage common technology tools, partnerships, platforms, and development resources across business units.
- ▶ Has an established governance structure for mobile initiatives that involves all relevant stakeholders, including line of business (LoB), IT, human resources (HR), and marketing.
- ▶ Takes into account both external and internal initiatives.
- ▶ Ensures that business cases consider both direct costs as well as benefits that accrue in areas not directly bearing the cost of the initiative.

1.3.2 Common mistakes

Some common mistakes organizations must avoid when adopting mobile technologies are:

- ▶ Slapping an out-of-the-box mobile app on an existing process

Too many organizations, in their rush to become mobile, are just taking the mobile applications available from their current software vendor and forcing it to fit on their existing processes and platforms. This approach leads to inefficiencies and errors because the out-of-the-box solutions are not always the best fit for the process. Very often, the process must be redesigned for mobile and the app must be built for the process to fully leverage the mobile capabilities, enhance, and add value to the process.

- Disconnected process improvement and mobile strategies

Many organizations approach their Smarter Process initiative separate from their overall corporate Mobile strategy. Both initiatives must be integrated from the beginning because Mobile is a critical component to fully achieve Smarter Process.

- Pushing Mobile where it does not belong

Organizations are under pressure to go mobile from both consumers and employees. These pressures are leading some organizations to go mobile on every process and system, even if they are not ready or going Mobile is not going to deliver value and benefit. To make matters worse, people do not like change and they resist it mainly if they do not clearly see the benefits that the change brings.

Achieving Smarter Process with mobile entails a systematic approach. The transition to Mobile must target the right use cases and processes, identify the value and benefits, and have the support of stakeholders, otherwise Mobile can produce the opposite effect of the expected outcome. Mobile users ultimately determine the success of the mobile initiative.

Key message: Align operational strategy with Mobile strategy.

1.3.3 Identifying mobile business usage patterns

The mobile strategy of an organization starts by identifying where within the organization mobile technologies can offer the greatest benefits.

There are three main usage patterns:

- Customer engagement. Delivers personalized, differentiated, and location-aware responses.
- Ecosystem-driven processes that may leverage machines, sensors, IoT (Internet of Things), and people to sense and correct situations.
- Workforce enablement or empowerment. Increases workforce effectiveness, productivity, and responsiveness.

The airline business provides an example around *customer engagement* that most people are familiar with that can help to identify use case patterns. Imagine that your flight is delayed or canceled. Nobody wants a situation like this one, but airlines that make the best of this bad situation are more likely to retain their customers. Imagine that you and a colleague are on your way to the airport and are notified of a flight delay. If the delay is longer, the airline might give you offers for a restaurant or hotel on the way to the airport. If it is an overnight delay, neither the airline or you want you to go all the way to the gate to learn of the bad news. On a shorter delay, the offer may vary based on whether you are in geofence of the airport grounds, and also based on your level of loyalty. You may receive in-airport coupons, or an invitation to the airline's lounge.

In the airline example, there is also an *ecosystem-driven process* that can happen at the airport. For example, the airline may be leveraging sensors or the Internet of Things to have passengers baggage rerouted. Another example is spotting and removing debris on runways. Debris causes a huge problem in the airline industry. Runway debris can cause significant damage to airplanes and costs the airline industry more than \$4 B per year. Worse yet, runway debris causes plane crashes. Machines can visually scan the runway in a few minutes, and then alerts can be sent to airport maintenance staff to pinpoint the location of the debris for removal.

The *Workforce enablement or empowerment* pattern includes use cases where the organization enables their workers to be more responsive and productive. Common examples

in an airport can be dynamic task lists that change or redirecting employees based on incoming traffic and weather patterns. Another example: If a flight caught some tailwinds and arrived 40 minutes early, it would be nice if a mobile notification went to someone so that the passengers did not have to wait for 20 minutes to leave the airplane. Or sending an alert when a bag is on the wrong conveyor and will not get on the right plane. Notifying an employee to make the best of that situation would be a huge win for both parties.

1.3.4 Identifying use cases that deliver highest value

As in the airline example described in 1.3.3, “Identifying mobile business usage patterns” on page 12, organizations should go through the exercise of identifying the main usage patterns that apply to them and look for the use cases that can truly drive the value and benefits that they can gain from reinventing their processes for those particular patterns with Mobile Smarter Processes. The key is to look for cases where the unique Mobile enablers, such as location, context, convenience, and more, amplify the value drivers typically seen from applying Smarter Process.

Figure 1-5 helps to identify the patterns and set of use cases to enable with Mobile Smarter Process. Mobile Smarter Process patterns provide a starting point that organizations can use to identify the patterns and use cases that apply to them and will provide higher business value. The patterns are similar to business process flows in concept; each represents a repeatable, proven set of value-producing actions.

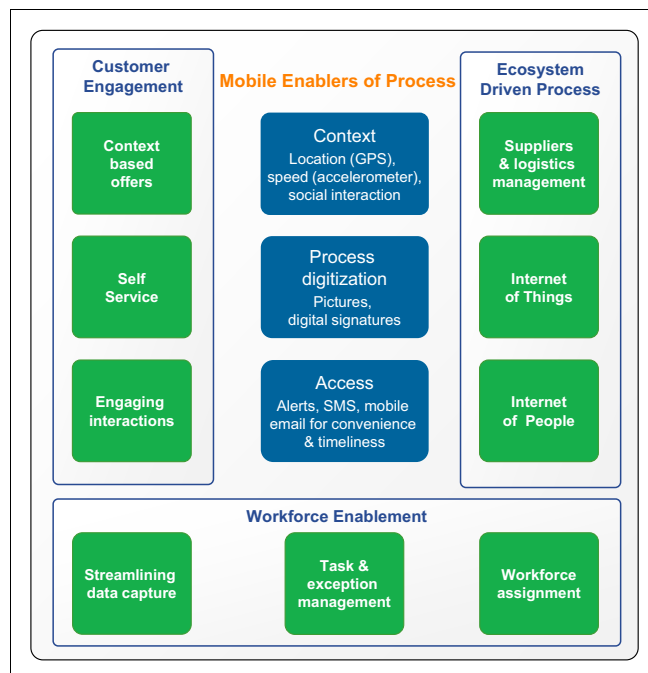


Figure 1-5 Patterns, use cases, and mobile enablers

Figure 1-5 shows the three use case patterns introduced in 1.3.3, “Identifying mobile business usage patterns” on page 12:

- ▶ Customer engagement
- ▶ Workforce enablement or empowerment
- ▶ Ecosystem-driven processes

Underneath these patterns, the figure shows some specific use cases, specifically targeted for the ideal scenarios for mobile-enabled Smarter Process. They are *ideal* scenarios

because the important point to keep in mind is that all the capabilities provided by Mobile Smarter Process do not amount to much unless the organization targets a good use case for it. In other words, not everything has to be mobile-enabled, can be mobile-enabled, and with certainty, not everything is going to deliver value and benefit by leveraging Mobile. However, the use cases shown in Figure 1-5 on page 13 for each Mobile Smarter Process pattern are the typical ones that organizations can target to deliver value to the business. For example, considering the airline scenario described in 1.3.3, “Identifying mobile business usage patterns” on page 12, the use cases that apply are:

- ▶ Context-based offer (Customer engagement)
- ▶ Engaging interaction (Customer engagement)
- ▶ Suppliers and logistic management (Ecosystem driven process)
- ▶ Task and exception management (Workforce enablement)

In real life, organizations are able to identify several patterns and look for the use cases that can truly drive the value and benefit using Mobile Smarter Processes.

The center of the figure shows the enablers of Mobile that can be applied to the use case patterns identified. For example:

- ▶ Context aware process: The process must be able to understand and respond to mobile contexts like location of team members or customers. Geolocation dictates what information and services the individual should receive. For example, assign a task to the closest team member or trigger an offer based on customer location.
- ▶ Process digitization: Provides insight into customer behaviors that was never before available. The process must be able to leverage data obtained from mobile device interaction and use. For example, capture data from the devices, such as digital signatures, pictures of a car damage, a bar code, using the camera to take pictures associated with a home study (for example child services, adoption) or an appraisal (for example, mortgage and banking). Using sensors on parts or containers to trigger inventory management processes.
- ▶ Access: Provide access any time using SMS, push notifications, email alerts for convenience and timeliness.

Applying the mobile enablers to the use case patterns leads to *value drivers and benefits* that come from combining Mobile and Smarter Process.

Back to the airline scenario, effectively the value and benefits delivered are:

- ▶ Customers receive instant information when a flight is canceled or delayed or a gate is changed.
- ▶ Timely actions.
- ▶ Continuous engagement from a quality perspective.
- ▶ Reduced capital expense from a cost perspective.

Using Figure 1-5 on page 13, organizations can quickly target the kind of use cases and patterns they want to enable with Mobile and Smarter Process.

1.3.5 Mobile Smarter Process transformation: Lessons learned in the trenches

This section includes a set of procedures and guidelines derived from IBM experience working with customers in their journey to transform their business processes to Mobile Smarter Processes.

Table 1-1 lists the foundational procedures that are considered most effective for those organizations starting their journey to Mobile Smarter Process.

Table 1-1 Foundational procedures

Procedure or Guideline	Description
Be comprehensive and bold	<ul style="list-style-type: none"> ► Identify areas where Mobile can <i>fundamentally change business processes and models to generate new revenue streams, lower costs, or redefine the organization's role in the value chain</i>. Top use cases in Retail, Life Sciences, Banking, Insurance, Airline, Government. ► Ensure that your enterprise mobile strategy considers both external as well as internal initiatives. ► Ensure that business cases consider both direct costs as well as benefits that accrue in areas not directly bearing the cost of the initiative.
Be targeted and iterative	<ul style="list-style-type: none"> ► Pilot new capabilities with selected customers, employees, and business partners to gain insights, reduce risk, and make adjustments before rolling out at an enterprise level. ► Assess the network structure to make sure that new initiatives can scale when needed. ► Rapidly adjust mobile offerings on an ongoing basis.
Be collaborative and inclusive	<ul style="list-style-type: none"> ► Make sure that your governance team involves all relevant stakeholders, including Line of Business, Information Technology, HR, and Marketing. ► Incorporate insights from <i>digital natives</i>. ► Look for opportunities to leverage common technology tools, partnerships, platforms, and development resources across business.
Identify processes where mobility can provide faster, higher value customer response	<ul style="list-style-type: none"> ► Reinvent existing processes to take advantage of mobile capabilities, such as self-service. ► Provide mobile personnel with tools that can provide rapid answers to customer assistance. ► Enable employees to collaborate more easily and effectively with experts across the organization.
Use mobile capabilities to facilitate routine <i>tasks</i> and <i>inquiries</i>	<ul style="list-style-type: none"> ► Provide employees with a mobile environment to accomplish administrative tasks at convenient times. ► Redesign internal processes, such as expense management and time and reporting to take advantage of mobile features. ► Consider adapting employee learning events and process changes to include mobile platforms.
Build security and privacy safeguards for employees using mobile devices for work-related activities	<ul style="list-style-type: none"> ► Develop clear rules and processes for mobile device use. ► Educate employees to minimize potential risks, including security violations, malware, and other IT hazards. ► Include extended workforce into <i>bring your own device</i> (BYOD) guidelines (for example, contractors, business partners).

Organizations that have already embarked in the transformation journey should adopt the procedures that are listed in Table 1-2 on page 16 to accelerate progress and attain higher value.

Table 1-2 Accelerate procedures

Procedure or Guideline	Description
Tap into the value of <i>scalable APIs</i>	<ul style="list-style-type: none"> ▶ Identify opportunities to consume third-party APIs to improve mobile applications. ▶ Attract and educate third-party developers to use APIs. ▶ Develop competencies in API development, promotion, and platform management to create new revenue streams.
Use <i>analytics</i> to provide end users and consumers with contextually relevant actions	<ul style="list-style-type: none"> ▶ Use mobile data to yield contextual information regarding the timing and location of mobile transactions. ▶ Apply analytics to understand how individuals are using their mobile applications and how these apps can be made more effective. ▶ Combine mobile data with transaction data to determine next best offers and actions.
Monitor emerging <i>cloud</i> services for opportunities to increase mobile engagement	<ul style="list-style-type: none"> ▶ Evaluate cloud capabilities to increase engagement through notifications or location-based services. ▶ Consider cloud delivered mobile lifecycle solutions if they meet requirements for security and fit with existing business constraints. ▶ Examine potential for cloud-based collaboration among mobile DevOps teams.
Incorporate machine-to-machine and sensor-based capabilities into the mobile strategy	<ul style="list-style-type: none"> ▶ Identify opportunities posed by embedding mobile capabilities into devices and sensors. ▶ Take advantage of data and insights being generated from these sensors within existing business processes. ▶ Consider new and emerging revenue streams and business models based on the data.
Invite trusted partners into your mobility strategy	<ul style="list-style-type: none"> ▶ Bring partners, customers, and other third parties early into the strategy development process. ▶ Identify opportunities for mutual gain and innovation today and in the future. ▶ Work jointly to identify future trends and technologies.
Plan for the future	<ul style="list-style-type: none"> ▶ Identify the skills and capabilities that will be needed to be competitive in the next three to five years. ▶ Highlight key investments that will be needed to sustain differentiation over the long term. ▶ Monitor key regulatory and policy developments to ensure that your organization is addressing future legal requirements.

Table 1-3 lists the summary considerations and key questions organizations should ask themselves.

Table 1-3 Summary considerations and key questions

Summary	Key questions to consider
Developing a mobile strategy	<ul style="list-style-type: none"> ▶ To what extent does your company have a well-defined enterprise-wide mobile strategy? How does your mobile strategy compare to your competitors? ▶ Who is involved in your mobile governance structure, for example, IT, marketing, line of business, HR, finance, accounting and how effectively do the functions work together on mobile strategy and execution? ▶ How does mobility drive business model innovation at your company?

Summary	Key questions to consider
Improving the customer experience and driving employee productivity	<ul style="list-style-type: none"> ▶ How can employees with mobile devices enable faster response time to customers and improve overall productivity? ▶ Which existing internal processes could be modified to take advantage of mobile capabilities? ▶ To what extent does your organization have a BYOD strategy? ▶ How effective are your policies and procedures in place for employee use of mobile devices?
Enabling the mobile enterprise	<ul style="list-style-type: none"> ▶ How effective is your mobile design and development process in understanding customer needs and rapidly developing solutions that fill those needs? ▶ To what extent are legacy systems a barrier for your organization in connecting mobile systems? ▶ How do you think mobile analytics can help you in providing a measurable return?

1.3.6 Mobile technology adoption patterns

Figure 1-6 shows the three main buckets that organizations fall into in their journey to mobile-enable their processes. These buckets can also be thought of as adoption patterns.

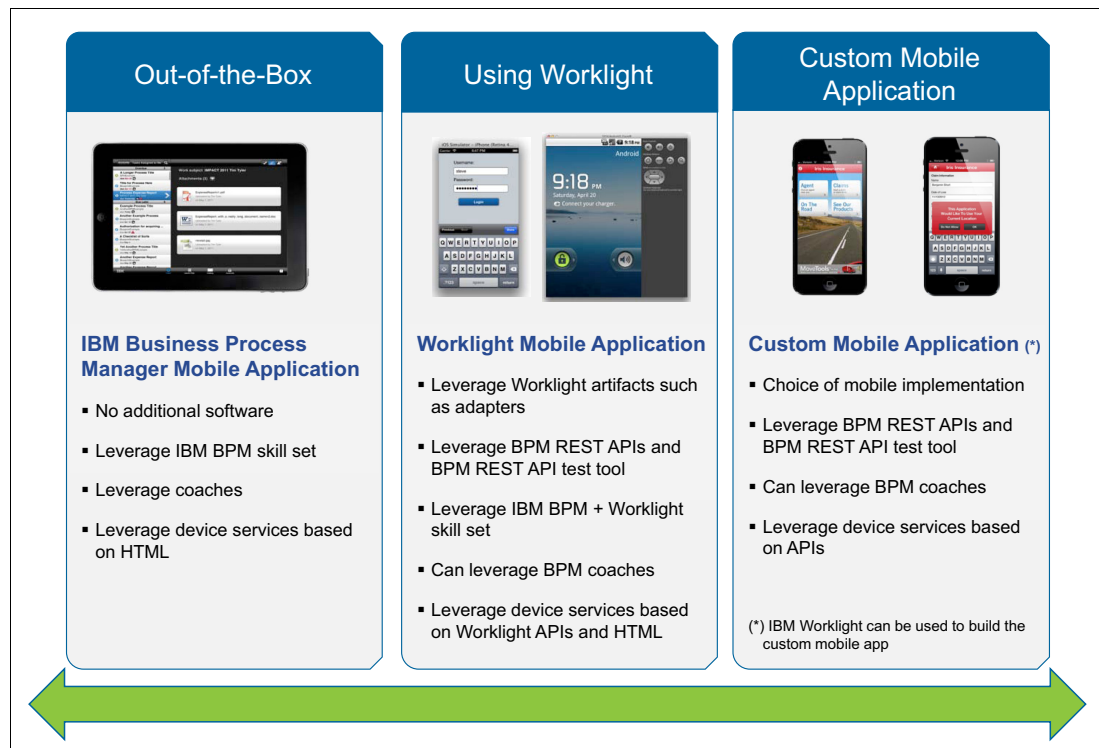


Figure 1-6 Mobile-enabling process: Typical adoption patterns

The typical adoption patterns shown in Figure 1-6 from left to right are:

- ▶ **Out-of-the-box:** IBM Business Process Manager mobile application

Organizations have leveraged IBM Business Process Manager processes effectively in traditional environments and now they want to deliver a mobile access to their processes

using IBM BPM out-of-the-box capabilities with no additional software or development effort. They want to leverage:

- IBM BPM skill set in the organization.
- The value proposition delivered with the IBM BPM product, such as:
 - Coach forms (IBM BPM UI framework).
 - IBM iOS app for IBM Business Process Manager.
 - Device services based on HTML, location, ability to take photos with the device camera and upload the pictures.

► Using IBM Worklight: Worklight Mobile Application

Organizations are looking to leverage or deliver a hybrid approach between out-of-the-box and custom mobile app. They still want to leverage IBM BPM as they do today via a portal or web interface using device services from a mobile device but they need to write and maintain a mobile application to do some additional work such as security and app management, or add additional capabilities such as working offline.

This adoption pattern is the sweet spot for combining IBM Business Process Manager and IBM Worklight. Developing a mobile app provides more flexibility, security, and efficiency than the out-of-the-box pattern. It leverages the IBM BPM and IBM Worklight skills in the organization.

The following features of IBM Worklight and IBM Business Process Manager can be leveraged:

- Worklight adapters
- Worklight APIs
- IBM BPM coaches and playback
- IBM BPM REST API
- Non-production license of IBM Worklight Enterprise Edition bundled with IBM Business Process Manager V8.5 Standard and Advanced for app development

► Custom mobile application

Organizations are building custom mobile applications; they may or may not use Worklight as the app development platform. Usually user interface standards for mobile applications are strictly governed and controlled by the mobile development team or determined by the organization's budget.

The best feature that IBM BPM has to offer for this adoption pattern is its rich IBM BPM REST APIs that are secured and deliver the kind of information that in this case the mobile application needs. The IBM BPM REST API test tool helps developers learn about this set of REST APIs, and to test them.

The mobile app can still leverage some of the key value provided with IBM BPM, such as coaches but typically the UI is done and delivered by the custom mobile app. Similarly, leveraging device services such as using the GPS or the camera is done by the mobile app itself.

In summary, no matter where organizations are on their adoption or their journey with Mobile, Mobile Smarter Process can offer a very good fit into their environment.

1.4 IBM BPM features to mobile-enable processes

Section 1.3.6, “Mobile technology adoption patterns” on page 17 provides a quick list of IBM BPM features that can be leveraged to mobile-enabled processes for each mobile adoption pattern. This section provides more information about each of the features. For an overview of IBM Business Process Manager and the new features in V8.5.5, see Chapter 3, “IBM Business Process Manager V8.5.5 overview” on page 53.

1.4.1 IBM BPM mobile app for iOS

This app provides an out-of-the-box artifact for mobile interaction with IBM BPM processes from Apple iPhone, iPad, or iPod touch devices. It allows users to interact with others to ensure that the right people are working on the right thing at the right time. Using this app, business users can:

- ▶ Review file attachments for additional context or approval.
- ▶ Attach photos to an existing task.
- ▶ View and complete tasks right from your mobile device.
- ▶ Launch new processes remotely.

Figure 1-7 shows a view of the IBM BPM app for iOS. Simple task forms and intelligent coaches guide user inputs.

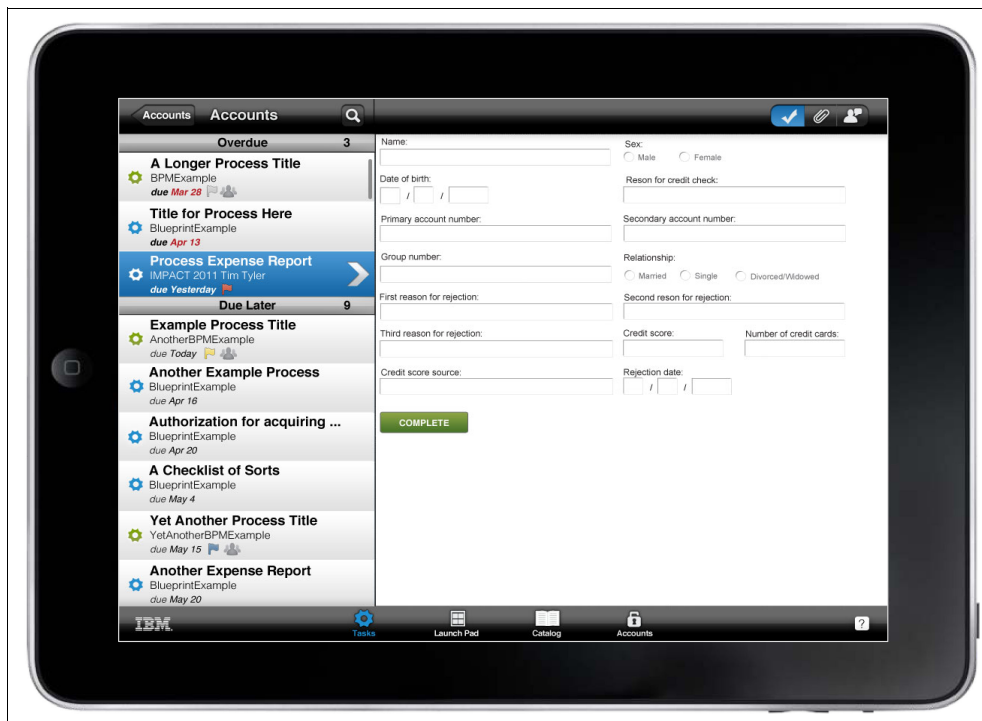


Figure 1-7 IBM BPM app for iOS

This app requires access to an existing IBM Blueworks Live™ account or IBM Business Process Manager server (running either on-premises or on the IBM SmartCloud® Enterprise). The app can be downloaded from the iTunes store at the following location:

<https://itunes.apple.com/us/app/id514082860?mt=8>

1.4.2 IBM Business Process Manager REST Interface

IBM BPM provides a set of APIs that are implemented using Representational State Transfer (REST) services. A set of REST resources is provided for accessing business process, human task, work basket, or business category data.

The IBM BPM REST Interface is core and foundation to the main UIs for the product itself, such as the IBM BPM iOS app and the Process Portal, which is the out-of-the-box UI to IBM BPM itself. The IBM BPM REST API have been tested and improved time and time again. Refer to *Developing client applications that use IBM BPM REST APIs* at:

http://www-01.ibm.com/support/knowledgecenter/SSFPJS_8.5.5/com.ibm.wbpm.bpc.doc/to/pics/cdev_restapis.html

IBM BPM REST API test tool

A test tool is provided with the IBM BPM REST APIs. Mobile app developers can use this tool to learn about the IBM BPM REST APIs, and to test those APIs that they are planning to use in their application. Figure 1-8 shows the IBM BPM REST API tester.

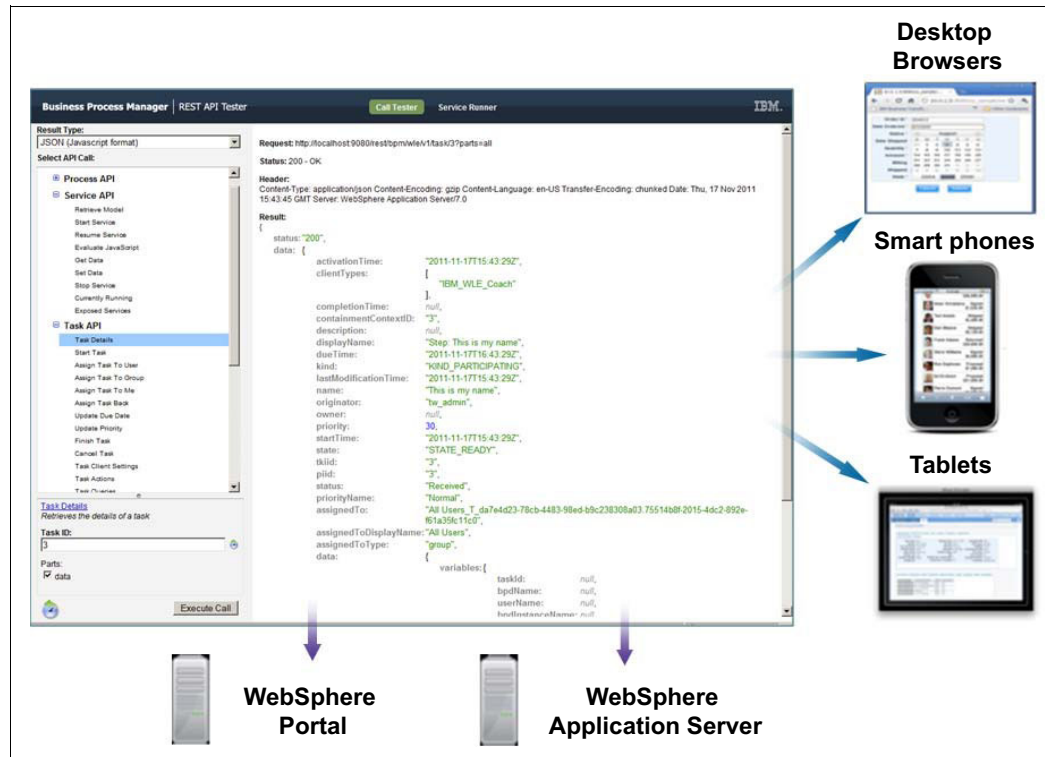


Figure 1-8 IBM BPM REST API test tool

For more information, refer to *Testing REST APIs* at the following location:

http://www-01.ibm.com/support/knowledgecenter/api/content/SSFPJS_8.5.5/com.ibm.wbpm.bpc.doc/topics/tdev_testingrestapis.html

1.4.3 Non-production entitlement to Worklight Enterprise Edition with IBM BPM V8.5 and later

IBM Business Process Manager Standard V8.5 and IBM Business Process Manager Advanced V8.5 and later bundle a non-production license of IBM Worklight Enterprise

Edition. Mobile app developers can use the non-production license of IBM Worklight Enterprise Edition to test and develop Worklight applications with IBM BPM. Developers can use all capabilities of Worklight Enterprise Edition to this extent. For a description of the IBM Worklight capabilities, refer to Chapter 2, “IBM Worklight Foundation V6.2 overview” on page 31.

Using the capabilities of non-production entitlement to IBM Worklight organizations can:

- ▶ Develop and test custom IBM BPM mobile applications using Worklight.
- ▶ Leverage the IBM BPM APIs and coaches.
- ▶ Accelerate development of customer-ready mobile experiences.
- ▶ Leverage business decisions from mobile devices.

The license does not entitle you to production use of the mobile applications that are developed. This statement means that a separate license needs to be purchased to use any application that is built using this entitlement in production.

All mobile applications that are developed using this entitlement must be used with IBM Business Process Manager V8.5 (or later) Standard or IBM Business Process Manager Advanced V8.5 (or later). You must purchase a separate Worklight license to support stand-alone mobile application development.

For more information, refer to *Understanding the IBM Business Process Manager (BPM) and IBM Worklight bundle* at the following site:

<http://www-01.ibm.com/support/docview.wss?uid=swg27038107#toc06>

1.4.4 Mobile app development accelerators

Artifacts are provided in the IBM BPM Community — SAMPLE EXCHANGE at the following site:

<http://wiki.bpmwiki.com/display/samples/SAMPLE+EXCHANGE+HOME>

Note: You have to register with the IBM BPM Community to access these links. There is no charge to register and join the community.

You can find links to the IBM BPM Mobile artifacts grouped under the following URL:

<http://wiki.bpmwiki.com/display/samples/Mobile+apps>

These artifacts aim at accelerating the development and delivery of mobile apps for IBM BPM processes by including examples that show:

- ▶ Location/Map
- ▶ Camera
- ▶ Mobile view layout
- ▶ UI controls
- ▶ Worklight BPM Adapter to call IBM BPM REST APIs

The *Lite Coaches toolkit for BPM 8.5.5* is a *technology demonstration* of mobile-optimized coach views for IBM Business Process Manager V8.5.5. You can use the coach views in the toolkit to develop responsive coaches that are suitable for both desktop and mobile platforms. This asset is targeted to deliver a faster, more responsive mobile experience and coach views that look native to the device. For more information see “Lite coach views” on page 24.

1.4.5 Client-side human services

Client-side human services are a new feature added in IBM BPM V8.5.5. When you use client-side human services, you can use web technology to improve human-service performance and provide support for case management and process management. You create and edit client-side human services in the Process Designer web editor, run them on the client-side in the web browser, and use them to call the server for data when necessary.

When you build a client-side human service, you use coaches, client-side scripts, services, and events to create a service flow that is run, tested, and refined entirely in a web browser. Enhanced authoring capabilities such as WYSIWYG (what you see is what you get) and responsive design elements help you build user interfaces for case and process instances and ensure scalability for mobile devices.

With client-side human service, the focus is on reducing the back and forth interaction between the client and the server to get the data. The approach is to cache an entire piece of the whole process, known as human services, on the actual mobile router. This approach helps deliver a quicker response and reduce server calls. The objective is to improve performance. It enables more concurrent users, reduces the amount of data that is transferred over the network and these are all key requirements for mobile.

Figure 1-9 provides a graphical representation of the client-side human services feature.

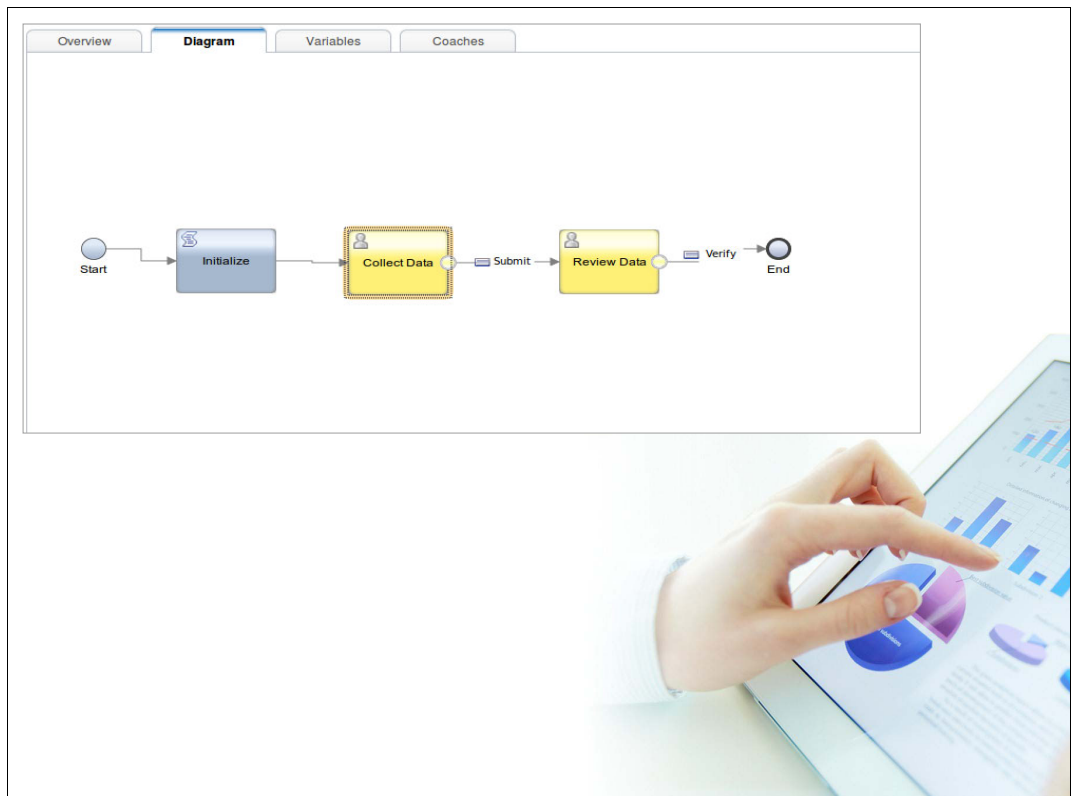


Figure 1-9 Client-side human services to deliver faster response

The main characteristics of client-side human services are:

- New human services architecture that improves performance, improves responsiveness, and reduces the amount of data transferred over the network.

- ▶ Cached and executed in a browser. Offloads server processing by shifting work to client browser (for example, JavaScript processing, validation).
- ▶ Dynamic data is not cached for security.
- ▶ Optimized network trips to server by caching static content, including human services model and generated coach code.
- ▶ Enables more concurrent users to access the IBM BPM server.
- ▶ Acts as a stepping stone for offline processes.
- ▶ New web-based and human services and coach editor. Web-based human services editor launched from Process Designer.

Comparing client-side human services with heritage human services

In IBM BPM V8.5.5, there are two kinds of human services:

- ▶ Heritage human services: They were available in IBM BPM versions pre-IBM BPM V8.5.5. They were called simply *human services*.
- ▶ Client-side human services (as described in 1.4.5, “Client-side human services” on page 22): New in IBM BPM V8.5.5. Their main features are:
 - New Architecture: Runs entirely on the client-side with occasional communication with the server.
 - Better Performance:
 - Leverages browser caching. The Client Side Human Service (CSHS) model and generated coach HTMLs are cached when running against a snapshot.
 - Fewer server calls, therefore network latency is less of an issue.
 - Coach-to-Coach transitions do not require a trip back to the server.
 - Less load on the server. Work is distributed across the client machines.

Table 1-4 compares heritage human services and client-side human services.

Table 1-4 Heritage human services versus client-side human services

Features and capabilities	Heritage human services	Client-side human services
Authoring	Eclipse editor	Web editor (WYSIWYG)
Runtime	On the server	Client-side, in the web browser, with calls for data to the server only when necessary
Case management support	Not available	Available
Collaboration	Supported	Not supported
Supported coach types	Available for both new coaches and heritage coaches	Available for new coaches only. Heritage coaches are not supported.
Usage of JavaScript	Server scripts that use server script syntax	Client-side scripts that use standard JavaScript syntax

1.4.6 Responsive coach design

Responsive coach views adapt to different form factors. You can build user interfaces that are responsive to the user’s runtime environment by using the new responsive settings for coach views available with IBM BPM V8.5.5. You can configure properties such as visibility,

formatting, or presentation style for different screen sizes so that you can design one interface that changes in appearance and behavior based on the user's screen size.

The same code is rendered on Worklight mobile apps, out-of-the-box default IBM BPM UIs, mobile browsers, or even is embedded in other applications such as IBM Notes and IBM Connections.

Figure 1-10 provides an overview of the environments where a single dynamic coach view can be displayed.

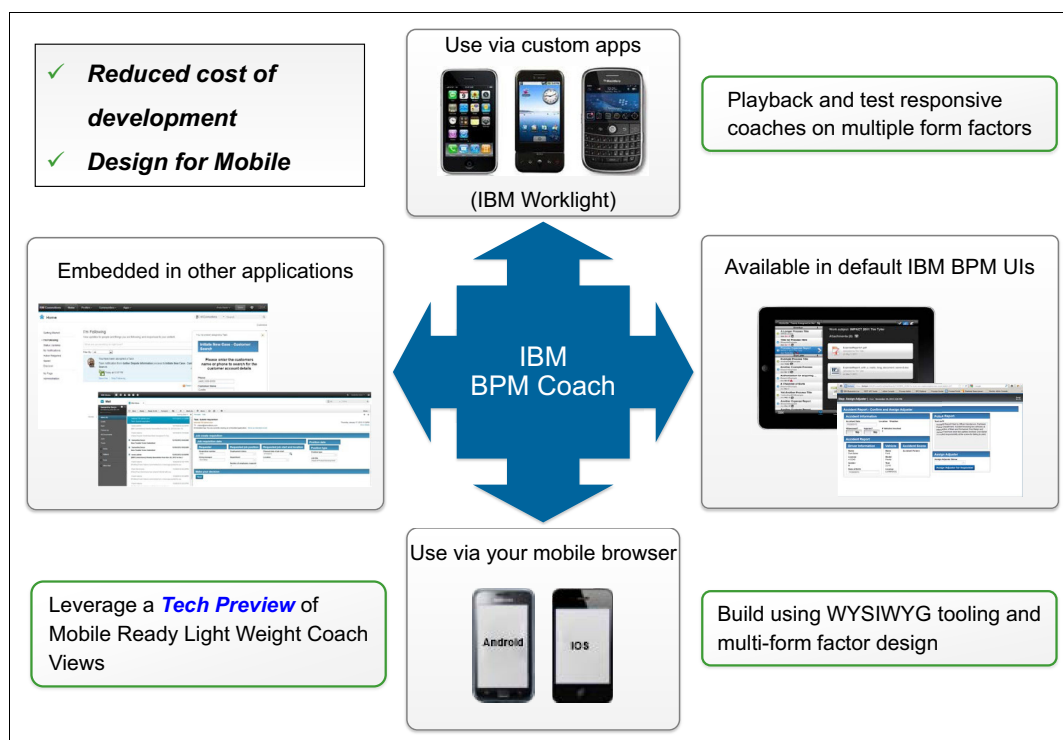


Figure 1-10 Responsive coach design: Single coach for multiple form factors

The responsive coach design feature allows you to:

- ▶ Design responsive coaches for use on multiple form factors.
- ▶ Playback coaches before deployment.
- ▶ Make mobile part of coach design.
- ▶ Leverage responsive coaches in any external user interfaces including IBM Worklight and custom mobile applications.

Lite coach views

The stock controls in IBM BPM V8.5.5 are best used on browsers and tablets, but not on smartphones. In fact, they are not supported on Android or iPhone. The IBM BPM V8.5.5 stock controls are supported on desktop or notebook browsers and the iPad Safari browser.

The stock controls are not fully responsive though. Although they can resize somewhat based on the form factor, they cannot adapt to the native look and feel on the target devices.

A new set of coach views has been released as a technology demonstration. They are known as the *Lite coaches*.

Unlike the stock controls in the standard coaches toolkit, the Lite controls in the Lite coaches toolkit are not Dojo-based. These Lite controls use a combination of HTML5, JavaScript, AngularJS, Angular UI-Bootstrap, and Cascading Style Sheets (CSS) to provide a responsive presentation that results in an optimal mobile and desktop user experience. They offer a more native look and feel with optimized performance on mobile devices.

The key benefits are:

- ▶ The responsive controls in the Lite coach views provide a native device look and feel and they render well on all screen sizes.
- ▶ Deliver crisper graphics for Retina displays.
- ▶ Render faster, less JavaScript on device.
- ▶ Provide native HTML controls such as date picker.
- ▶ Non-Dojo lightweight coach views targeted to deliver a more responsive experience on mobile devices.
- ▶ Combined and minified JavaScript and CSS files.
- ▶ Optimization to run on mobile networks (with a minimal number of network requests and minimized size of data transferred over network to optimize bandwidth usage).
- ▶ Testing and support for mobile form factors.
- ▶ Accelerated development of client-ready mobile experiences.
- ▶ No overhead for rarely used capabilities like non-Gregorian calendar.
- ▶ Input controls that look native to the device.

The Lite Coaches toolkit for BPM 8.5.5 (SAMPLE EXCHANGE) can be downloaded from the IBM BPM Community at the following URL:

<http://wiki.bpmwiki.com/display/samples/Lite+Coaches+toolkit+for+BPM+8.5.5>

Figure 1-11 on page 26 shows how the UI components are delivered with a native look and feel on an iPad when using the responsive controls.

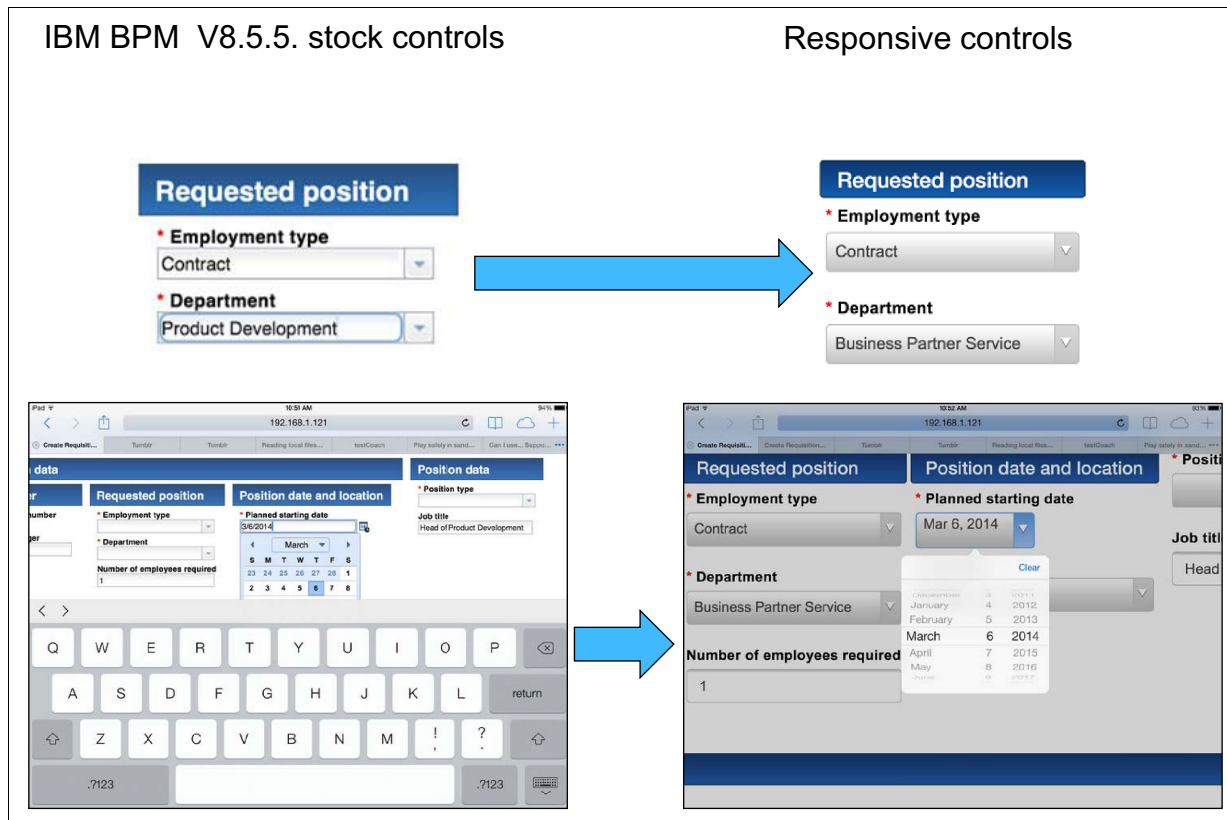


Figure 1-11 Rendering UI elements: IBM BPM V8.5.5 stock controls versus responsive controls

1.5 IBM Worklight features to mobile-enable processes

This section provides an overview of the features in IBM Worklight that help to mobile-enable processes. For an overview of IBM Worklight and the new features in V6.2, see Chapter 2, “IBM Worklight Foundation V6.2 overview” on page 31.

1.5.1 Worklight adapters

You can use IBM Worklight to create custom mobile applications that integrate with IBM BPM processes and tasks. The primary interface from IBM Worklight is an adapter, which is a transport layer used by the IBM Worklight Platform to connect to various back-end systems.

Adapters are the server-side code of applications that are deployed on and serviced by the IBM Worklight Mobile Application Platform. These are the standard adapter types:

- ▶ SQL Adapter
- ▶ HTTP Adapter. This is the adapter type that was used in the examples of this book to invoke the IBM BPM REST APIs.
- ▶ IBM Cast Iron® Adapter
- ▶ SAP Netweaver Gateway Adapter
- ▶ JMS Adapter

Adapters connect to enterprise applications (also referred to as *back-end systems*), deliver data to and from mobile applications, and perform any necessary server-side logic on this data.

When integrating with IBM BPM, an IBM BPM adapter (which is delivered as sample code in IBM BPM V8.5.5) connects to the IBM BPM REST APIs for performing actions, such as starting a process, retrieving a list of tasks, retrieving a coach, viewing the user profile, commenting on a process instance, and completing tasks.

1.5.2 IBM Worklight Studio

IBM Worklight Studio (based on Eclipse) is the development environment for IBM Worklight. For more information in Worklight Studio, see 2.1.1, “Worklight Studio” on page 35.

When integrating your Worklight application (and its adapters) with IBM BPM, the adapter you use to connect to IBM BPM REST APIs should be configured to use the fully qualified host name of your IBM BPM Process Server. For the port, use the same port used by IBM Process Portal (the default port is 9080 for HTTP and 9443 for HTTPS). IBM Worklight Studio provides an embedded server based on Jetty.

IBM BPM coach forms can be used in mobile apps developed in IBM Worklight Studio. For example, IBM BPM REST APIs can be invoked to retrieve the coach and include the display of the coach for a human service inside the IBM Worklight application. The coach form is imbedded in an iFrame.

1.5.3 Command-line interface for IBM Worklight developers

IBM Worklight Foundation provides command-line tools as an alternative to the integrated development environment (IDE) of Worklight Studio.

To help developers get a better tools experience, IBM Worklight Foundation provides a command-line interface (CLI) tool to easily create and manage both native and hybrid apps. The CLI enables developers to use their preferred text editors or alternative IDEs to create mobile applications.

The command-line interface does not require Worklight Studio for most standard activities. The commands support tasks such as creating, adding, and configuring with the Worklight API library, adding the client-side Worklight properties file and performing the build and deploy of the Worklight application. Adapter creation, deployment, and local testing can be performed within the command line. Administration of the Worklight project can be done from CLI or REST services, or the Console, where developers can easily control the local server and observe the logs. Command-line tools can be used on their own, or in parallel with the Worklight Studio tools.

1.5.4 Worklight APIs

Worklight client-side API provides capabilities to improve application development, and Worklight server-side API provides capabilities to improve client/server integration and communication between mobile applications and back-end systems such as IBM BPM.

With the Worklight client-side API, mobile applications have access to various Worklight features during run time, by using libraries that are bundled into the application. The libraries integrate mobile applications with the Worklight Server by using predefined communication interfaces.

The Worklight Server provides a set of mobile capabilities with the use of client/server integration and communication between mobile applications and back-end systems. You can develop server-side application code and optimize performance, security, and maintenance. By developing server-side application code, mobile applications have direct access to back-end transactional capabilities and cloud-based services. This capability improves error handling, and enhances security by including more custom steps for request validation or process authorization.

1.5.5 Location services (geolocation)

Geolocation is a powerful differentiator of mobile apps. However, because geolocation coordinates must be constantly polled to understand where a mobile device is located, the resulting stream of geographic information can be difficult to manage without exhausting resources such as battery and network.

IBM Worklight includes location services that handle multiple geo modalities such as GPS, WiFi sampling, and interpolation. You can set policies for acquiring geolocation data and transmitting it to the server in order to optimize battery and network usage.

With location services, the data that is acquired by a mobile device can be used to trigger processes that benefit both the owner of the device and the enterprise that receives the data as demonstrated by the airline example described in 1.3.3, “Identifying mobile business usage patterns” on page 12.

1.5.6 Push notification

Push notification is the ability of a mobile device to receive messages that are pushed from a server, for example a step in a process that requires to provide timely notification to the user. The response from the business process can be delivered back to the mobile user using push notifications. The most common form of notification is Short Message Service (SMS). Notifications are received regardless of whether the application is currently running.

1.5.7 Worklight security and LTPA

Worklight has comprehensive support for various authentication and authorization methods, including Lightweight Third-Party Authentication (LTPA). This feature helps to secure mobile applications.

Use a common LDAP directory as a user registry supporting single sign-on. Use the LTPA authentication login module provided with IBM Worklight to allow IBM Worklight to authenticate to the IBM BPM server when the user logs in to the Worklight mobile application. This approach provides a secure channel for access to the IBM BPM REST APIs.

Worklight provides secure, end-to-end communication by positioning a server that oversees the flow of data between the mobile application and the back-end systems. IBM Worklight provides the possibility to define custom security handlers for any access to this flow of data. Because any access to data of a mobile application has to go through this server instance, you can define different security handlers for mobile applications, web applications, and back-end access. With this kind of granular security, you can define separate levels of authentication for different functions of the mobile application or avoid sensitive information being accessed from a mobile application entirely.

1.5.8 Offline-enabled mobile applications

In terms of connectivity, mobile applications can operate offline, online, or in a mixed mode. IBM Worklight uses a client/server architecture that can detect whether a device has network connectivity, and the quality of the connection. Acting as a client, mobile applications periodically attempt to connect to the server and to assess the strength of the connection. An offline-enabled mobile application can be used when a mobile device lacks connectivity but some functions can be limited. For offline-enabled mobile application, it is useful to store information about the mobile device that can help preserve its functionality in offline mode. This information typically comes from a back-end system.

1.5.9 Operational analytics

The operational analytics feature enables searching across apps, services, devices, and other sources to collect data about usage or detect problems. The analytics feature enables enterprises to search across logs and events that are collected from devices, apps, and servers for patterns, problems, and platform usage statistics. For more details, see Chapter 8, “Deeper insight through IBM Business Process Manager and IBM Worklight analytics” on page 283.

1.6 Conclusion

Organizations are designing their enterprise mobility strategy to improve customer experience, drive employee productivity and most importantly, deliver new and innovative value propositions in the marketplace. The mobile strategy of an organization must address:

- ▶ Where within the organization mobile technologies can offer the greatest benefits.
- ▶ How organizations can enable their IT functions to develop mobile solutions more efficiently.
- ▶ What segments of the workforce need to be *mobilized* to achieve the greatest return on investment.

It is important to prioritize the mobile efforts and integrate with existing investments and infrastructure of which the processes of an organization are a key component.

The capabilities in Mobile Smarter Process described in this chapter aim at helping organizations to improve the time-to-value to mobile-enable their processes. They provide options ranging from out-of-the-box artifacts that can be used with no additional software to features and enhancements that help to reduce the mobile app development cycle, reduce the development cost, improve the efficiency of mobile apps, and provide an enjoyable customer experience.

It is not just about process orchestration and automation and building mobile apps, but it is about:

- ▶ Developing a sound mobile strategy.
- ▶ Harnessing the mobile interactions with a well designed process to deliver real value and benefits from Mobile.
- ▶ Simplifying and reinventing processes (designing processes for Mobile).
- ▶ Monitoring and managing the process.
- ▶ Managing and securing the apps.

- ▶ Developing and testing the apps for multiple form factors.
- ▶ Deploying the apps to the appropriate stores.
- ▶ Analyzing the apps and processes for continuous improvement.

IBM Mobile Smarter Process combines market leading capabilities of both IBM Smarter Process and IBM MobileFirst to provide a very comprehensive suite ranked as a leader by analysts for both Intelligent Business Process Management suites and mobile application platforms. Mobile-enabled Smarter Process brings together the proven expertise of IBM.



IBM Worklight Foundation V6.2 overview

This chapter provides an overview about the IBM Worklight platform. It describes the main components, business value, and what's new in Worklight V6.2.

This chapter includes the following topics:

- ▶ 2.1, “Description of Worklight” on page 32
- ▶ 2.2, “Worklight in IBM MobileFirst” on page 39
- ▶ 2.3, “Business benefits” on page 40
- ▶ 2.4, “System topology” on page 41
- ▶ 2.5, “Components” on page 43

2.1 Description of Worklight

Worklight helps organizations extend their business to mobile devices. It provides an open and comprehensive platform to not only build, but test, run and manage native, hybrid and mobile web apps. Organizations can design, develop, and deploy mobile applications (apps) for multiple device types and operating systems with an intuitive set of tools, programming models, and systems.

Worklight enables mobile apps to embody the following characteristics:

- ▶ Developed for multiple device types and operating systems
- ▶ Secured and governed
- ▶ Given access to back-end services, applications, and data
- ▶ Monitored and analyzed for usage, trends, and feedback
- ▶ Deployed and managed in an enterprise

The Worklight platform includes a comprehensive set of tools, application programming interfaces (APIs), and runtime components. It is designed to help organizations build, run, and manage mobile apps, and it is part of the IBM MobileFirst Platform product suite.

Using Worklight, browser, hybrid, mixed, and native mobile apps (Figure 2-1) can be efficiently developed in a short time, giving businesses a competitive edge:

- ▶ Faster time-to-market
- ▶ Reduced development costs
- ▶ Enhanced mobile application governance and security

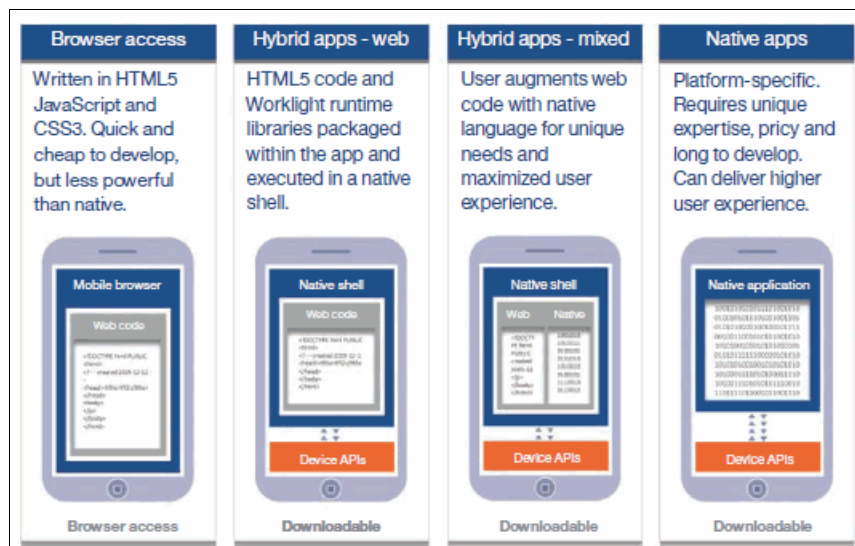


Figure 2-1 Mobile application implementations with Worklight¹

¹ From IBM Worklight 6.0 Technology Overview
<http://public.dhe.ibm.com/common/ssi/ecm/en/wsw14181usen/WSW14181USEN.PDF>

Remember: The following list describes application types:

- ▶ Mobile web apps are the same as those used in a desktop browser, but are adapted to fit on the smaller screen of a mobile device. They are typically written using HTML5, Cascading Style Sheets (CSS), and JavaScript, and are portable across multiple devices and operating systems. Web applications cannot access the hardware on the mobile device, such as the camera and the global positioning system (GPS).
- ▶ Native apps are written in an operating system-specific language, and have full access to the mobile device hardware, features, and functions.
- ▶ Hybrid apps combine both the web and native models.

Development activity is carried out in an environment that can address all supported mobile operating systems. Therefore, an app can be created that looks and feels the same across all deployed devices, but takes advantage of the native capabilities of the mobile device.

Alternatively, developers can use the Worklight command-line interface (CLI) to generate a native project and continue development using vendor-specific integrated development environments (IDEs) such as xCode, Android Studio, and Microsoft Visual Studio.

Worklight uses standards-based technologies to integrate directly with the various mobile platform software development kits (SDKs), avoiding the use of code translation, proprietary interpreters, and the less well-known scripting languages.

Worklight capabilities: With Worklight, businesses are able to accomplish the following tasks:

- ▶ Support multiple mobile operating system environments and devices.
- ▶ Connect and synchronize with enterprise data, applications, and cloud services.
- ▶ Safeguard mobile security at the device, application, and network layer.
- ▶ Securely share lightweight information among a family of applications on a single device.
- ▶ Govern the mobile application from one central interface.
- ▶ Test applications by recording and playing back a process or function on a mobile device or emulated mobile environment.

The five key components of Worklight are shown in Figure 2-2 on page 34.

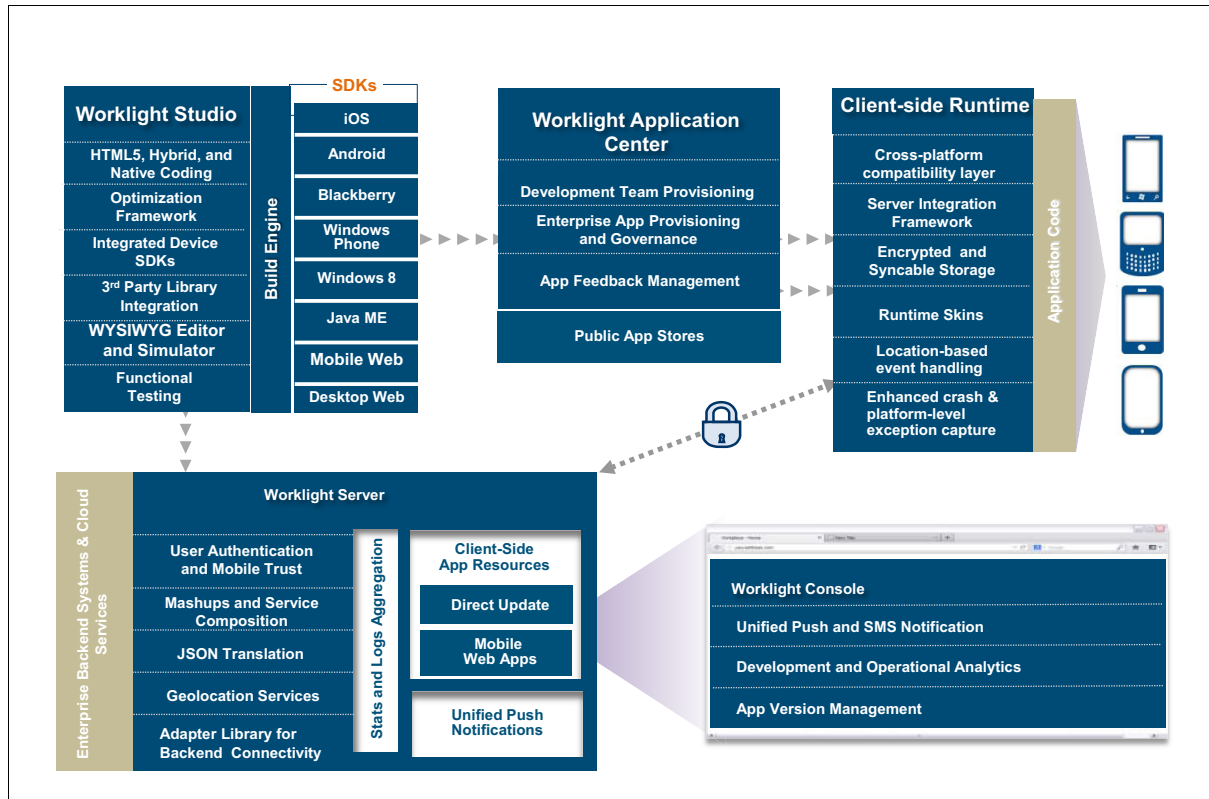


Figure 2-2 Worklight architecture overview and key components

The following components are shown in Figure 2-2:

- **Worklight Studio**

An Eclipse-based development platform designed for coding, testing, and integration tasks for web, hybrid, or native mobile apps. It interfaces with native tools, such as XCode and Android Studio.

- **Worklight Server**

A scalable mobile middleware that sits between the mobile app and the enterprise back-end services. It acts as an auditable control point for mobile devices, and provides a strong security layer.

Worklight Server contains multi-factor authentication and mobile app authenticity checking. It enables data connectivity for multi-source data extraction and manipulation, mobile application management, analytics, and runtime services, such as push notifications and geolocation capabilities.

- **Worklight device runtime components**

Client-side runtime code compiled into hybrid apps that embed functionality, such as offline, encrypted, and syncable data stores that interact with the Worklight Server.

- Worklight Application Center

A multi-platform enterprise application store to enable organizations to govern and distribute production-ready mobile apps. It has built-in access controls and role-based security, and can elicit and organize user feedback. It can also be used as part of the development lifecycle to distribute prerelease software to be analyzed by developers for feedback by version and device.

- Worklight Console

A web-based user interface for monitoring and administering the Worklight Server and its deployed applications. Adapters to connect to services and push notifications are also managed in the console. Furthermore, summaries of platform usage can be displayed on dashboards.

2.1.1 Worklight Studio

The Worklight Studio SDK and integrated development environment (IDE) offers simplified development of applications across multiple mobile platforms, including iOS, Android, Windows Phone, and Java Platform, Micro Edition (Java ME). The Worklight framework enables a high level of code reuse, and enables developers to deliver a rich and consistent user experience matching the capabilities of the device.

Worklight Studio features a drag-and-drop user interface (UI) for the design and development of the mobile app. Interfaces, such as JQuery Mobile, HTML, and JavaScript files are easily created by dragging HTML5 and Dojo Mobile components from a built-in palette onto the HTML work area, known as the *canvas*.

If the developer opts for a hybrid development style, the majority of application code can be shared across multiple mobile device environments, without altering the platform-specific user experience or application functionality. For example, an application can be developed once, with feature and function code unique to the iPhone held in a separate folder.

Common application code is stored in a shared folder, with device-specific or environment-specific code stored in isolated folders (see Figure 2-3 on page 36) that can overwrite or compliment the common shared code. As a result, application logic is consistent across the different environments, while the user experience behaves as expected, natively and geared to the functionality and design guidelines of the device.

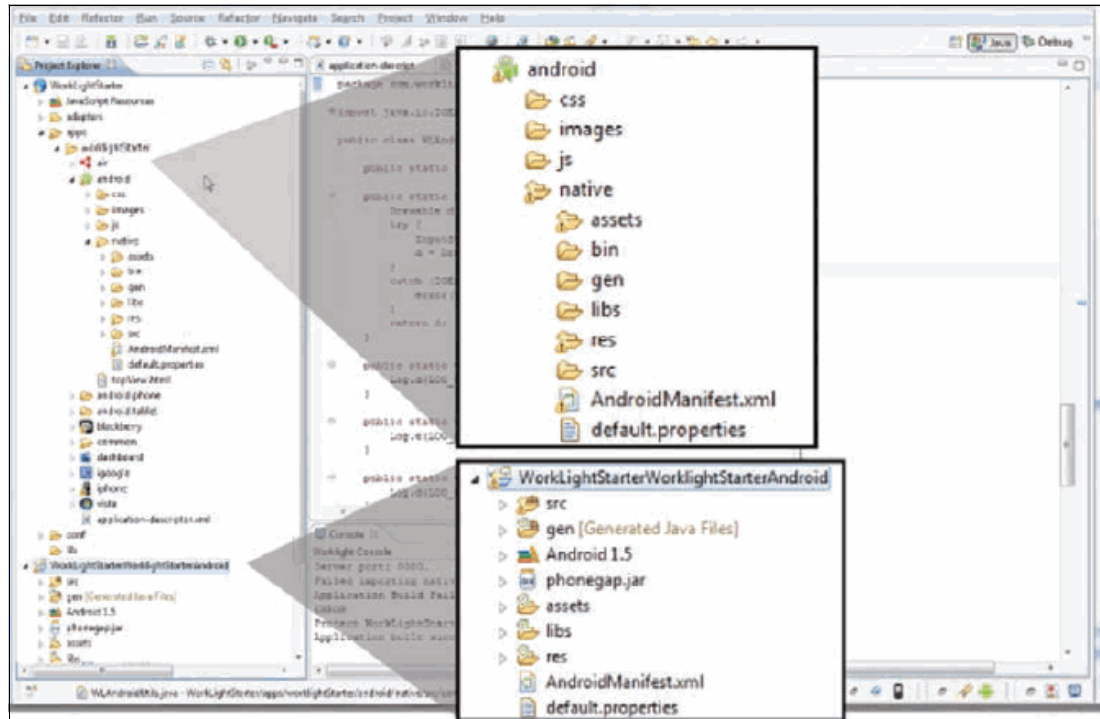


Figure 2-3 Multiple device-specific code stored in a single project²

When developing hybrid or native applications, application developers can directly access the APIs that the devices offer, and can easily integrate with third-party libraries, frameworks, and tools. Because developers are not constrained by intermediary build-time or runtime layers, when new APIs are released, they are accessible for use immediately.

Worklight Studio enables developers to choose how they develop applications. They can either use pure native code (Objective-C, Java, or C#), standard web technologies (HTML5, CSS3, JavaScript), or a combination of both.

For pure native development, Worklight Studio provides command-line tools. The command-line interface (CLI) supports tasks such as creating, adding, and configuring Worklight projects. This enables developers to use their preferred text editors or alternative IDEs to create mobile applications.

For hybrid application development, Worklight Studio supports the following scenarios:

- ▶ HTML to call native code using a plug-in, which can be used to perform non-user interactive functions, such as reading the compass, or to overlay a user interface on top of the browser.
- ▶ Implementation of complete screens, with switching between native screens and web screens made seamless with smooth or animated transitions.

The following list includes further development capabilities in Worklight Studio:

- ▶ Runtime skins, which enable applications to automatically adjust to different devices using the same operating system, such as different screen sizes, different screen resolutions, and different input methods.

² From IBM Worklight 6.0 Technology Overview
<http://public.dhe.ibm.com/common/ssi/ecm/en/wsw14181usen/WSW14181USEN.PDF>

- ▶ Geolocation toolkit, enabling the gathering and analysis of location data from GPS and WiFi efficiently and with minimal drain on the device.
- ▶ Screen templates to automate the creation of mobile screens, such as lists, authentication, navigation, search, and configuration.
- ▶ Standard data retrieval, enabling developers to access hierarchical data from many back-end database management systems, such as IBM DB2®, and convert to JavaScript Object Notation (JSON) for application use.
- ▶ The ability to invoke back-end services and applications, such as those running in IBM Business Process Manager, directly from within the Worklight Studio using JSON and Extensible Markup Language (XML).
- ▶ Unified push notifications to preconfigure automatic alerts from one centralized interface, making the communication with users completely transparent to the developer.

Worklight Studio includes Mobile Test Workbench for integrated and automated functional testing of Android and iOS native and hybrid apps. This feature enables developers and test teams to rapidly test Worklight applications, by recording a sequence of actions on a mobile device to generate a test script, which is then replayed on a real mobile device, or on an emulator/simulator. The results can be viewed and shared using a generated HTML report. Device-specific functions, such as camera, accelerometer, and compass, can be emulated and tested with the application.

2.1.2 Worklight Server

The WebSphere based Worklight Server is a scalable gateway sitting between applications, external services, and the enterprise, and acts as a container for Worklight application packages.

The Worklight Server is designed to integrate seamlessly with existing infrastructure, based on industry-standard adapters and server-side code. It is a scalable solution, enabling hundreds of thousands of users to perform transactional processes against the mobile app and back-end systems.

In summary, the Worklight Server provides capabilities including those in the following list:

- ▶ Encrypted communications between mobile devices and the enterprise
- ▶ Back-end connectivity to existing application services and servers
- ▶ Data manipulation, such as converting hierarchical data to JSON
- ▶ Single sign-on (SSO) authentication, with integration into existing enterprise authentication services
- ▶ Automatic collection of user-adoption and usage data for analysis

Server-side entities, such as configuration files and integration code, that are used in the Worklight Server, can be created and managed from within Worklight Studio. These artifacts are automatically built into web archive (WAR) files using Worklight Studio, and deployed onto the Worklight Server.

In providing the physical connectivity between the mobile device and enterprise systems, the Worklight Server supports a wide variety of adapter technology, such as SOAP, Representational State Transfer (REST), and Structured Query Language (SQL). It integrates multiple source data mashups into one stream for serving to the user, and can host server-side logic to deliver back-end data for mobile consumption.

2.1.3 Worklight device runtime components

Worklight provides client-side runtime code that services HTML5, hybrid, or native applications, and includes libraries for native and JavaScript implementations.

The runtime components enable a mobile app developer to accomplish the following tasks:

- ▶ Access back-end data and transactions through the invocation of Worklight services.
- ▶ Authenticate using pre-configured code to manage the authentication sequence, and to secure application data.
- ▶ Provide offline access with a local JavaScript Object Notation (JSON) database for data persistence, with back-end synchronization. Supports encryption and large data sets.
- ▶ Manage new application versions and disable applications in accordance with set policies.
- ▶ Troubleshoot code for detecting runtime connectivity problems in the application, and for collecting diagnostic information.
- ▶ Collect usage reporting for audit and analysis to be recorded by the Worklight Server.
- ▶ Ensure cross-platform compatibility (a set of uniform APIs to hide the differences in features and functions across different devices).
- ▶ Adjust features and functions during run time, optimizing the application for different devices across the same operating system (for example, a phone and a tablet).

2.1.4 Worklight Application Center

Worklight Application Center provides an enterprise application store for the distribution, governance, and management of prerelease and production-ready mobile apps.

The Worklight Application Center could be used, for example, to deploy applications to employees with their own bring your own device (BYOD) or company-owned mobile asset. Existing authentication frameworks can be used to manage application distribution by department, job function, geographical location, or other rule. Consequently, users accessing the store will see only the applications that they are entitled to download. Employees can rate apps and provide feedback.

For development teams, Worklight Application Center can be used to distribute prerelease software to developers and testers. Feedback can be quickly obtained and organized by version and device to isolate and resolve defects. It will integrate with the software build process to automate the distribution of the latest releases to project teams.

Key functions of the Worklight Application Center console include the following abilities:

- ▶ Upload different versions of mobile applications.
- ▶ Remove unwanted applications.
- ▶ Control access to applications.
- ▶ View feedback that mobile users have left for an application.
- ▶ Obtain information about applications installed on a device.
- ▶ Make an application inactive so that it cannot be downloaded.

From the mobile device, the Worklight Application Center enables users to accomplish the following tasks:

- ▶ List available mobile apps.
- ▶ Install a new app on a device.
- ▶ Send feedback about an app.

The Worklight Application Center supports applications for Android, iOS, and BlackBerry devices.

2.1.5 Worklight Console

The Worklight Console is a web-based user interface for the administration of the Worklight Server and deployed applications, adapters, and push-notifications. Analytics enable an administrator to search across logs and events that are collected from devices, applications, and servers for patterns, problems, and platform usage statistics.

By using the Worklight Console, an administrator can accomplish the following tasks:

- ▶ View dashboards that monitor all deployed adapters and applications.
- ▶ Control and monitor push notification services, event sources, and related apps.
- ▶ Assign device-specific security to support the installation of business apps on sanctioned devices.
- ▶ Manage multiple versions of the same app, and remotely disable apps by version and mobile operating system type.
- ▶ Define device-based access control policies to control the access to apps (Worklight v6.1 and later).

Operational analytics are collated in the Worklight Console. The operational analytics platform collects data about applications, adapters, devices, and logs to give a high-level view of the client interaction with the Worklight Server and assist with problem detection.

The operational analytics feature includes near real-time analytics for client activity with the Worklight Server, network latency analysis, client and server log searches and downloads, along with the ability to search both crash and stack traces. Sources for operational analytics data include:

- ▶ Crash events of an application on iOS and Android devices.
- ▶ Interactions of any nature between application and server.
- ▶ Server-side logs that are captured in traditional Worklight log files.

For more information, see 8.3, “Worklight operational analytics” on page 295.

2.2 Worklight in IBM MobileFirst

Worklight is the application and data platform component in the IBM MobileFirst suite of products, and participates in many aspects of the mobile app development lifecycle.

Many of the functions in Worklight are complimented and extended by other products in the IBM MobileFirst product set, such as advanced analytics with IBM Tealeaf® CX Mobile, and automated mobile testing with Rational Test Workbench.

In relation to the lifecycle, Worklight provides the following major capabilities (Figure 2-4 on page 40):

- ▶ Design and develop mobile enterprise-ready apps, rapidly and across platforms.
- ▶ Instrument apps to understand what is happening with the app in terms of security and usability.
- ▶ Integrate to back-end application services and data to reuse assets of value.

- ▶ Test with the IBM Mobile Test Workbench for Worklight (MTWW).
- ▶ Deploy mobile apps with the Worklight Application Center.
- ▶ Manage mobile apps through the Worklight Console and Worklight Application Center.

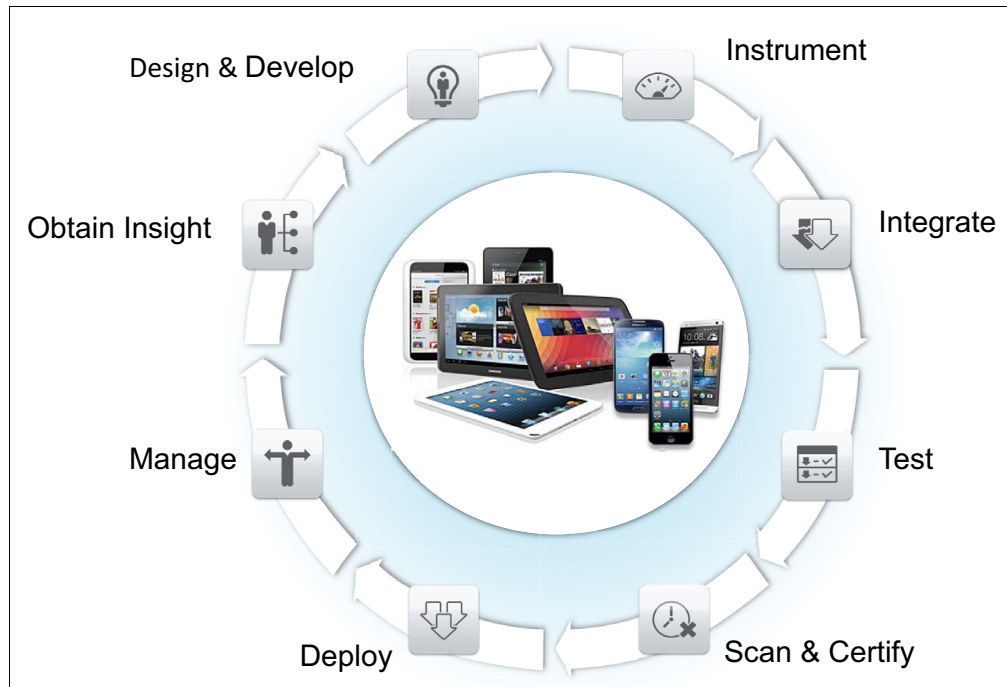


Figure 2-4 Mobile app development lifecycle

The mobile app development lifecycle could begin with a business problem or opportunity, but equally it can be initiated by a need to make changes to a deployed mobile app following insight from actual use.

You can use operational analysis to measure functional and non-functional attributes of the app, such as responsiveness and abandoned transactions, which directly feeds into a new iteration of the development lifecycle. This information can be used to make improvements, add functionality, or even remove functionality that is not adding benefit.

Worklight provides ready-for-use integration with Tealeaf CX Mobile, with the appropriate Tealeaf CX Mobile libraries for iOS, Android, and mobile web included. As soon as a Worklight project is created, a developer can immediately start instrumenting the app using Tealeaf CX Mobile technologies.

After the development of an app, the app can be deployed for testing. This deployment directly feeds into the testing phase of IBM MobileFirst, integrating with IBM Rational Test Workbench.

2.3 Business benefits

In the rapidly developing and changing world of mobile apps, businesses need to adapt and evolve constantly. User expectations of mobile apps are as high, if not higher, than their experience of other channels, such as web and in-store.

Worklight enables mobile apps to be designed and developed in a short space of time, and provides the deployment, management, governance, and security that results in a successful, enterprise-ready application.

Worklight advantages: Worklight decreases the cost of multi-platform application development, integration, and maintenance, and speeds up time to deployment.

Worklight brings the following key business benefits:

- ▶ Rapid time to value from developing to deploying a mobile app.
- ▶ Reduced development costs through a powerful platform API and features that enable enterprise-wide code sharing
- ▶ Meet customer's expectations the first time by simulating how the app will appear, and how it will behave, with application record and playback facility.

2.4 System topology

Figure 2-5 on page 42 shows a high-level view of the Worklight platform in an enterprise.

Through the Worklight platform, applications are distributed to the mobile devices through Worklight Application Center. Native, hybrid, and web-based mobile apps connect back to the mobile enterprise application platform (MEAP) to use the services that it provides. In this case, the MEAP is Worklight.

Application management is provided through Worklight Application Center to ensure that the applications are at the correct version, and accessible by the appropriate users.

In terms of security, Worklight facilitates user authentication and integration into existing security systems.

Worklight provides integration with back-end applications, systems, and services. Existing applications and database management systems can be accessed directly through common service-oriented architectures (SOAs) and connectors provided by the Worklight platform.

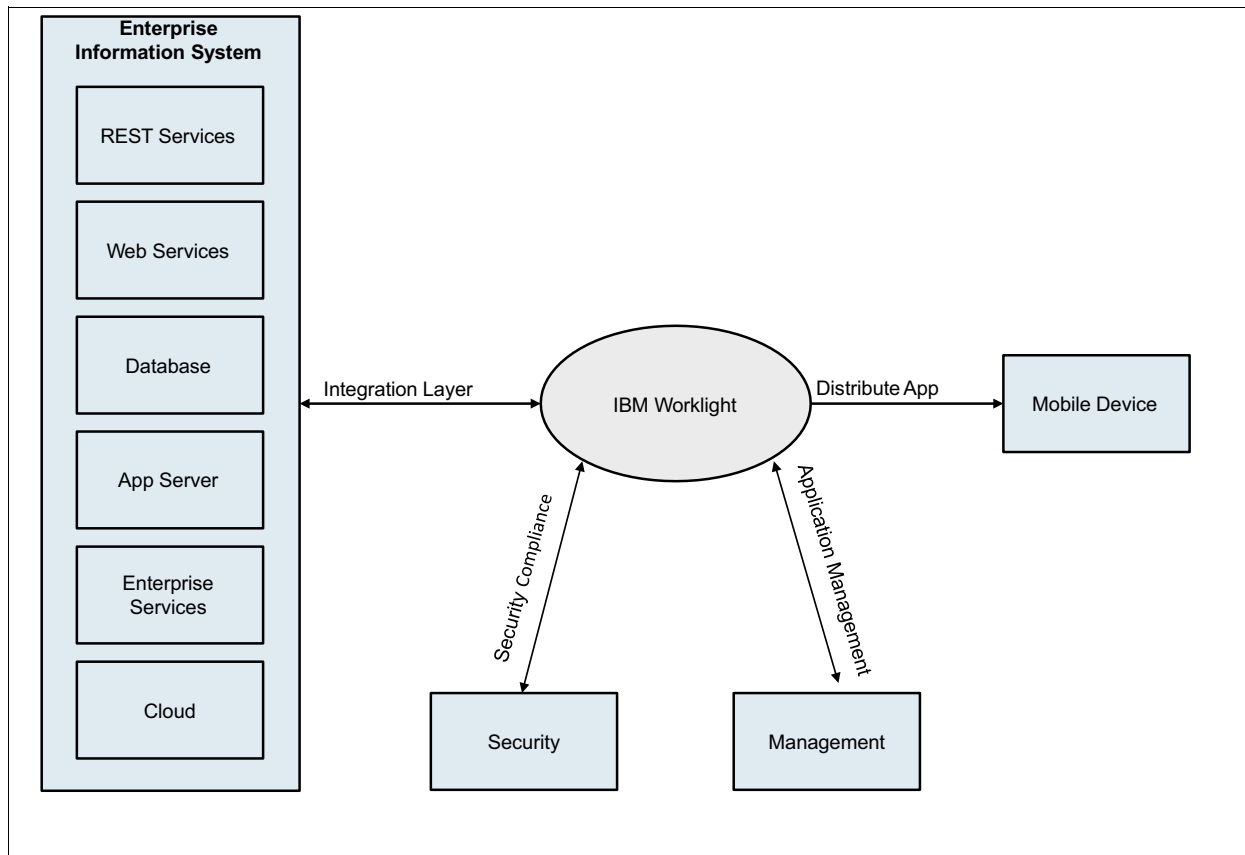


Figure 2-5 Worklight system overview

Figure 2-6 on page 43 shows a software layered view of a typical enterprise software infrastructure. The figure shows that the Worklight platform sits in the middleware layer to provide services between mobile devices and the enterprise.

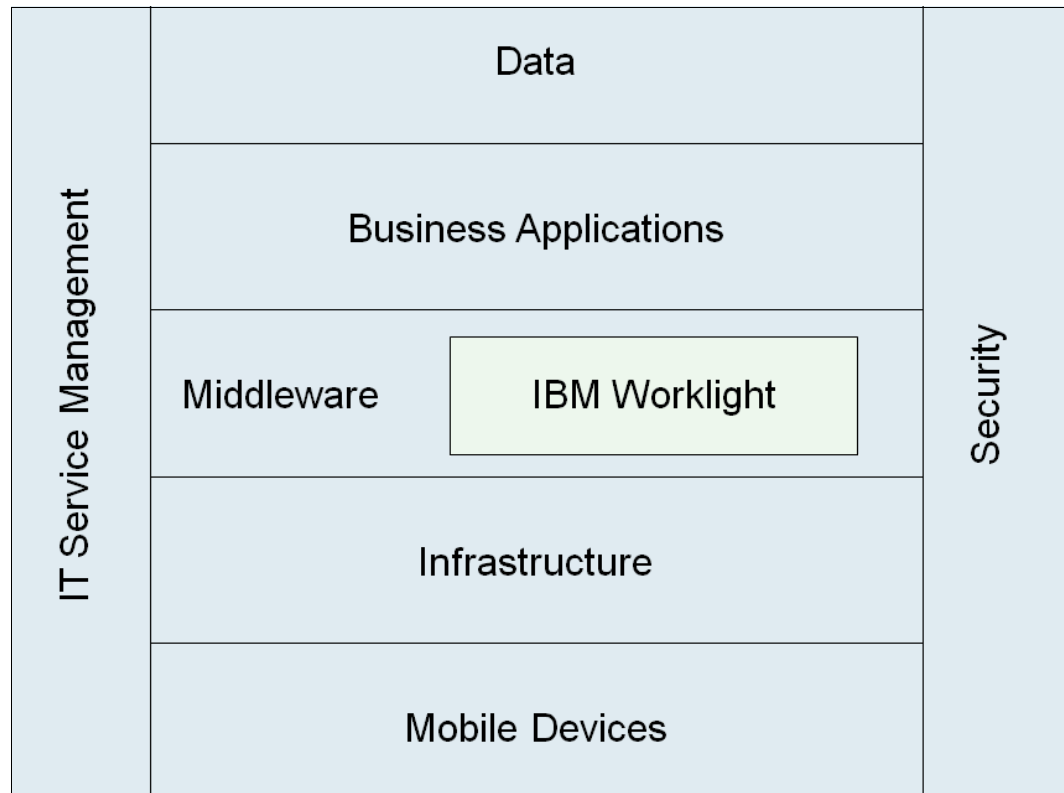


Figure 2-6 Enterprise software layered view and Worklight positioning

2.5 Components

This section provides more detail on the Worklight components described in 2.1, “Description of Worklight” on page 32. Each component provides a specific set of functions and supports a different stage in the lifecycle of a mobile application (Figure 2-2 on page 34).

- ▶ **Worklight Studio:** An Eclipse-based development environment with a rich set of tools for developing cross-platform applications.
- ▶ **Worklight Client-side Runtime:** A runtime environment with consistent application programming interfaces (APIs) that can work across different mobile platforms and integrate with existing services and security
- ▶ **Worklight Server:** A runtime server that enables secure access to enterprise services, application version management, unified push notifications, and direct application updates.
- ▶ **Worklight Console:** A web-based user interface for real-time analytics, push notification authoring, and mobile application version management.
- ▶ **Worklight Application Center:** A private, cross-platform mobile application store that is tailored for the specific needs of an application development team.

Worklight provides the development platform with which to create and modify mobile apps, but also provides the necessary APIs, services, and standards to interface with enterprise security, business applications, database management systems, infrastructure, and service management.

The development infrastructure supports the functions and capabilities of Worklight to provide the complete mobile app development and test solution, from building the app to deployment and run time.

2.5.1 Worklight Studio and command-line interface for IBM Worklight Developers

Worklight Studio is an Eclipse-based IDE to enable all of the coding and integration tasks required to develop a fully operational mobile app for various mobile operating systems. As described in 2.1.1, “Worklight Studio” on page 35, the tool supports open, multi-platform device development.

Additionally, Worklight provides command-line interface (CLI) tools as an alternative to the integrated development environment (IDE) of Worklight Studio.

The command-line interface for IBM Worklight Developers provides basic scaffolding, project, and app support for native and hybrid development environments. This provision enables developers to use their preferred text editors to create mobile applications outside of the traditional Worklight Studio environment.

The shell approach to mobile app development is a feature of the Worklight Studio platform that enables the application to be split into two portions (Figure 2-7 on page 45):

- ▶ An external shell
- ▶ An inner application

This approach enables development teams with different skill sets to develop independently, without affecting the overall appearance and functionality of the application.

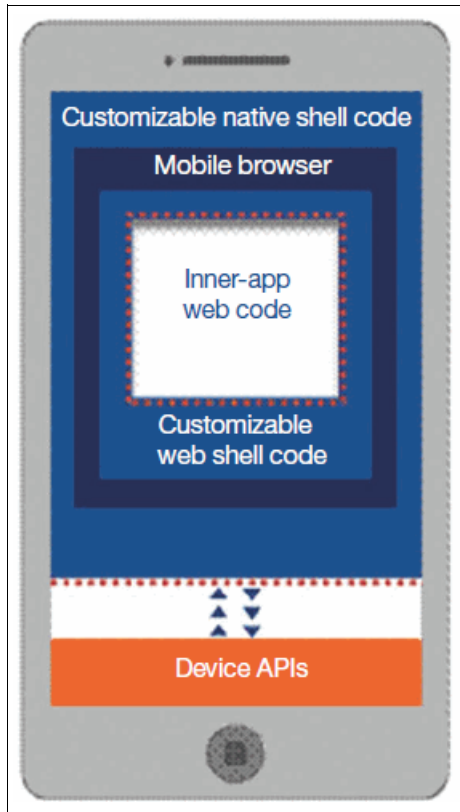


Figure 2-7 The shell approach to mobile development³

The shell enables developers to encapsulate mobile code and other assets into a reusable foundation. One development team can be responsible for developing the branding, security configurations, auditing, and authentication, and another can concentrate on the functional code. Any number of shells can be created, enforcing the inner applications to automatically comply with these standards.

With corporate policies enforced by the shell, application developers can focus on the core functionality of the application, such as the user interface, the business logic, and data integration. When completed, the shell and inner application development constructs are distributed into a single app.

2.5.2 Client-side runtime environment

The client runtime environment, as discussed in 2.1.3, “Worklight device runtime components” on page 38, consists of frameworks, libraries, APIs, and code bases to support the execution of apps on the mobile device.

This functionality enables the deployment of runtime code to the mobile device, with embedded server functionality.

Worklight provides a framework that enables the development, optimization, integration, and management of secure apps that run on client devices. The framework provides the following features:

- Automatic packaging and provisioning of application resources.

³ From IBM Worklight 6.0 Technology Overview
<http://public.dhe.ibm.com/common/ssi/ecm/en/wsw14181usen/WSW14181USEN.PDF>

- Tools that provide uniform access to back-end enterprise data, processes, and transactions.
- Uniform persistence.
- Flexible authentication and automatic application protection from web attacks.

Worklight uses the Apache Cordova development framework (Table 2-1) to deliver a bridge between standard web technologies (such as HTML5, CSS3, and JavaScript) and the native functions that the mobile device provides. Therefore, your team does not need to learn or adopt a proprietary programming language or model.

Table 2-1 Worklight development frameworks and programming models

Mobile OS	Devices	Native applications	Hybrid applications	Web applications
iOS4+	iPhone, iPad, iPod Touch	Objective C	Cordova	Yes
Android 2.3.3 - 4.4+	Phones and tablets	Java	Cordova	Yes
BlackBerry 6, 7, and 10	Phones	-	Web Works	Yes
Windows Phone 8.0+	Phones	C#	Cordova	Yes
Windows 8.0+ (including RT)	Desktops, notebooks, and tablets	-	Cordova	Yes
Feature Phones	Phones	Java ME	-	Yes (mobile browser must support HTML4, CSS 2.1, or JavaScript 1.5, or later versions)

Client-side runtime APIs are provided to improve application development, enabling the application to access various features during runtime. The libraries are bundled into the application, and integrate with the mobile app and the Worklight Server through predefined communication interfaces.

Client-side APIs: Client-side APIs include the following key features:

- ▶ Cross-platform compatibility to support development across all platforms, such as accessing common control elements of tab bars, clipboards, location services, and the camera.
- ▶ Client-to-server integration to ensure transparent communication between a mobile app and the Worklight Server. Worklight mobile apps always use an SSL-enabled connection to the server.
- ▶ An encrypted data store to access private data, using ISO/IEC 18033-3 standards, such as AES256 or PKCS#5.
- ▶ The ability to synchronize mobile app data with related data on the back-end, using JSON.
- ▶ Runtime skinning for abstracting devices from the design of the user interface. The runtime skin is applied during run time to configure to screen resolution, operating system, and form factor.

2.5.3 Worklight Server runtime environment

The Worklight Server, discussed in 2.1.2, “Worklight Server” on page 37, provides a run time for adapters, analytics, push notifications, and application hosting services.

Server-side APIs are provided to integrate communication between the mobile app and back-end systems. JavaScript and Java APIs can be called to perform functions, such as authentication, accessing web services, accessing a database, and subscribing to push notifications.

By developing server-side application code, the mobile app has direct access to back-end transactional and business process systems, such as IBM Business Process Manager, and cloud-based services. Performance, security, and maintenance are centralized.

Built-in JSON translation is provided to reduce the amount of data transferred between the mobile app and the Worklight Server. JSON is a lightweight and human-readable data interchange format, and because it is smaller than other data interchange protocols, such as XML, it can be quickly generated and parsed. The Worklight Server can automatically convert hierarchical data to the JSON format, further optimizing delivery and consumption by the mobile app.

A built-in security framework provides connectivity and integration into existing enterprise security systems, and sends connection credentials to the back-end. This approach enables the mobile app to use existing security systems, with credentials residing on the server, not the device.

The adapter library can be used to connect to various back-end systems, with adapters provided for SOAP, XML over HTTP, Java Database Connectivity (JDBC), and Java Message Service (JMS), as shown in Figure 2-8. With this capability, complex lookup procedures can be defined, and data combined from multiple back-end services. This aggregation reduces the amount of network traffic to and from the mobile device.

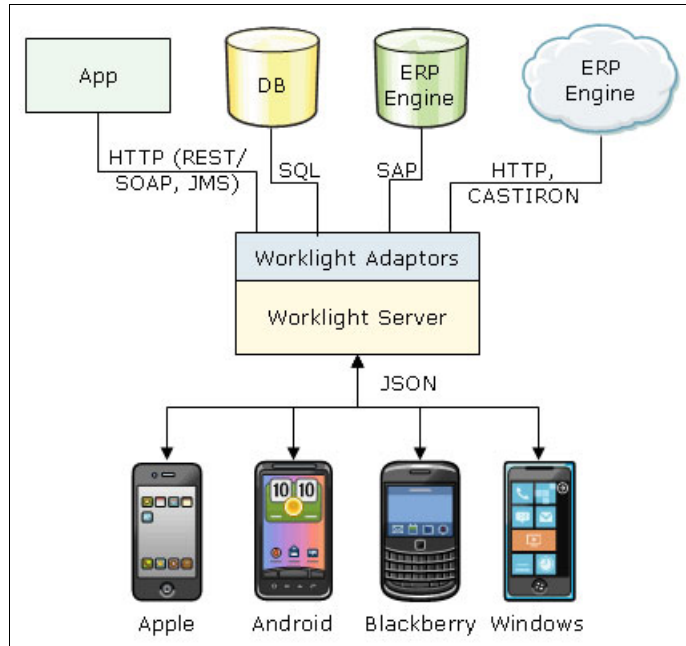


Figure 2-8 Worklight adapters for back-end connectivity.

The push notification service in the server runtime environment (Figure 2-9 on page 49) is an abstraction layer for sending notifications to the mobile device, using either the device vendor's infrastructure or the Worklight Server Software Management Services (SMS) capabilities.

The request to subscribe to notifications is sent from the user's application to the Worklight Server, containing information about the device and platform. The system administrator can manage subscriptions, notifications from back-end systems, and use the Worklight Application Center to send notifications to the devices.

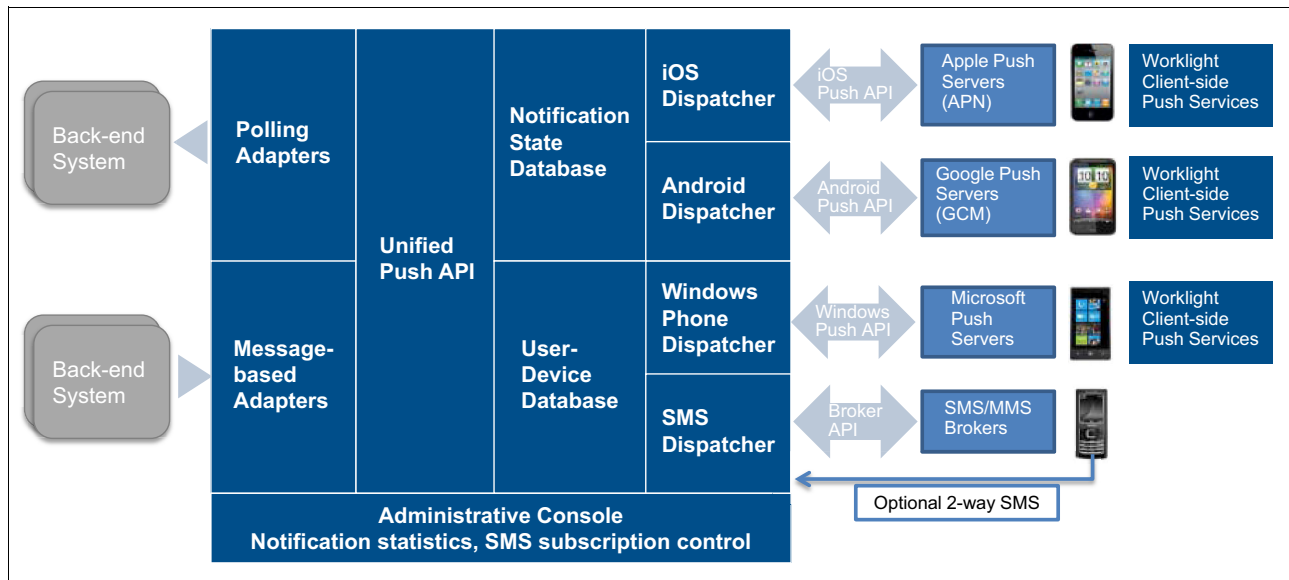


Figure 2-9 Worklight push notification service

The operational analytics feature enables searches across applications, services, devices, and other sources, to reveal usage data or detect problems. In addition to summary report data, more detailed operational analytics are available in the Worklight Console, described in 2.1.5, “Worklight Console” on page 39, and in IBM Tealeaf CX Mobile.

Worklight includes the IBM WebSphere Analytics Platform (IWAP), which drives the Worklight analytics feature. IWAP is designed to help understand the volume, velocity, veracity (amount of noise in data), and variety of mobile data.

Figure 2-10 shows how Worklight can interface with IWAP, and together with Tealeaf CX Mobile, can create detailed reports on mobile data to be accessed by Business Intelligence and Reporting Tools (BIRT).

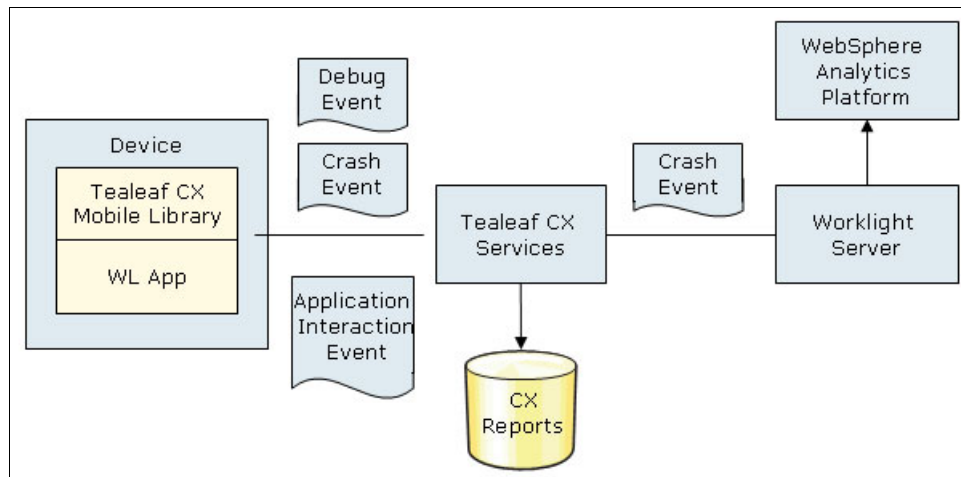


Figure 2-10 Worklight Operational Reporting interfacing with Tealeaf CX Mobile

2.5.4 Worklight Console

The Worklight Console is a centralized web-based portal, used for the control and management of the deployed application, and supports the analysis of user statistics collected by the Worklight Server runtime environment.

Details of the functionality and capability of the console are described in 2.1.5, “Worklight Console” on page 39.

The console consists of application catalogs, a dashboard, and a push notification display. Figure 2-11 shows an example of the push notifications panel, with the left column displaying the list of data sources configured in the Worklight Server and how many users are subscribed to it. The column on the right displays the deployed applications that can receive push notifications, how many notifications have been sent since system start, and any errors that have occurred.

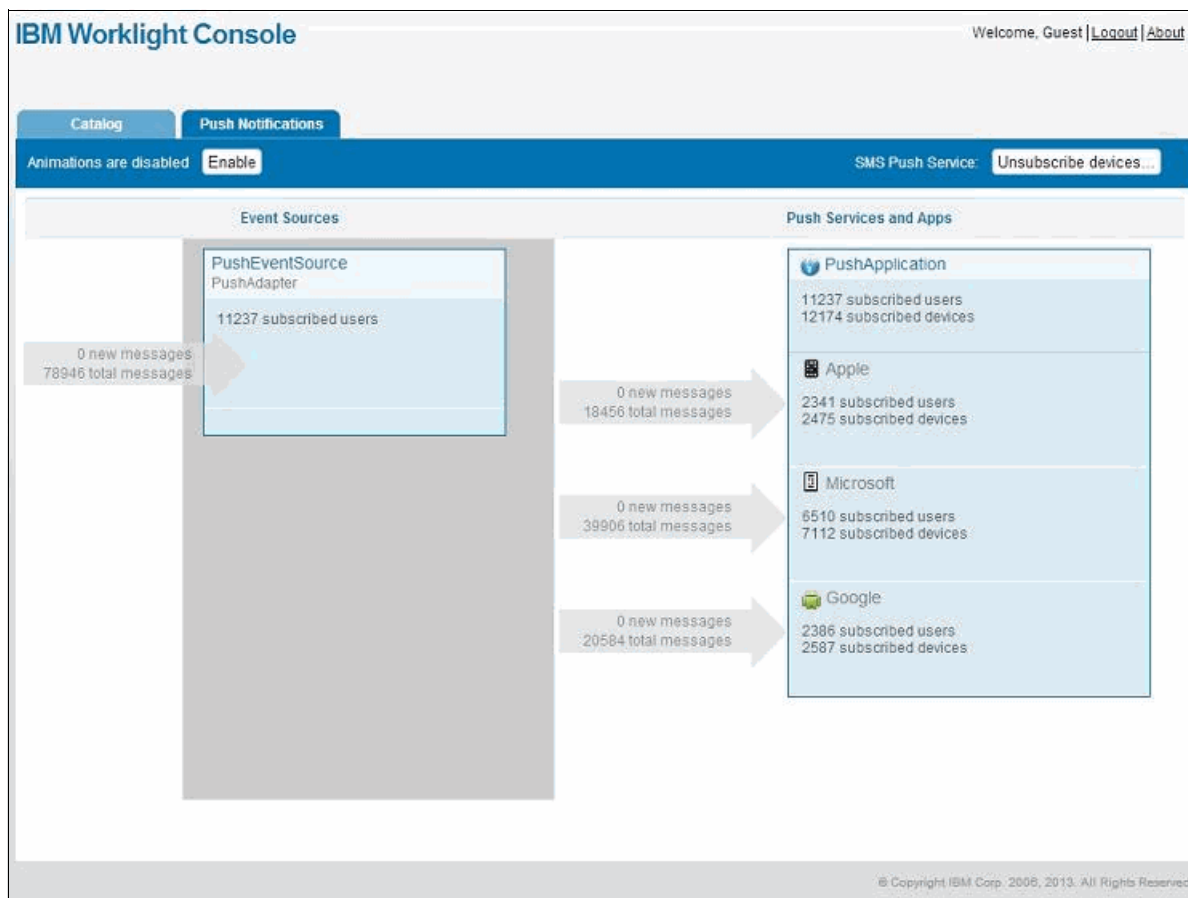


Figure 2-11 Worklight Console push notification

2.5.5 Worklight Application Center

The Worklight Application Center is composed of four main components:

- ▶ The mobile client application is used to install applications on a mobile device, and to send feedback about the application to the server.
- ▶ The web console is a Java based application, hosted server-side, that must be deployed in a web application server, such as IBM WebSphere Application Server.

- ▶ Application center services provide functions, such as listing available applications, delivering binary files to the mobile device, and registering feedback and ratings.
- ▶ The server-side component consists of an administration console and the mobile apps.

Details are described in 2.1.4, “Worklight Application Center” on page 38.

A database repository is held on the server. It contains information regarding which apps are installed on mobile devices, the feedback about apps, and the application binary files.

A web administration console is provided to enable users to manage apps, user access rights to install apps, feedback, and details about the apps installed on devices.

Figure 2-12 shows how the Worklight Application Center can be used to view all of the apps installed on a device.

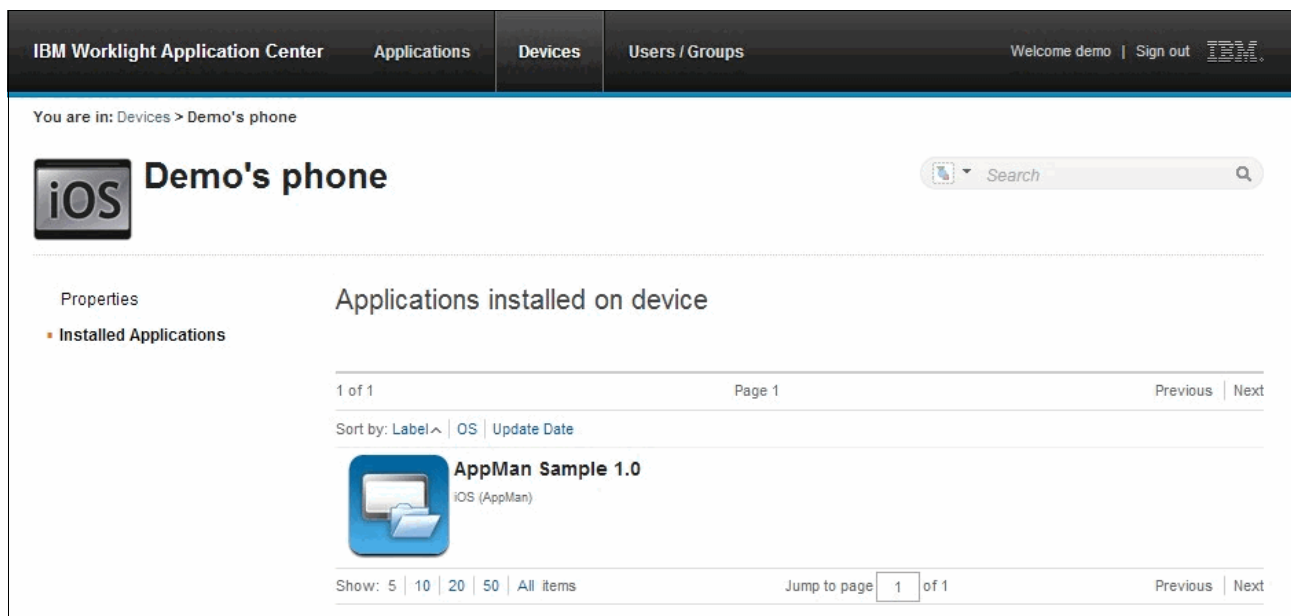


Figure 2-12 Worklight Application Center showing installed applications

Figure 2-13 on page 52 shows the architecture of the Worklight Application Center, and how the Application Services can use services.

Mobile devices can view the apps available to install through the catalog service or the install service, and they can provide feedback through the feedback services. The Worklight Application Center mobile app is installed locally on the mobile device, and an administrator uses the Worklight Application Center Console to manage the entire process.

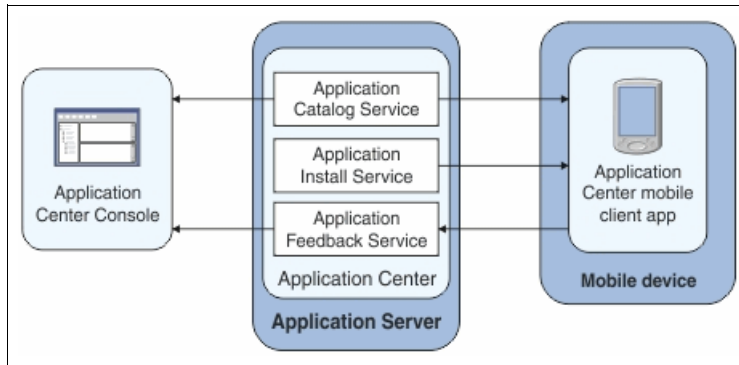


Figure 2-13 Worklight Application Center architecture

Applications that are developed in the Worklight Studio can be published directly into the Worklight Application Center.



IBM Business Process Manager V8.5.5 overview

This chapter provides a general overview of the IBM Business Process Manager (IBM BPM) product and includes the following topics:

- ▶ Understanding business process management
- ▶ Lifecycle of a business process
- ▶ IBM Business Process Manager V8.5.5 new features

3.1 Understanding business process management

The business process management discipline enables organizations to be more efficient, effective, and capable of change while managing time and costs. In this discipline, processes are viewed as strategic assets of an organization that must be understood, managed, and improved to deliver added value products and services to process consumers.

In a typical organization, there are IT systems in place to solve specific functional areas. However, at the human level, the work is not well-controlled and there is poor end-to-end visibility into the entire process that spans humans and systems, as shown in Figure 3-1.

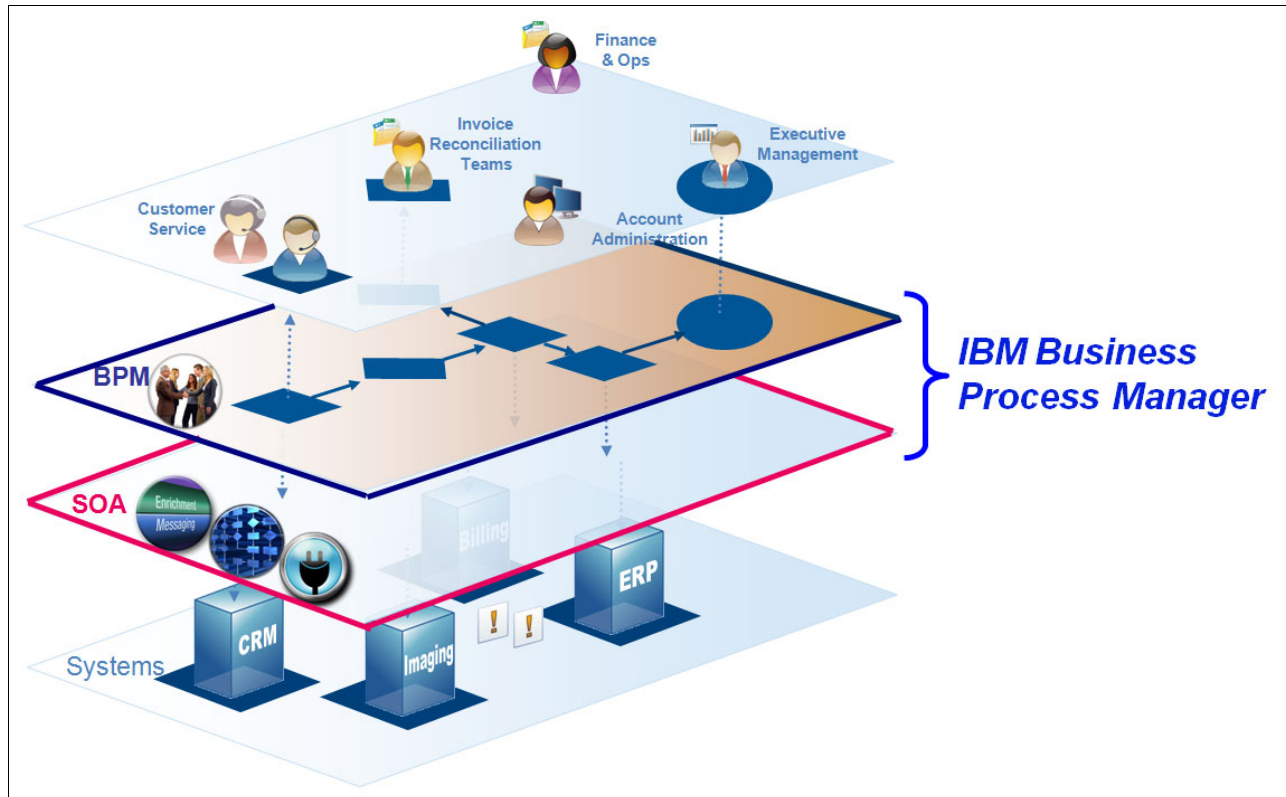


Figure 3-1 How IBM BPM fits into the IT ecosystem

IBM BPM is a platform that helps process participants gain organization-wide visibility of your business processes so that improvements can be made quickly by those who know your processes best. This platform encourages collaboration and supports strong governance of process change.

IBM BPM provides a common software platform for process improvement and business process management lifecycle governance, which offers the power and robustness that is required for mission-critical enterprise solutions while combining the simplicity that is required for business engagement through collaboration. Built-in analytics and search capabilities help you improve and optimize your business processes now and in the future. IBM BPM includes tooling and run time for process design, execution, monitoring, and optimization.

3.2 Lifecycle of a business process

If a process is used, there are always actions that are related to it. Process improvement activity is a continuous, ongoing activity even after a process is deployed and in use.

The lifecycle of a business process consists of the following stages:

- Discover and define

This is the first stage of the process lifecycle where requirements are identified and analyzed. Key process milestones are identified and process metrics to be captured are defined.

- Design and implement

In this phase, the actual development of the business process occurs including any external system integrations.

- Execute and monitor

In this phase, the newly developed business process is distributed to the environment where it is run and used by a business organization and monitored by administrators.

- Measure and optimize

This phase concentrates on collecting process metrics and process improvement activities that are related to the business process, such as optimizing and initiating updates to the process.

Figure 3-2 shows these phases of a business process lifecycle.

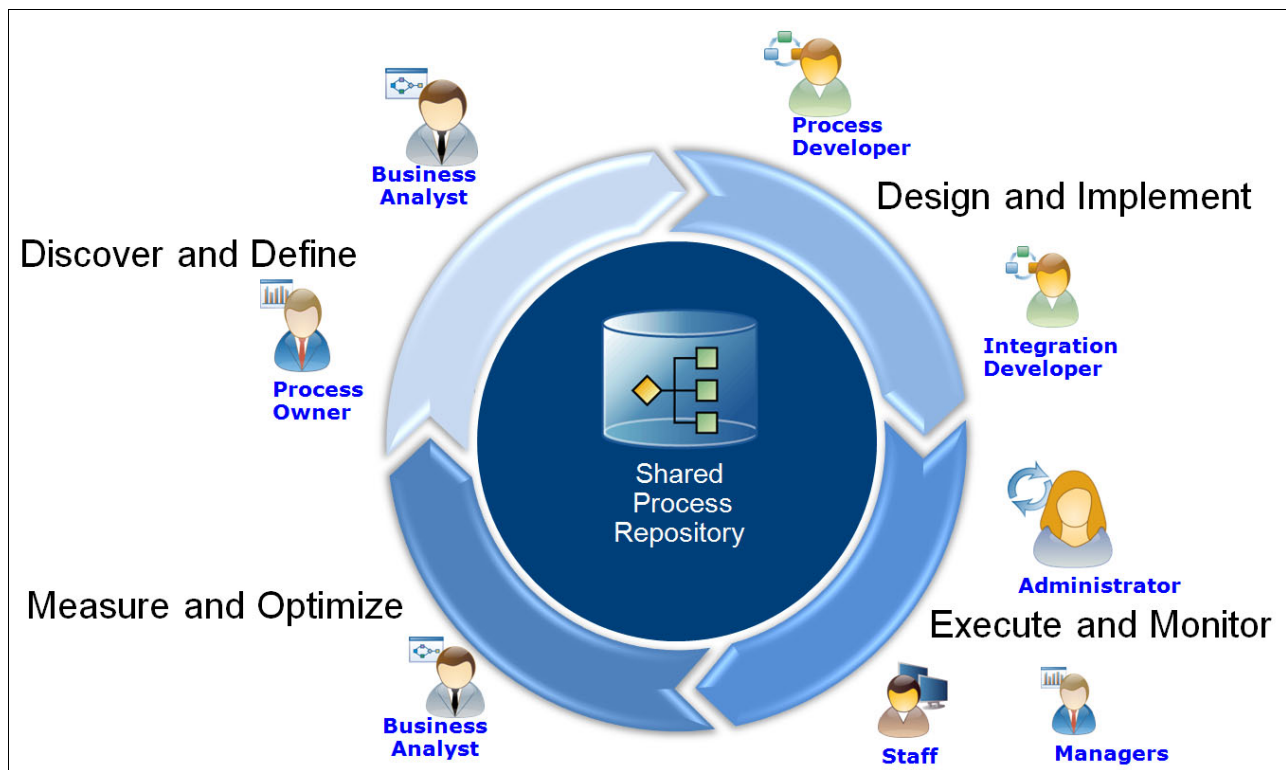


Figure 3-2 Lifecycle of a business process

3.3 IBM Business Process Manager

IBM Business Process Manager brings together a number of technologies and capabilities under a single unified platform with which customers can build human-centric and system-centric business processes. It enables organizations to implement businesses process lifecycle and includes tooling and run time for process design, execution, monitoring, and optimization.

A business process management environment mainly consists of a repository environment to store and develop IBM BPM artifacts, authoring tools for process developers to develop and test processes, one or more runtime environments to deploy processes, and a set of administration and monitoring tools to administer and monitor business processes.

Figure 3-3 is a high-level overview of a basic IBM BPM environment.

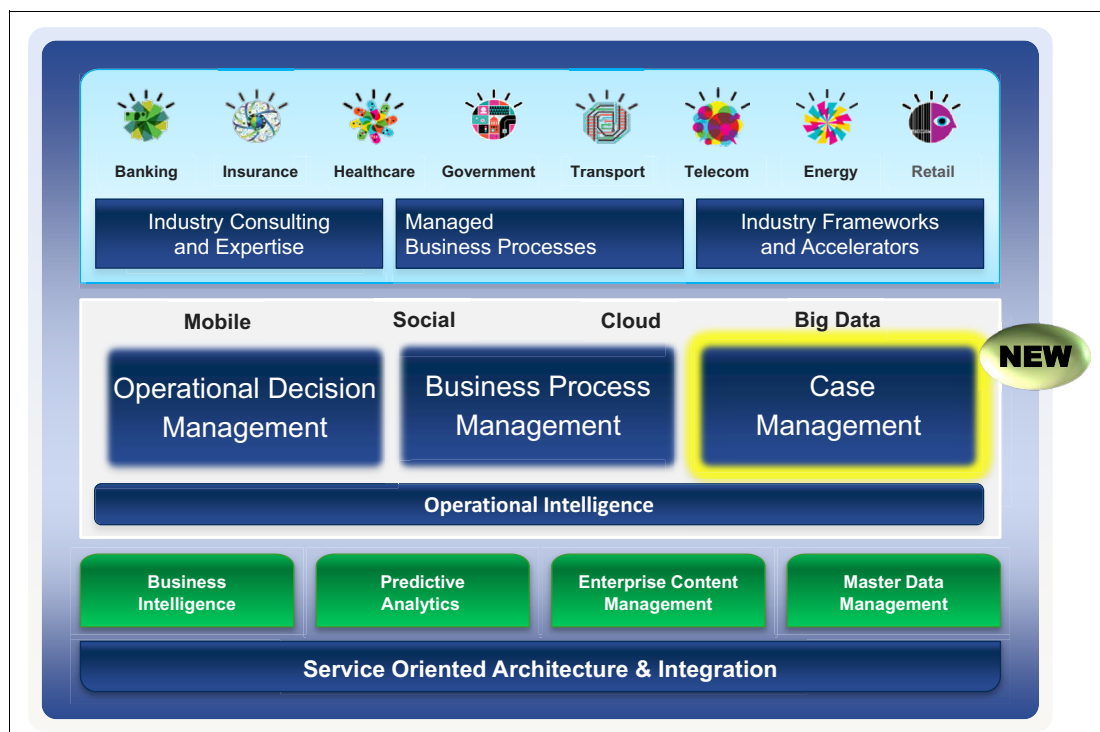


Figure 3-3 IBM Smarter Process v8.5.5 Capabilities

IBM Smarter Process platform includes three industry-leading and complementary services and capabilities:

- ▶ IBM Business Process Manager
- ▶ IBM Case Manager
- ▶ IBM Operational Decision Manager

IBM Business Process Manager is designed for process-centric workloads, IBM Case Manager is optimized for content-centric workloads, and IBM Operational Decision Manager automates operational decisions with business rules and events.

Organizations may use these products individually or in combination to solve business challenges. Organizations with all three offerings are able to leverage the unique benefits of each to satisfy the full spectrum of IBM Smarter Process work patterns.

3.3.1 Product overview

Figure 3-4 provides an overview of the main components of the IBM BPM product.

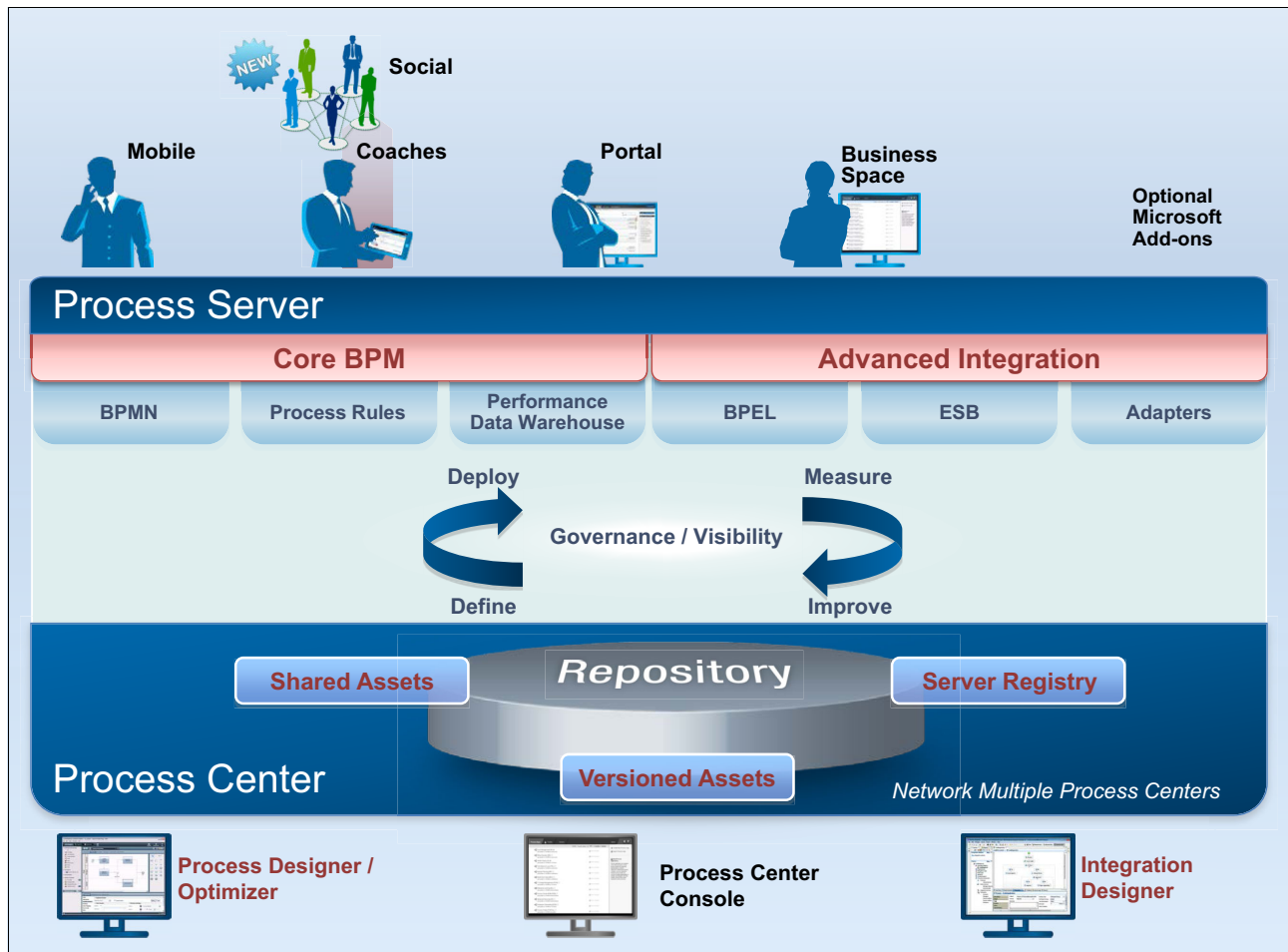


Figure 3-4 IBM BPM main components

The following sections describe the components.

Process Center

The Process Center includes a repository for all process artifacts and provides the tooling that is required to develop, deploy, and manage process applications.

The Process Center has a console web application with which users can maintain the repository, deploy process applications to Process Server environments, and manage running instances of process applications in configured environments. The Process Center also includes a Process Server (the playback server) and Performance Data Warehouse, with which users who are working in the authoring environments can run processes and store performance data for testing and playback purposes.

Authoring environments

IBM Business Process Manager uses the following authoring environments:

- IBM Process Designer

IBM Process Designer is a design time tool that is used to develop process applications. It is an Eclipse-based tool that is used by business process authors. IBM Process Designer

is available in all configurations of the product, and offers the capability to model and implement business processes as process applications. IBM Process Designer includes other tooling, that is, the Process Inspector, and the Process Optimizer, for interacting with processes that are running on the Process Center Server (playback server) or a connected Process Server deployment target.

► **IBM Integration Designer**

IBM Integration Designer is a development environment for building end-to-end SOA applications. It includes numerous pre-packaged integration adapters to build complex automated processes.

IBM Integration Designer is available only in the advanced configurations of the product. IBM Integration Designer is used to author complex integrations and fully automated processes that support process applications that are designed in the IBM Process Designer. By using IBM Integration Designer, with a fully integrated testing environment that uses test cases and test suites, IT developers can build reusable SOA services, orchestrate those services, and access back-end systems.

Process Server

Process Server is an IBM BPM runtime environment that supports running a range of business processes and integrations. By using Process Server, you can run processes as you build them. Process Server runs processes and services that authors build by using the Authoring Environments. Process Server handles the access of a process to external applications, the coaches (the user interfaces for process participants), and the business logic. Every process component is loaded and runs in Process Server at run time.

An organization often has multiple Process Server environments, such as test, user acceptance test, and production. Each Process Server registers with the Process Center and allows the Process Center to facilitate code deployment and management of the runtime process server environments.

The Process Server also provides a number of tools, for example, the Process Admin Console, which enables the management of the Process Server and deployed applications. Another tool is the Process Portal, which can be used to give individual users access to run and manage the tasks of the business processes. In addition, the Process Portal measures performance of business processes, teams, and individuals. The Process Portal delivers an accurate overview that covers all aspects of the governed running business processes. This server environment also contains the Performance Data Warehouse component that collects and aggregates process data from processes that are running on the process servers. This data can be used to improve business processes.

3.3.2 Product editions

IBM Business Process Manager is a unified platform that combines human-centric and integration-centric capabilities into one product. Different configurations of the product are available for different users, and satisfy different needs in the organization. Product configurations can be combined for collaborative authoring and network-deployed runtime environments.

This product is available in three different editions. Table 3-1 on page 59 provides a high-level overview of each edition.

Table 3-1 List of IBM BPM product editions

Advanced Edition	Complete set of business process management capabilities: <ul style="list-style-type: none"> ▶ Extended support for high-volume process automation. ▶ Built-in SOA components for extensive organization-wide service integration and orchestration.
Standard Edition	Configured for typical business process management projects: <ul style="list-style-type: none"> ▶ For multi-project improvement programs, with high business involvement. ▶ Basic system integration support. ▶ Rapid time-to-value and improved user productivity.
Express Edition	Configured for first business process management project: <ul style="list-style-type: none"> ▶ Rapid time-to-value; improved user productivity. ▶ Low entry price. ▶ Easy installation and configuration.

Table 3-2 shows the capabilities and features that are available in each product edition.

Table 3-2 IBM Business Process Manager capabilities by edition

Capability	Advanced	Standard	Express
WebSphere Lombardi Edition compatible execution	X	X	X
Process Designer (BPMN)	X	X	X
Collaborative editing and immediate playback	X	X	X
Interactive process coach user interfaces	X	X	X
ODM-based process rules	X	X	X
Process Portal	X	X	X
Real-time monitoring and reporting	X	X	X
Performance analytics and optimizer	X	X	X
Performance Data Warehouse	X	X	X
Process Center/shared asset repository	X	X	X
Unlimited process authors and users	X	X	200 users/ three authors
High availability: Clustering and unlimited cores	X	X	<ul style="list-style-type: none"> ▶ Four cores production ▶ Two cores development ▶ No cluster
WebSphere Process Server compatible execution	X		

Capability	Advanced	Standard	Express
Integration Designer (BPEL/SOA)	X		
Transaction support	X		
Integration adapters	X		
Flexible Business Space user interface	X	X	X

For more information about IBM BPM, refer to the IBM BPM V8.5.5 documentation in the IBM Knowledge Center at:

http://www.ibm.com/support/knowledgecenter/SSFPJS_8.5.5/com.ibm.wbpm.main.doc/kc-homepage-bpm.html

Also, see the IBM Redbooks publication *Business Process Management Deployment Guide Using IBM Business Process Manager V8.5*, SG24-8175.

3.4 What's new in IBM BPM V8.5.5

IBM BPM V8.5.5 is an update to the IBM comprehensive and consumable BPM platform that provides visibility and management of your business processes. It includes tooling and runtime for process design and execution, along with capabilities for monitoring and optimizing work executed within the platform. It is specifically designed to enable process owners and business users to engage directly in the improvement of their business processes. This collaboration between business and IT enables superior time to value and improved user productivity as compared to traditional methods of development.

A highly integrated environment that scales smoothly and easily from initial project to enterprise-wide program, IBM BPM V8.5.5 is designed for easy deployment and is ready to use immediately or as an easily customizable configuration.

Figure 3-5 on page 61 provides an overview of the new features in IBM BPM V8.5.5. For information about the IBM BPM V8.5.5 enhancements, refer to the IBM United States Software Announcement 214-141 at the following link:

http://www.ibm.com/common/ssi/rep_ca/1/897/ENUS214-141/ENUS214-141.PDF

Dynamic Processes & Basic Case Management

- Unstructured ad-hoc work
- Basic case folder & document handling
- Customizable Instance UIs
- Web-based Case Editor



Mobile Ready End-User Experience

- WYSIWYG Coach Designer
- Responsive Coach Views
- Client-side Human Services



BPMN2 Enhancements

- (*) SCA-based event triggering with correlation
- Latency optimizations



Various Improvements

- Business Monitor update
- Migration Tooling enhancements
- Configuration Management improvements



Figure 3-5 IBM BPM V8.5.5 new features

All editions of IBM BPM V8.5.5 include new design capabilities for creating responsive user interfaces that can be designed once and run on any device form factor (smartphone, tablet, or desktop), to support mobile-ready process applications. Mobile-optimized UIs can now be created with new *what you see is what you get* (WYSIWYG) tooling that enables the design of responsive process coaches. These coach UIs are targeted for multiple mobile device form factors: small (phone), medium (tablet), and large (desktop). User interactions on mobile devices can also be optimized by taking advantage of new client-side services that are downloaded and cached on the device.

New features built into IBM BPM V8.5.5 Advanced include:

- ▶ Basic case management capabilities that are a built-in feature of IBM BPM Advanced. IBM BPM Advanced V8.5.5 embeds basic case management capabilities that are derived from IBM Case Manager technology. These capabilities allow you to exploit your existing IBM BPM investments and skills, and simplify the implementation and management of process-oriented solutions that require basic case handling. IBM BPM Advanced continues to support other process-oriented work patterns, including straight-through processing and user-oriented workflow. The basic case management feature is enabled by ordering a separate license.
- ▶ A new Case Designer that extends the unified design experience that is built upon the centralized IBM Process Center repository to simplify the development and deployment of process and basic case applications.
- ▶ Embedded content repository that is restricted to support the basic case management documents and folders. The restricted use content repository can be extended to support unlimited content management use cases with IBM Enterprise Content Management.
- ▶ Expanded Process Portal to allow knowledge workers to easily interact with cases, folders, and documents.

The programs packaged with IBM BPM V8.5.5 are:

- ▶ **WebSphere Application Server Network Deployment V8.5.5**

IBM BPM V8.5.5 runs on WebSphere Application Server Network Deployment V8.5.5, which enables near-continuous availability, advanced management, and automated performance optimization. IBM BPM inherits these capabilities and delivers a high-availability and highly scalable environment for your process applications.

- ▶ **IBM DB2 Workgroup Server Edition**

DB2 Workgroup Server Edition offers increased data-protection capabilities, scalability, and performance for all database-intensive operations, which, based on DB2 technology, are designed to manage data more effectively and efficiently. Greater availability is delivered through enhancements such as online automated database reorganization. In addition, the increased scalability and the ability to take advantage of the latest in server technology helps deliver increased performance of backup and recovery processes.

Note: DB2 Express Edition is included as a default database server in all IBM BPM V8.5.5 editions. DB2 Workgroup Server Edition is also packaged with IBM BPM Standard V8.5.5 and IBM BPM Advanced V8.5.5 for optional use.

- ▶ **IBM Worklight Enterprise Edition for Non-production Environment**

IBM Worklight enables you to develop rich, cross-platform applications that can access the full capabilities of a wide range of mobile devices. Worklight can help reduce time to market, cost, and complexity of development, while enabling an optimized customer and employee-user experience across multiple environments.

Note: Worklight Enterprise Edition for Non-production Environment is packaged with IBM BPM Standard V8.5.5 and IBM BPM Advanced V8.5.5 for optional use.

3.4.1 IBM Business Monitor V8.5.5

IBM Business Monitor is a separate product from IBM BPM, but it is closely linked. When clients must collect key performance indicators (KPIs) about their business processes, they can either use the Performance Data Warehouse (which is included with IBM BPM) or they can leverage IBM BPM Monitor. Clients sometimes require the extra features that IBM BPM Monitor provides.

IBM Business Monitor V8.5.5 is an update to the IBM BPM comprehensive, operational, intelligence solution that offers visibility into real-time, end-to-end business operations, transactions, and processes to help optimize processes and increase efficiency.

Business Monitor V8.5.5 includes:

- ▶ Simplified, event infrastructure that is optimized for performance.
- ▶ Support for the IBM BPM V8.5.5 operating environment.
- ▶ Powerful, new visualizations of data that are based on IBM Cognos® BI V10.2.1.

Business Monitor V8.5.5 delivers comprehensive, operational intelligence capabilities that offer business users and managers visibility into real-time, end-to-end business operations, transactions, and processes to help increase revenue, lower costs, and improve service. The monitoring capabilities of Business Monitor, which complement those of IBM BPM, enhance the ability to have visibility into key business transactions that execute in multiple systems across the enterprise.

Business Monitor provides customizable dashboards that calculate and display KPIs and metrics that are derived from business transactions, processes, business activity data, and business events. Business users can view dashboards through lightweight web interfaces, mobile devices, and corporate portals, and they can drill into specific transaction and process instances.

Business Monitor V8.5.5 is an update to Version 8.0.1 and includes the following capabilities:

- ▶ Enable operational environment currency. Business Monitor V8.5.5 utilizes more current versions of supporting products. It includes:
 - Support for DB2 Enterprise Server Edition V10.5, DB2 Workgroup Server Edition V10.5, and DB2 Express Edition V10.5.
 - Support for IBM Cognos Business Intelligence V10.2.1.
 - Support for IBM WebSphere Application Server V8.5.5.
- ▶ Simplified event infrastructure that is optimized for performance.

Business Monitor V8.5.5 offers improved event handling performance with the Dynamic Event Framework (DEF), an alternative to the Common Event Infrastructure (CEI). While the CEI framework remains available and supported, the DEF handles events more efficiently resulting in better performance.

- ▶ Access powerful new visualizations of data.

Based on Cognos Business Intelligence V10.2.1, IBM Business Monitor users have access to new and innovative chart types as well as more interactive versions of existing charts. IBM Rapidly Adaptive Visualization Engine (RAVE) visualizations enhance the Cognos Active Report consumption experience across consumption environments that include mobile devices. Report authors benefit from an improved authoring experience when RAVE visualizations are used. Enhanced flexibility provides the opportunity to customize charts to meet business-specific needs.

- ▶ Deliver faster time to value with simplified installation and configuration.

Business Monitor V8.5.5 provides functions that help improve and simplify installation.

In addition, Business Monitor V8.5.5 (and earlier versions) helps you to:

- ▶ Discover and monitor SAP processes. Mine SAP business events to discover actual processes and act in real time to business challenges:
 - View SAP transaction sequences as processes without process orchestration.
 - Simply turn on the SAP Business Events as needed and it is nonintrusive to SAP.
 - Understand and document the actual SAP processes that are being used to prepare for process innovation, detect business challenges as they happen, and address issues before they become problems.

For more information about extending SAP capabilities with IBM BPM, refer to the IBM Redbooks publication *IBM Software for SAP Solutions*, SG24-8230.

- ▶ Enable insight where and when you need it. Business Monitor V8.5.5 supports the new Cognos Mobile application for iPad to view Business Monitor reports from iPad mobile devices.



Architecture patterns for mobile-enabled processes

This chapter discusses architectural patterns for exposing business processes to mobile environments. It includes an overview of IBM MobileFirst reference architecture and deployment considerations and comprises the following key topics:

- ▶ Mobile architecture and deployment topology
- ▶ IBM Business Process Manager interaction patterns

4.1 IBM MobileFirst reference architecture

IBM MobileFirst reference architecture is a collection of architectural artifacts that describe how to deliver enterprise mobile solutions using the IBM MobileFirst software portfolio. The artifacts include definitions for platform and application development, mobile security, mobile device management, and mobile analytics. The reference architecture defines a set of capabilities that should be considered when developing an enterprise mobile solution (see Figure 4-1). The focus of this chapter is on the architectural aspects related to the direct runtime path from mobile apps through to the IBM BPM engine and integration to enterprise applications along with topology considerations. From the reference architecture perspective, these aspects include:

- ▶ Security
 - Access gateway
 - Security connectivity
- ▶ Applications and Data Platform
 - Mobile specific middleware
 - Service composition
- ▶ Cloud and managed services



Figure 4-1 IBM MobileFirst reference architecture capabilities

For staff deployments, the management capability becomes an important consideration when deploying to hundreds if not thousands of mobile devices within an enterprise. The

provisioning of mobile applications, version management, usage monitoring, and enterprise security policy enforcement are key to supporting larger scale rollouts.

Security is also a key dimension that requires particular focus as potentially sensitive corporate data suddenly finds itself on many portable devices that may be easily misplaced or stolen.

4.2 Enterprise mobile architecture

An enterprise mobile architecture can be represented at a high level as a stack of layers that follow the pathway of interaction between the mobile application and enterprise systems (see Figure 4-2). Catering for both customer-facing mobile applications and staff-facing mobile applications, the architecture covers a range of device types including smartphones, tablets, and Internet of Things (IoT)-based devices (embedded devices and smart appliances).

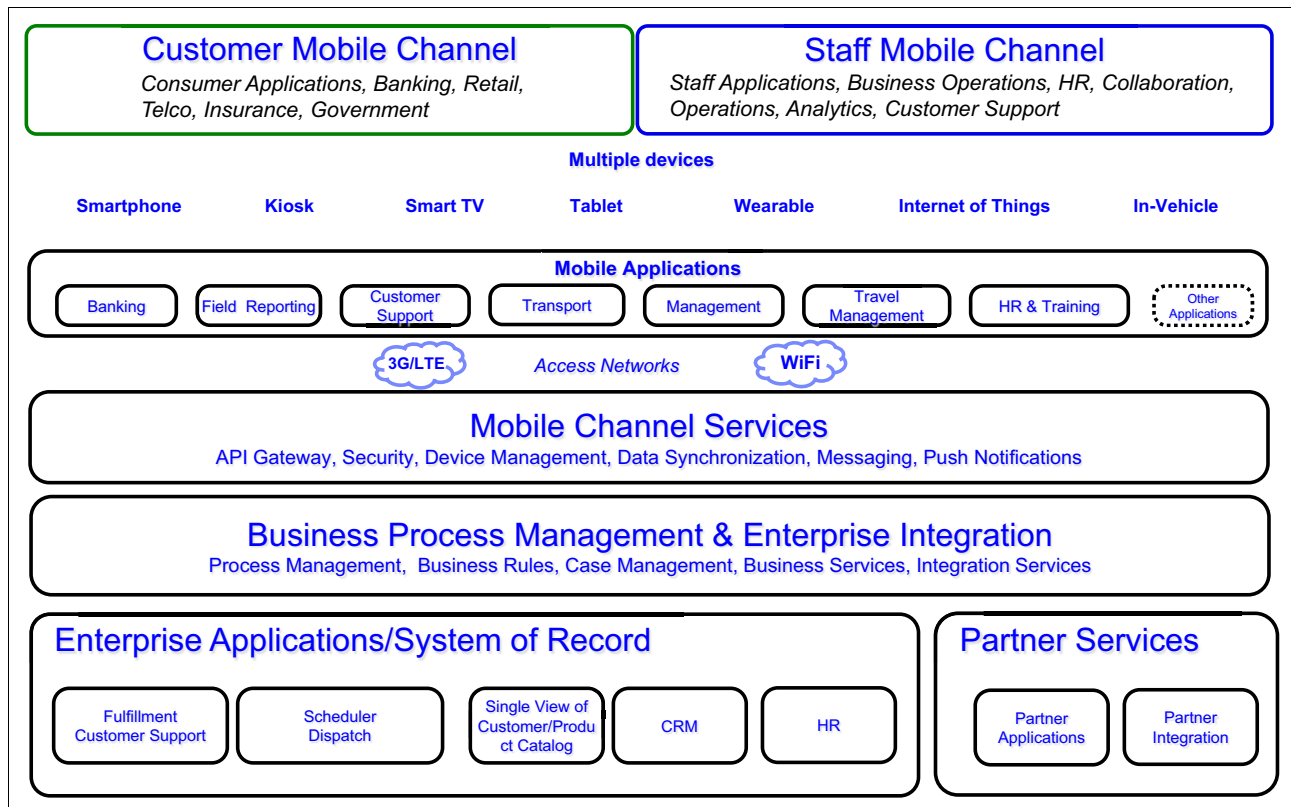


Figure 4-2 Enterprise mobile architecture

4.2.1 Mobile devices and applications

The type of mobile devices that may be used by an enterprise can vary depending on the nature of the work activities. Field staff that operate remotely may benefit from smartphones, tablets, and notebooks. Some field staff may require embedded devices such as a vehicle-mounted tablet or notebook, such as in a police vehicle. The embedded device may include access to telemetry data being received from on-board systems.

Most modern smartphones incorporate geospatial capability to support location-based services. Technologies such as iBeacon from Apple offers greater accuracy for location detection and open new possibilities for automated event and process triggering.

With this range of devices comes a range of mobile application development models:

- ▶ **Native apps**

Mobile apps written using the device's native platform Software Development Kit (SDK), such as Java on Android, Objective-C/Swift on iOS, C/C++ for Arduino/Raspberry Pi/BeagleBone. In this model, the mobile app makes calls to a remote API for retrieving and submitting data. The presentation layer lives entirely on the device.

- ▶ **Web**

Mobile apps written using web standards including HTML, JavaScript, and CSS. These apps are served by a web server or application server and are rendered within a web browser operating on the device. In this model, the mobile app may continue to load its views remotely from the web or application server. It may also call remote APIs to retrieve and submit data. The presentation layer lives on the server.

- ▶ **Hybrid**

A mobile app that uses a mixture of native code and web standard code (HTML, JavaScript, CSS). The container, typically native in this model, loads the HTML, JavaScript, or CSS code from within the local mobile app bundle running on the device. In this model, the mobile app makes calls to a remote API for retrieving and submitting data. The presentation layer, for most hybrid apps lives on the device.

4.2.2 Mobile channel services

In connecting to enterprise systems, the mobile apps that use APIs for sending and retrieving data leverage one or more services within the mobile channel services layer (see Figure 4-2 on page 67). In this layer, various components may be included to facilitate connectivity, security, and intermediary functions. Intermediary functions include API gateway, integration adapters, data synchronization, push notification, or other custom mobile application channel functions.

Mobile channel services can be implemented on premises, cloud, or split between both. Cloud-based implementations may include using Platform as a Service (PaaS) such as IBM Bluemix, Infrastructure as a Service (IaaS) such as IBM SoftLayer®, or Software as a Service (SaaS) offerings.

Connectivity and security services

Connecting through various remote networks requires that the user is authenticated and authorized to access the requested services and APIs. Depending on the degree of security requirements, such as information security classification, the components used for security vary. In some cases, a basic security system can be used that comprises a simple basic authentication mechanism with SSL/TLS for encryption. A more stringent set of security requirements may lead to OAuth, Open ID, certificates, or VPN. The security services may also include single sign-on (SSO) capabilities to allow a user to authenticate once and access a number of downstream services without the user needing to authenticate to them individually.

The mobile app security pattern selected by an enterprise may also be influenced by the wanted user experience. Table 4-1 on page 69 includes a list of popular mobile app security patterns with their associated user experience.

Refer to Chapter 5, “Enterprise mobile security with IBM Security Access Manager and IBM Worklight” on page 83 for more details about mobile security.

Table 4-1 Mobile app popular security patterns

#	Pattern	Description	User Experience	Considerations
1	User ID and password	The user must enter their User ID and password each time the app is launched. The app uses these credentials to authenticate the user.	Low - the user must recall and enter their password each time they use the app. Depending on the password policy, it may involve a lengthy password that is difficult to enter on a smaller keyboard.	The credentials are transmitted over an encrypted session using SSL/TLS or VPN.
2	User ID and password with Save Password check box	Extending pattern 1 with the addition of a Save Password check box allows the password to be saved locally so that the user does not need to reenter it the next time the app is launched.	Medium - the user is still presented with a login page and clicks login but does not need to enter anything. Users are then taken to the main screen.	Saving the password locally is a security risk; if the device is compromised the password may be retrieved.
3	Token-based authorization, for example OAuth 2.0 or OpenID	The user performs an initial authentication using user ID and password. Once authenticated, the app is registered and given a unique token (for example, access token) that will be used by the app to make API requests. When the app is launched, the token is validated silently and the user is taken directly to the main screen.	High - the user is taken directly to the main screen.	If the device itself is inadequately secured with a system password or PIN, anyone can pick up the device, launch the app, and gain access to potentially sensitive data.
4	Token-based authorization with PIN, for example, OAuth 2.0 or OpenID	Extending pattern 3 with the addition of a PIN validation step after the app has silently validated the access token.	Medium - the user must enter a PIN (for example, four digits) as an additional security measure.	Each app is individually authorized and may be revoked from a central system.
5	Token-based authorization with PIN and biometric, for example, OAuth 2.0 or OpenID	Extending pattern 4 with the addition of a biometric capture. The PIN may be used as a backup if the biometric capture is not successful.	Medium - the user must allow for a biometric capture or a PIN (for example, four digits) as an additional security measure.	Biometric can include voice or thumbprint (for example, Apple Touch ID).

Some IBM products that support this layer include:

- IBM DataPower® XG45

Provides edge of network security to provide web-facing applications, firewall, or web services proxy, authentication and authorization services, and OAuth.

- IBM Security Access Manager for Mobile

Provides access security for web and mobile applications supporting authentication, authorization, single sign-on, and session management. It includes support for one-time password (OTP), RSA SecureID tokens, OAuth, and context aware authorization.

- ▶ IBM Mobile Connect

A Virtual Private Network (VPN) software that supports end-to-end data encryption.

Some IBM cloud-based options include:

- ▶ IBM Bluemix Mobile Application Security (MAS)

It is a cloud-based security system that supports identifying users and devices ensuring that unauthorized, compromised, or lost devices can no longer access cloud services and data.

- ▶ IBM Bluemix single sign-on services

It is a cloud-based security system that supports user authentication to web and mobile applications with a simple policy-based user access security.

Intermediary Services

The interfaces exposed by enterprise applications and systems of records can vary widely. In some cases, certain enterprise systems do not offer an easily consumable interface or one that is suitable for consumption by a mobile app. Intermediary services provide the first terminating point for mobile apps when integrating into enterprise systems or providing mobile optimized functions. In addition, direct connection from a mobile app to an enterprise system is discouraged from a security perspective.

Intermediary services can include the following capabilities:

- ▶ API gateway

The endpoint for exposed APIs that performs routing and possibly simple transformation. The API gateway may also enforce API consumption policies including throttling, app identification and secret, and developer identifier and secret.

- ▶ Device Management

Manages the deployment of enterprise devices to staff with distribution of device profiles to control device settings such as security and mobile app deployment.

- ▶ Data synchronization

Supports the synchronization of business data persisted on the mobile device and a centralized repository using technology such as multi-master replication. Data synchronization is an alternative communication pattern of information exchange to single-shot request/response-based data protocols. Data synchronization is particularly effective with scenarios involving the exchange of large and complex data sets where minimizing the data over the air is desirable.

- ▶ Messaging

Includes using protocols such as MQ Telemetry Transport (MQTT) for lightweight queue-based message exchange that supports publish/subscribe-based communication with quality of service.

- ▶ Push Notification

An alternative to Short Message Service (SMS) available with smartphone platforms that is used to minimize battery consumption due to polling. Server-side applications wanting to alert users of an event can send a notification message via the respective push notification system of the device platform. The device platform will alert the user of the message even if the associated app is not in the foreground or the device is in *suspend* mode.

Some IBM products that support intermediary services include:

- ▶ IBM API Management
Includes an API gateway that can expose APIs and either provide straight through proxying or basic data mapping. Additionally includes developer portal and API management functions.
- ▶ IBM Worklight
Provides a mobile application development platform that includes exposing APIs (through the Worklight SDK), integration adapters, push notification, and data synchronization. Additional features include geolocation and security functions.
- ▶ IBM DataPower XI52
- ▶ Provides a mechanism for exposing APIs and is the core of the IBM API Management product. It includes Advanced Integration Services and high performance data mapping for JSON and XML payloads.
- ▶ IBM MessageSight
Supports telemetry data and lightweight messaging using MQTT. MQTT is an asynchronous, bidirectional messaging protocol that may be used as an alternative to HTTP. It can also be used as a means of sending push notifications. It can support an event driven model with publish/subscribe-based topics with a range of quality of service (QoS) settings.

IBM Bluemix is a Platform as a Service (PaaS) that includes a range of services that may be deployed as part of the channel services layer. These services include both runtimes and pre-built services. IBM Software as a Service (SaaS) offerings may also be leveraged to provide intermediary services.

Some IBM cloud-based options include:

- ▶ IBM Bluemix Node.js, Liberty for Java, Ruby on Rails
Runtime environments to deploy custom application logic that may be used to either supplement enterprise application functions or provide channel specific capabilities.
- ▶ IBM Bluemix MQ Light
A messaging service that provides a flexible and easy-to-use messaging service for Bluemix applications. It can be used as a queuing service to send single messages to individual recipients by way of queues or as a publish/subscribe broker for sending notifications to multiple recipients.
- ▶ IBM Bluemix Workflow
Enables the creation of long-running, stateful workflows that orchestrate tasks and services with synchronous or asynchronous event-driven interactions.
- ▶ IBM Bluemix Cloud integration
Enables users to integrate cloud services with enterprise systems of record. It exposes the back-end systems of record as Representational State Transfer (REST) APIs to be used by applications.
- ▶ IBM API Management for Cloud
It is a Software as a Service (SaaS) implementation running entirely in cloud that offers both an API gateway and API management functions.

4.2.3 Business process management and enterprise integration

Many of the intermediary services such as those provided by IBM API Management, IBM Worklight, IBM DataPower, or the various IBM Bluemix services may support integration with enterprise applications directly. Examples of enterprise applications include customer relationship management (CRM), human resources (HR), fulfillment, provisioning, master data management (MDM), order management, scheduling, or transaction management systems. Integration may also be established directly to databases.

For cases where enterprise applications expose well defined and easily consumable interfaces, direct integration to these applications may be appropriate. Some enterprises already have a strategic integration component such as an enterprise service bus (ESB) that has existing interfaces to various enterprise applications. This approach simplifies the integration from the Intermediary services as ESBs are typically implemented with well defined, consistent service interfaces that promote reuse of services and have already catered for complex data and protocol transformation. Organizations may have legacy systems that require custom adapters using non-standard protocols and data formats.

The BPM and enterprise integration layer is where IBM BPM and integration technology may be added where the existing platform is not suitable or does not exist.

IBM BPM provides a business process engine that orchestrates activities involving human tasks and system tasks. It may include case management that encapsulates a collection of ad hoc activities, documents, and business state. System tasks may require the update or retrieval of data from external systems that include the enterprise applications and cloud services.

Long running processes

A business process may be implemented as a long running process where the process may take minutes, hours, or days to complete. This situation may be partly due to human tasks where a process is blocked until an individual performs a specific action or the process waits for an external event (wait activity implemented using an intermediate message event) to proceed.

When a long-running process is triggered by a mobile app, the user experience is improved if the mobile app does not block or wait for the long running process to complete. The trigger is an asynchronous one where the mobile app receives an acknowledgment of it calling IBM BPM but not including the final outcome of the process.

A reference to the instantiated process (see Figure 4-3 on page 73) in the form of, for example, a REST API URL resource can be returned. The mobile app can then request the status of the instantiated process using the returned reference resource at a later stage. The mobile app does not need to maintain the reference as it should be able to query the IBM BPM system in the future to retrieve the list of one or more references for active process instances related to the user context.

In addition to retrieving the status of the instantiated process, the IBM BPM process may also include tasks to periodically send push notifications to the mobile app via timers. In that case, the IBM BPM system uses an external push notification service such as that provided by IBM Worklight to inform the mobile app and its user of any further steps or milestones achieved by the process.

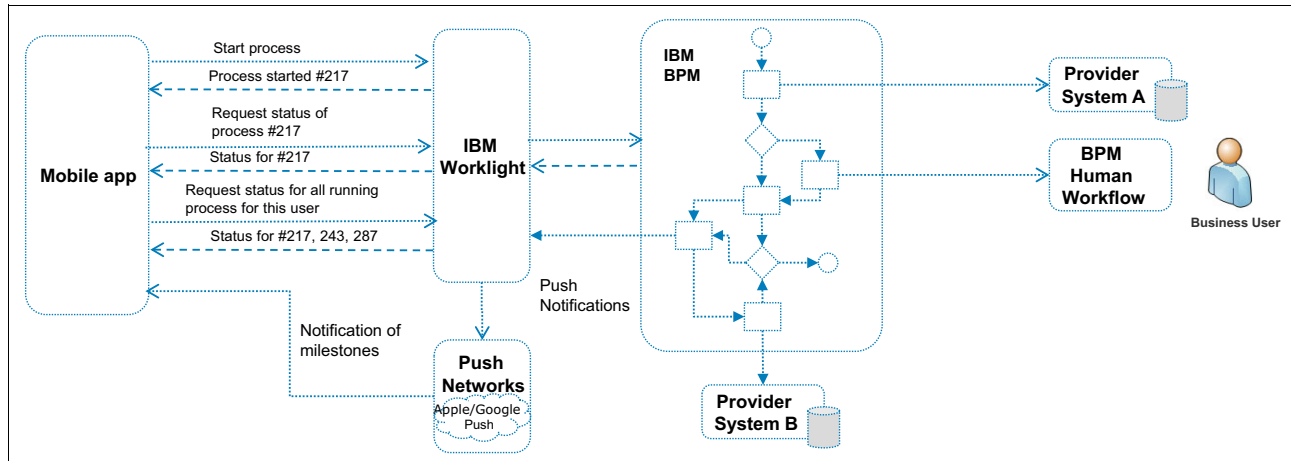


Figure 4-3 Mobile app invoking long running processes

Within this architecture layer, the BPM component provides a process engine that supports the implementation of automated business process orchestration. The business process engine manages the execution of business processes that comprise a mixture of human interaction services, accessed via a user interface such as a browser, and automated steps that may require orchestration of multiple provider system calls. Advanced transactional management including compensation may also be included within the BPM component. IBM Business Process Manager software is the flagship BPM software from IBM, available as Express Edition, Standard Edition, and Advanced Edition.

While the BPM component may be accessed via a graphical user interface (GUI) by process participants, it can also be accessed via its exposed REST APIs. The REST APIs can be accessible to the channel services layer and subsequently allow business process resources to be consumed by mobile applications.

Some IBM enterprise integration products that support advanced integration include:

- ▶ **IBM Integration Bus (formerly IBM WebSphere Message Broker)**
Provides a universal integration foundation based on an enterprise service bus (ESB) supporting SOA and non-SOA environments. It supports various protocols and interface types such as HTTP, JMS, WebSphere MQ, database and file connectivity with support for mediation flows, transformation, and routing.
- ▶ **IBM DataPower XI52**
An integration hardware appliance or virtual appliance, the XI52 provides message transformation, integration, and routing. It is built for simplified deployment and hardened security, bridging multiple protocols, and performing conversations at wire speed.
- ▶ **IBM Business Process Manager (BPM) Advanced Edition**
It includes advanced mediation flows, transformation, orchestration, transaction management, and compensation. It also includes various protocols and interface types such as HTTP, JMS, WebSphere MQ, database and file connectivity. Orchestration services can be implemented using Business Process Execution Language (BPEL) and mediations.

Some IBM cloud-based options for integration include:

- ▶ **IBM API Management on Cloud**
A Software as a Service (SaaS) implementation of IBM API Management

- ▶ IBM Bluemix Cloud integration
Enables users to integrate cloud services with enterprise systems of record. It exposes the back-end systems of record as REST APIs to be used by applications.
- ▶ IBM Business Process Manager (BPM) for Cloud
A Software as a Service implementation of IBM BPM Advanced.
- ▶ Cast Iron Live
Is a multi-tenant, cloud-based platform for integrating cloud and on-premises applications and enterprise systems in a hybrid environment. It enables you to configure, run, and manage integration in the cloud without any infrastructure footprint.

4.3 Deployment topology

An implementation of the mobile architecture can be deployed using cloud or on premises infrastructures. Figure 4-4 on page 75 shows an example solution comprising IBM Worklight, IBM BPM, and IBM Security Access Manager. This topology depicts an on premises implementation and follows a Defense-in-Depth principle for secure mobile access by remote enterprise staff. The example is based on an OAuth 2.0 security mechanism and with single sign-on to IBM BPM via IBM Worklight. IBM Worklight and IBM BPM use a shared directory server.

The connectivity and security layer is implemented using IBM Security Access Manager for Web. IBM Security Access Manager can be deployed as two components to allow for locating its components across multiple security tiers. IBM Security Access Manager for Web provides a security gateway function using an underlying reverse proxy acting as a policy enforcement point (PEP).

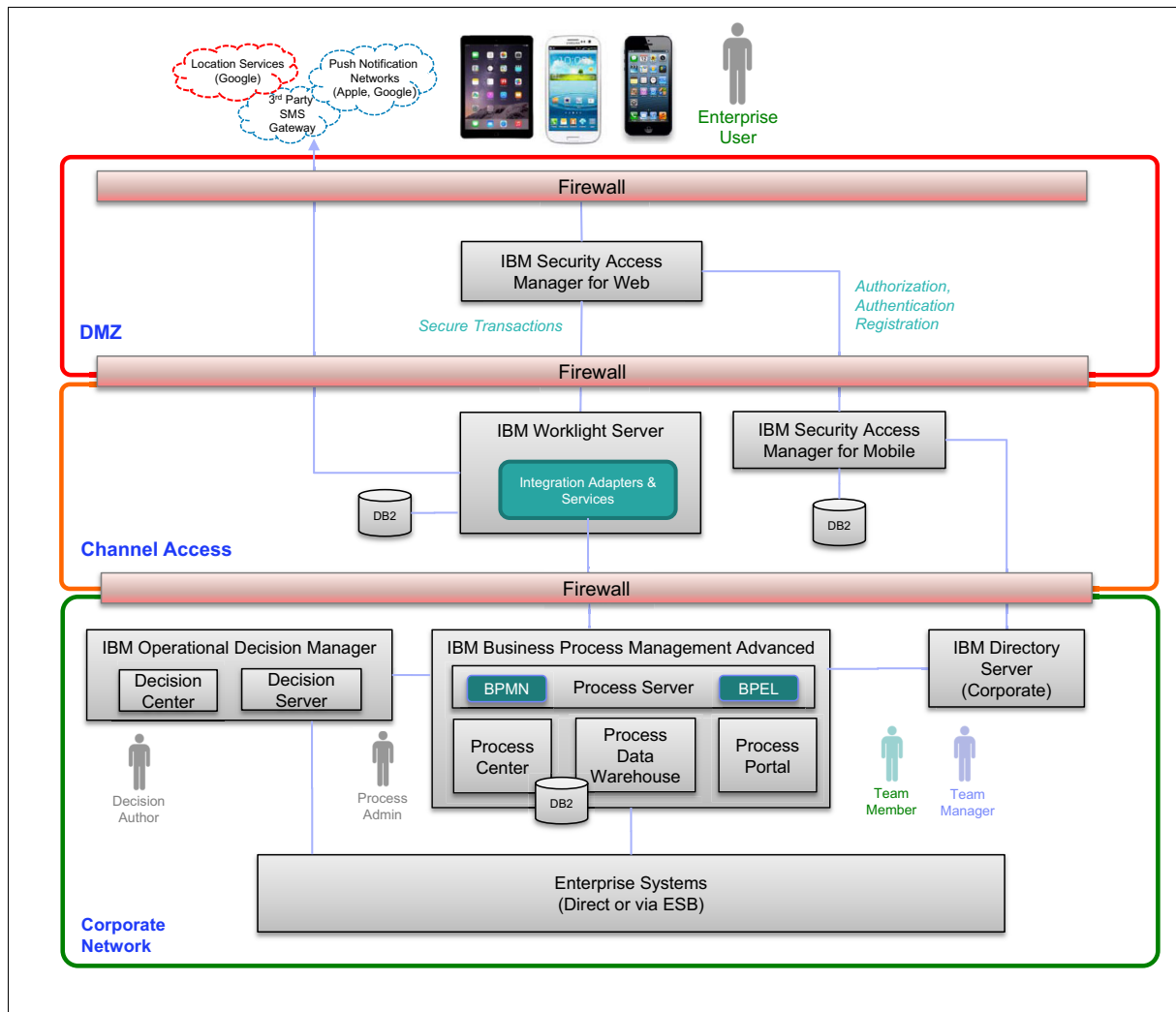


Figure 4-4 Example mobile solution topology with Worklight, IBM BPM, and IBM Security Access Manager

Refer to Chapter 5, “Enterprise mobile security with IBM Security Access Manager and IBM Worklight” on page 83 for more details about security technical implementation based on IBM Security Access Manager.

4.4 IBM BPM interaction patterns

Interaction with IBM BPM is supported via both human-based interaction and direct system interaction. Human interaction includes using a user interface that is served by IBM BPM. Direct system interaction involves an external application that may itself support human interaction, integrating directly with service interfaces exposed by IBM BPM. Figure 4-5 on page 76 illustrates the key interaction patterns that are available in IBM BPM, numbered 1 - 6.

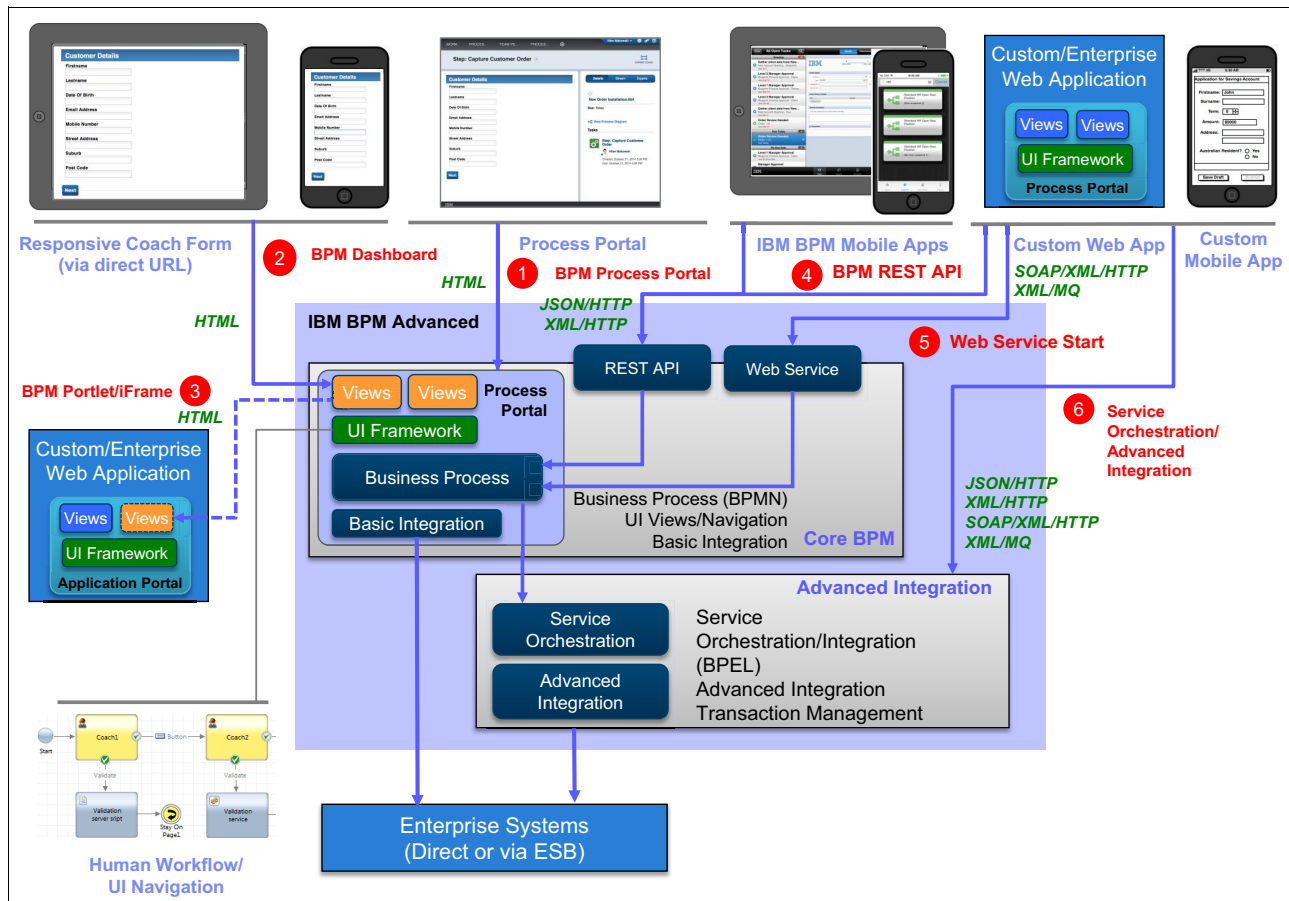


Figure 4-5 IBM BPM interaction patterns

4.4.1 Pattern 1: IBM BPM Process Portal

Using IBM BPM human task capability, views can be created as part of a business process to facilitate the capture and presentation of business data. The views, implemented as coach forms within IBM BPM, can be easily constructed using IBM Process Designer with drag and drop from a palette of user interface elements onto the coach form canvas.

New in IBM BPM v8.5.5 are responsive user interface elements that allow for implementing coach forms that can self-adjust and reflow themselves based on-screen resolution. This responsive design approach enables a coach form to be rendered effectively across different categories of devices, smartphone, tablet, and desktop with varied resolution within each category.

Coach forms may be accessed via the IBM BPM Process Portal. The Process Portal segments its user interface into sections within which the process participant can view, accept and manage activities and tasks, collaborate with other participants, and observe the process and analytics. A coach form may have been implemented using responsive user interface elements, however, if the process participant accesses the coach form within Process Portal from a mobile device with a limited screen resolution, the responsive capabilities of the coach form may not be optimal due to the encapsulating Process Portal frames (in IBM BPM v8.5.5, Process Portal does not use a responsive design).

4.4.2 Pattern 2: IBM BPM dashboard

An alternative method to using the IBM BPM Process Portal is to access the coach form directly via its URL path. The URL path is static and can be determined by calling the IBM BPM REST API GET `/rest/bpm/wle/v1/exposed/service`. It returns a list of exposed services including exposed human services that comprise coach forms. Each item returned includes a `runURL` property from which the exposed coach form can be accessed directly outside the IBM BPM Process Portal container. This approach allows the responsive design features of the coach form to be effective, adjusting and reflowing itself based on-screen resolution.

4.4.3 Pattern 3: IBM BPM portlet and iFrame

An existing enterprise application or a new custom web application being developed containing views and forms as part of its primary feature set may be required to incorporate the IBM BPM user interface such as a coach form. The coach form is rendered within an HTML iFrame covering a portion of the overall page. To implement this pattern, the external UI container (for example a CRM web application) has to first determine the task ID that corresponds to the wanted coach form. The IBM BPM REST API `ExecuteQuery` can be used to return a list of tasks. This REST API can first be called by the external UI container and parsed for the task and its task ID to use. The REST API is as follows:

```
https://<bpmhostname>:<port>/rest/bpm/wle/v1/search/query?organization=byInstance&run=true&shared=false&filterByCurrentUser=false
```

Once the task ID is known, it can be substituted in this URL to display the coach for the task ID. The URL format is:

```
http://<bpmhostname>:<port>/teamworks/process.lsw?zWorkflowState=1&zTaskId=<Task ID>
```

4.4.4 Pattern 4: IBM BPM REST API

IBM BPM provides a set of APIs to enable external application interfacing with IBM BPM. The APIs include support for interacting with IBM BPM at various different levels including process, activities, tasks, and services in addition to operational functions.

IBM BPM REST API allows for scenarios where the presentation layer of an application is implemented externally and not through coach forms. An application could be in the form of a custom web application, a third-party application, or a custom mobile application (either hybrid or native). The IBM BPM mobile app for iOS (for iPhone, iPad, or iPod touch devices) also uses the IBM BPM REST API.

The complete list of IBM BPM REST APIs can be found online at the following link:

http://www-01.ibm.com/support/knowledgecenter/SSFPJS_8.5.5/com.ibm.wbpm.ref.doc/covw_bspa_ref.html?lang=en

The IBM BPM REST API tester is a tool that is included with IBM BPM to test the REST API using test data via the browser. It can be accessed from the following link:

```
https://<hostname>:<portnumber>/bpmrest-ui/BPMRestAPITester/index.jsp
```

The IBM BPM REST APIs for business process definition (BPD)-related resources are grouped as follows:

- ▶ **Process API:** Used to interact with business processes including starting, suspending, and stopping processes. The status of a process can also be retrieved.
- ▶ **Service API:** Used to interact with a service such as a human service or integration service. The exposed services can be retrieved including starting the service, retrieving and setting data of a service, and stopping a service.
- ▶ **Task API:** Used to interact with tasks including starting, and finishing a task. Task data may also be set and retrieved. The scenario 1 described in Chapter 6, “Scenario 1: Getting started” on page 111 uses the Task API to allow the Field Service app to retrieve work orders and update work order status and report data via Worklight.
- ▶ **Search API:** Supports the execution of queries against IBM BPM that returns filtered results when searching across processes, tasks, or business data within tasks. Scenario 1 described in Chapter 6, “Scenario 1: Getting started” on page 111 uses the Search API to allow the Field Service app to retrieve the list of available work orders for the field technician team. A work order correlates to a task in this scenario where the translation between the two is performed in Worklight as part of the API facade pattern (see section 4.4.5, “Pattern 5: API facade” on page 78).
- ▶ **Social API:** Used to enable social interaction around processes including adding comments to tasks, following a task, and retrieving the stream of collaboration activities about a task.
- ▶ **Task template API:** Used to retrieve a specific task template, the client settings for a specific task template, and APIs to query data about task templates.
- ▶ **Activity API:** New in IBM BPM v8.5.5. Used to integrate with ad hoc activities.
- ▶ **Business object API:** Used to retrieve details about a particular business object.
- ▶ **Organization API:** Used to manage and retrieve details about users and groups.
- ▶ **Process documentation API:** Used to retrieve details and assets defined in a process.
- ▶ **Event handling API:** Only one API to send an Enterprise Content Management event to IBM Business Process Manager.
- ▶ **Other:** Various APIs to retrieve system information, exposed artifacts, and so on.

4.4.5 Pattern 5: API facade

Some of the IBM BPM REST APIs such as the Process API, Service API, Task API, Activity API, and Search API are very much IBM BPM-oriented. Their resource names, parameters, and body attributes may not be adequately self-describing within the business context of a mobile app that its developers can use intuitively.

Figure 4-6 on page 79 shows an example facade where Worklight is used as an intermediary component within the channel services layer of the mobile architecture to provide a set of adapters exposed as business-oriented operations. The mobile app invokes these operations, named in business terms (in this case for a Field Service mobile app) `getWorkOrders` and `acceptWorkOrder`. These business-oriented operations are mapped to IBM BPM REST API resources. The IBM BPM REST API typically returns many more attributes than what is required or applicable to the API that the mobile app needs to consume. Therefore, the intermediary, in this example Worklight, can filter these unwanted attributes or perform semantic or syntactic transformation of attributes as required.

Similarly, a facade pattern can be implemented with an intermediary provided by an API gateway exposing a REST API where API resources are given business-oriented names and mapped to associated IBM BPM API resources.

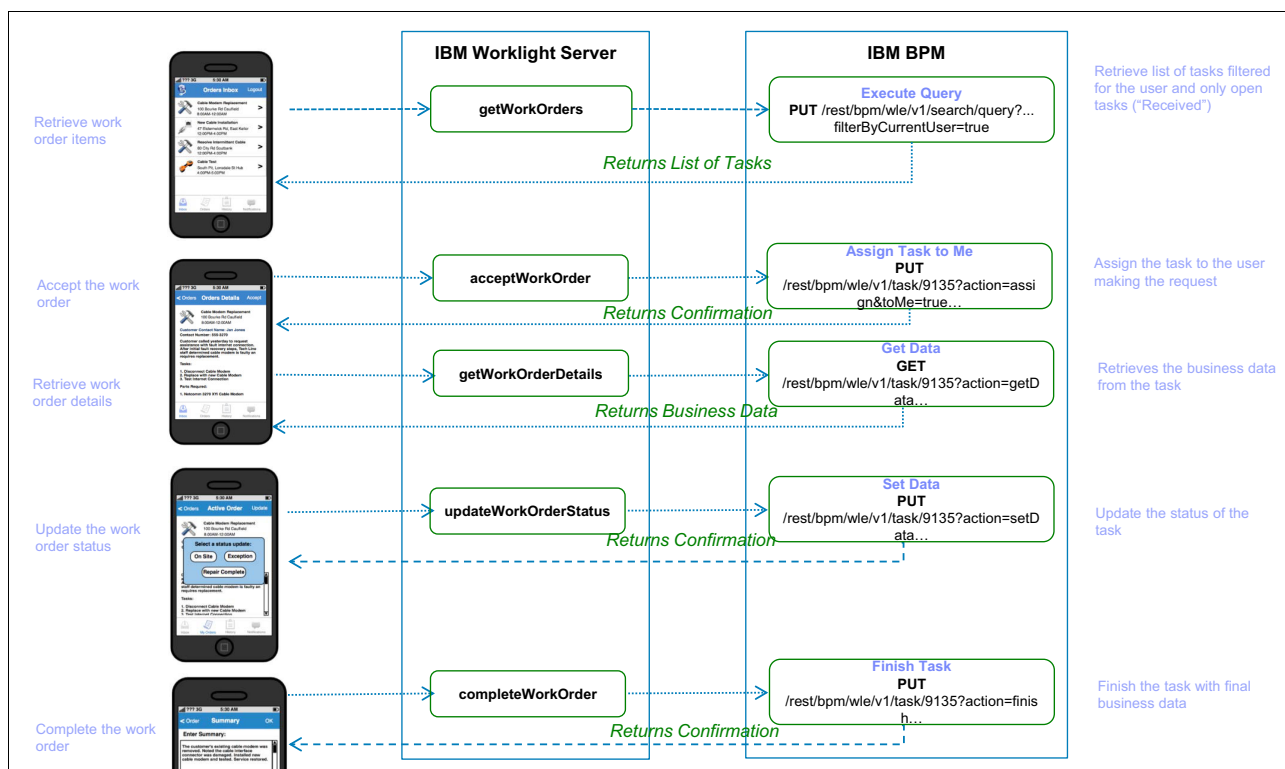


Figure 4-6 Worklight facade for IBM BPM REST API

4.4.6 Pattern 6: Message or web service start

A business process within IBM BPM can be started via the Process Portal by a human or via a system using web service start also known as *inbound web service* (see *Creating an inbound web service* at the following link):

http://www-01.ibm.com/support/knowledgecenter/SSFPJS_8.5.5/com.ibm.wbpm.wle.editor.doc/topics/building_sample_inbound_integration_F.html

The web service is defined as SOAP service with an XML payload. An external application such as a web application or mobile application can pass business data into the web service start so that the business process can start with data captured externally.

An example scenario is where a customer support application is used to capture a service call and the customer support application then invokes an escalation business process in IBM BPM. The implementation involves the customer support application calling the exposed escalation process (exposed by web service start in IBM BPM), passing in the support business data for context.

JMS messages can also be used to trigger a business process.

To get started, see *Building a sample inbound integration* at the following link:

http://www-01.ibm.com/support/knowledgecenter/SSFPJS_8.5.5/com.ibm.wbpm.wle.editor.doc/topics/building_sample_inbound_integration.html

4.4.7 Pattern 7: Service orchestration and advanced integration

The IBM BPM Advanced Edition includes the Advanced Integration Services (AIS) component within the Process Server that supports the execution of mediation flows and BPEL created with IBM Integration Designer. The mediation flows available in IBM Integration Designer provide additional integration capabilities beyond what is provided by IBM Process Designer. These capabilities include advanced transformation, routing, protocols, and interface connectors. The BPEL engine within IBM BPM Advanced Edition provides support for creating services that comprise orchestration of system activities including transaction management with quality of service and compensation and rollback. These advanced features can be used to perform complex system integration across enterprise applications and legacy systems. It may be used with the IBM Integration Bus if non-standard or custom connection adapters are required that are not supported by IBM BPM Advanced Edition.

4.4.8 Pattern selection guide

Table 4-2 provides guidelines on when to use the patterns described in section 4.4, “IBM BPM interaction patterns” on page 75.

Table 4-2 Pattern selection guide

Pattern	When to use this pattern
1. IBM BPM Process Portal	<ul style="list-style-type: none">▶ Where a business process is required to be implemented rapidly including user interfaces and automated system workflow.▶ Device category is tablet or desktop/notebook with larger screen resolution.▶ As an alternative to building a custom application that encompasses most of a business process.▶ Leverage an out-of-the-box Process Portal to facilitate the activity and task access and management within the business process.
2. IBM BPM dashboard	One user task with one or more coach forms to be exposed as a static URL to a browser running on either small or large screen devices. The exposed user task can be used to start a new process, integrate with an existing process, or integrate with an external system.
3. IBM BPM portlet and iFrame	Rapidly implement business process and user interfaces using IBM BPM Process Designer while incorporating some parts of the business process user interface within an existing web-based application. Note that a heritage human service that is exposed as a dashboard can be exported as a portlet suitable for IBM WebSphere Portal
4. IBM BPM REST API	Implement the presentation layer of one or more portions of a business process as part of a mobile or web application.
5. API facade	Use this pattern when the IBM BPM REST API is to be exposed within a business concept for mobile app developers to consume. This pattern allows to filter many of the IBM BPM-focused attributes within the payload as they may not be needed by the mobile app. In addition, the URI resource path can use business-specific terminology rather than the generic IBM BPM-focused terms.

Pattern	When to use this pattern
6. Web service start	Start a business process using a SOAP/XML-based web service called from a mobile or web application. Business data can be passed in with the call.
7. Service orchestration and advanced integration	Perform complex integration with an external system that requires advanced mediation flow, orchestration, and transaction management.



Enterprise mobile security with IBM Security Access Manager and IBM Worklight

IT security departments are key decision makers in the design and implementation of any enterprise mobile app program. The risks to be mitigated in such programs vary depending on the type of apps being developed, the target users, and the value of the data being accessed.

The breadth of mobile security concerns spans multiple domains, including application security, device security, fraud, service security, and governance. Hence, it is critical that a broad view is taken on the risks a mobile app presents, so that IT Security can propose controls and countermeasures to address those risks.

From an IBM perspective, an entire division is dedicated to the security domain. Whether an enterprise is interested in infrastructure, cloud, mobile, access and identity or applications, this division is responsible for delivering end-to-end security solutions for IBM customers.

This chapter describes the most important aspects of mobile security that enterprises must consider when designing a mobile app security program. This chapter introduces features of IBM Security Access Manager and IBM Worklight for mobile security.

5.1 Security principals

This section describes the basic principles such as authentication, access control, and single sign-on as they apply to the mobile domain.

5.1.1 Authentication

Authentication is the act of identifying a particular account (and mostly an identity) for accessing online services. For web access, authentication is typically done through an

identity such as user name and password, that is, something you *know*. This approach is generally thought of as a weak authentication mechanism.

More recently, web applications have become reliant on authentication based on something you *have*, such as tokens or one-time-password. In addition, something you *are* technologies, for example, biometrics, has been used in physical access scenarios.

Many methods have been attempted to combine authentication mechanisms to improve assurance, for example tokens, certificates, and so on. However, these methods have largely been rejected by the user population and the business, as the cost of bearing them has rendered implementations cost prohibitive.

With the proliferation and accessibility of mobile devices, the opportunity for apps to leverage hardware capabilities is driving authentication assurance improvements. In addition, software standards such as OAuth are driving alternative token implementation mechanisms. The combination of these standards with Mobile is driving new patterns around Access Management.

5.1.2 Access control

Access control comes in various forms within web applications today, and this document assumes that the reader is familiar with standard user and resource access control methods.

This section considers the two additional authorization aspects encountered within mobile app design discussions: device access control and app access control.

Device access control

Web Access Management techniques have evolved to include considering context information. This context information is multi-faceted, and includes the device, the environment, and the user.

Knowing that a device is compromised is critical input to a transactional use case, and Access Management solutions, such as IBM Security Access Manager for Mobile, when combined with IBM Security Trusteer can provide insight into the device and its trustworthiness.

App access control

The OAuth standard provides a delegated authorization solution that allows resource owners to specify who can access their resources, and what scope should be granted. The concept of scoped access has been historically implemented using physical access systems. For example, a contractor may be given a temporary access badge with access scoped for low privilege access. Enforcement points on physical doors control whether the badge may be granted access.

In the same way, a user is able to delegate permission for a third-party service to have limited access to their set of resources. This scope of access is bound to a token provided to the third-party service. Hence, in web or mobile enforcement points, the concept of scope has a technical implementation that controls what a token provides access to.

In a Mobile use case, the concept of delegated authorization allows a user to delegate to a mobile app, the ability to perform operations on the service. The enforcement of such policy becomes part of the context used for performing run time authorization. This approach allows a user to dedicate different authorization scopes to different Apps, and for a service to enforce that scope.

One important consideration is security controls that are associated with the installation and registration by the user of the app running on the device. By combining mobile app registration processes with device access control and app access control mechanisms, security controls can be greatly enhanced. The use case described in section 5.8, “Example OAuth flow with PIN validation using IBM Security Access Manager for Mobile” on page 101 provides a mechanism to securely register an app, allowing for future run time authorization decisions to assert that the app was registered by the user on this device.

IBM Security Access Manager for Mobile provides support for implementation of these complex use cases, and will be demonstrated in the example in section 5.8, “Example OAuth flow with PIN validation using IBM Security Access Manager for Mobile” on page 101.

5.1.3 Server side single sign-on

From a mobile app deployment perspective, single sign-on (SSO) occurs in the same manner as that for web applications. SSO can be defined as the ability to pass authentication credentials from one application to another to establish an authenticated user session. The authentication credentials chosen can vary, relying on the level of trust between the two applications. An example is as follows:

A user accesses a web application using their browser. To access the application, the user's request passes through a web enforcement point. This enforcement point has policy that requires a user to authenticate to access the web application. The user authenticates using a user name and password. The web enforcement point then validates the user name and password before passing the request to the web application. Since this web application needs to know who the user is, the enforcement point passes an HTTP header that contains the user name. It encapsulates this request in a secure payload, using SSL.

In this example, the application is able to trust the user name in the HTTP header because it trusts (via SSL) the web enforcement point.

In a mobile app example, this pattern does not change. The main difference is that the payload within the HTTP request is likely to be JSON/REST, but this fact has no effect on the SSO mechanism being used.

Numerous methods for providing user credentials as part of SSO operations are supported in IBM Security Access Manager:

- ▶ LTPA: An encrypted token that contains user (or LDAP DN) information.
- ▶ Kerberos: A Kerberos token representing the user, established with a Windows domain controller.
- ▶ HTTP header: Simply the user name in an HTTP header.
- ▶ Basic authentication: A user name and password (it can be a generic system account).
- ▶ Form-based SSO: A mechanism that is able to intercept a web login form and automatically POST credential data to the web application.

All of these mechanisms are well established capabilities of IBM Security Access Manager; the choice of which type to rely upon depends on the web application that requires SSO to occur.

5.2 Mobile security principals

Unlike publicly developed and delivered mobile apps, enterprise mobile development must cater to two distinct target audiences, mainly company staff or customer access. These two access patterns require different considerations. In the case of staff access, the data being accessed may be critical to the business, so appropriate controls must be put in place to secure the numerous risk vectors associated with releasing such data to a device. For customer access, an open service must be made available for any customers, regardless of the state of their devices. Transactional mobile apps must be aware of the threat vectors associated with financial fraud.

The following sections consider these different patterns in detail, along with the risks that they represent.

5.2.1 Staff (internal and external)

Staff applications can be generally referred to as being under the management of the enterprise. This control may extend to the device itself, mandating that any device that is to be used on an enterprise network run endpoint software so the device can be managed. IBM Endpoint Manager is an example of such software. Having such control then allows the organization to trust the device for accessing mobile apps through an enterprise app store. In doing so, adequate controls need to still be provided so that when these mobile apps are accessing enterprise applications, server-side protections are put in place, ensuring:

- ▶ Devices are known and appropriately secured: An endpoint management type solution for registering the device and ensuring its trustworthiness.
- ▶ Users are authenticated: Access control mechanisms are in place to ensure that users are identified for accessing the resources.
- ▶ Apps are authentic: No malware or other app is interfacing with the enterprise services.
- ▶ Users and their devices are authorized to access apps: Not simply the user, but the combination of device, app, and the user are authorized to access the service.

Of course, standard security controls such as Identity Management and audit are assumed.

5.2.2 Customer (external)

The security requirements listed for staff mobile apps in 5.2.1, “Staff (internal and external)” on page 86 are also valid for customer access use cases. However, some aspects of the implementation must be established using different techniques.

Consumer applications run on devices that are largely uncontrolled. Therefore, the device itself cannot run software mandated by the enterprise delivering the mobile app. This fact limits the mechanisms that can be employed to control and monitor the device. As such, a broader set of techniques must be employed in order to secure an app running on a customer's device. These additional requirements include:

- ▶ Authenticity checking: Ensuring that upon start, the code executing is authentic and has not been tampered with. Since the device is untrusted, the app source code itself needs protection.
- ▶ Device trustworthiness: Understanding that the device is trustworthy, that it is not hosting malware and is not jailbroken. It is useful to know whether a device should be trusted when performing transactions.

- User threat protection: Ensuring that the user accessing the services has not been compromised, that the account being used has not been the victim of attacks. When transactional operations are occurring, knowing whether an account has been subject to a take-over threat is useful.

The white paper *Getting Started with Identity and Access Management for Mobile Security* available on the IBM Identity and Access Management (IAM) Business Value Accelerators introduces you to mobile business scenarios and the challenges associated with them that cannot be addressed using traditional IT security. The white paper also explores the identity and access maturity model for mobile security, and provides reference implementations of each level using IBM Security products. The white paper can be downloaded from the following site:

<https://ibm.biz/isammobileaccelerator>

5.3 IBM Worklight security overview

IBM Worklight integrates security into the entire mobile application lifecycle. IBM Worklight safeguards mobile security at the device, application, and network layer:

- Protects sensitive information from malware attacks and device theft.
- Ensures timely propagation and adoption of critical security updates to the entire installation base.
- Enforces multi-factor authentication, SSO, and device SSO while integrating with existing authentication and security approaches.
- Enables secure delivery and operation of mobile apps for employee-owned devices or device types that are not allowed on the corporate network.
- Manages approved and rejected devices for controlled mobile application installation and remote application disablement.

This section provides an overview of the IBM Worklight security capabilities. For detailed information about Worklight security, see the following documents:

- IBM Redbooks publication *Securing Your Mobile Business with IBM Worklight*, SG24-8179.
- IBM Redbooks Solution Guide *Enhancing Your Mobile Enterprise Security with IBM Worklight*, TIPS1054.
- IBM Knowledge Center *Worklight security overview* at the following site:
http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.deploy.doc/admin/c_security_overview.html?lang=en

5.3.1 IBM Worklight security capabilities

Worklight provides a set of security capabilities that address the following mobile app security objectives:

- Protecting data on the device
- Securing the application
- Enforcing security updates
- Providing robust authentication and authorization
- Streamlining corporate security processes

Figure 5-1 shows the mapping of mobile security objectives and IBM Worklight security capabilities, which are described in this section.

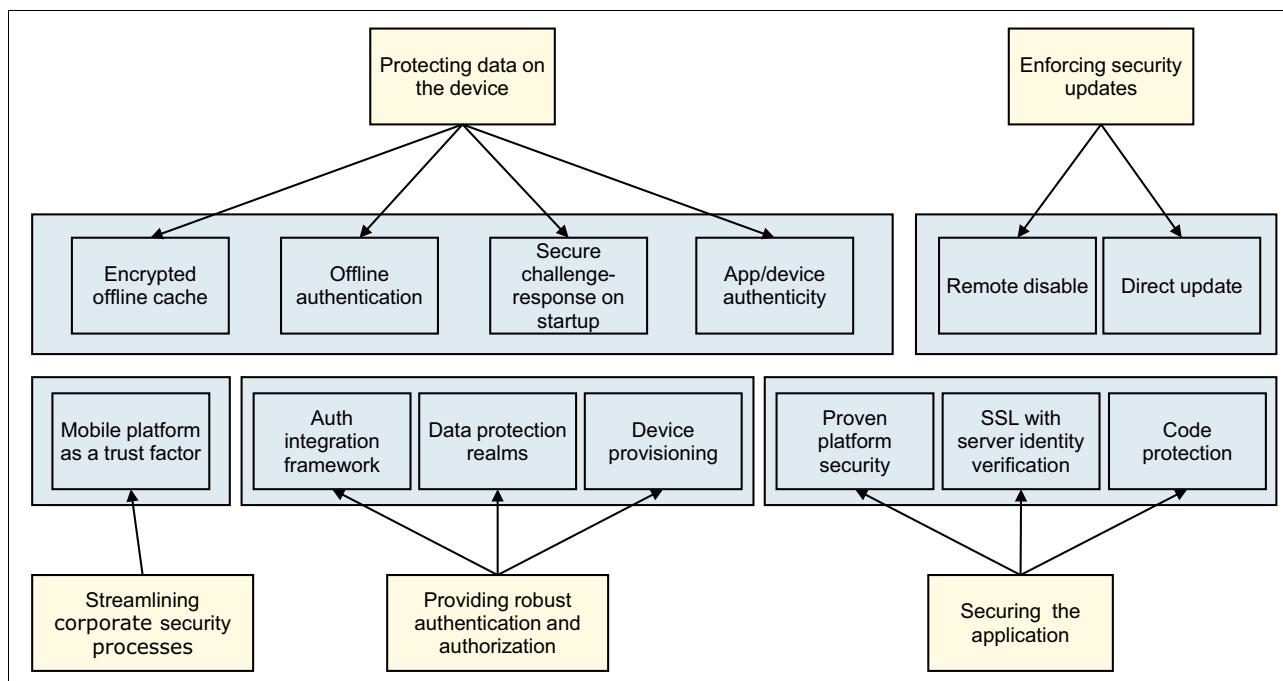


Figure 5-1 Mapping of Worklight security capabilities to mobile security objectives

Protecting data on the device

On-device storage of data can be tampered with by malware on the device; if the device is lost or stolen, sensitive data can be extracted by unauthorized third parties. Worklight provides the following capabilities to protect data on the device:

- ▶ Encrypted offline cache

Worklight encrypts data on the device by using Advanced Encryption Standards (AESS) and Public Key Cryptography Standards (PKCS). The data can be stored on the device as a cache or in the Worklight mobile storage JSONStore.

- ▶ Offline authentication

When applications are running on mobile devices that are not connected to the network, the need for user authentication still exists. The encrypted cache feature in Worklight can be used to achieve more offline authentication because only the correct passwords can unlock the offline cache.

- ▶ Secure challenge-response on start

Worklight provides extended authentication with a server by using secure challenges and responses.

- ▶ Application and device authenticity

Worklight provides application and device authenticity to ensure that only valid applications on authorized mobile devices can be used. Worklight generates a unique identification for the application and the device, and protects them from tampering by using digital signatures. Whenever the application tries to access back-end systems through the Worklight Server, the server verifies the application authenticity and device authenticity if activated and allows access only from legitimate applications.

Securing the application

IBM Worklight protects the application and prevents hackers from unpacking the legitimate mobile application and repackaging it with malicious code:

- ▶ Proven platform security

IBM Worklight has security mechanisms that are deployed by enterprises with extreme security requirements, such as top-tier financial institutions. Running IBM Worklight on IBM WebSphere Application Server further strengthens its security features with those provided by WebSphere Application Server.

- ▶ SSL with server identity verification

IBM Worklight enables a security-rich client and server communication over HTTPS to prevent data leakage and to prevent automatic server certificate verification to thwart known attacks, such as a man-in-the-middle attack.

- ▶ Code protection

Worklight provides capabilities to obfuscate and encrypt the application code and web resources to prevent tampering of the application.

Enforcing security updates

Worklight offers the direct update and remote disable features to help administrators ensure that critical updates to their applications are delivered in a timely manner:

- ▶ Direct update

The direct update feature enables developers to drive updates of the web content of the deployed HTML5 and hybrid applications directly from the Worklight Server upon application start.

- ▶ Remote disable

The remote disable feature provides administrators with the ability to disable the old version of the application for situations in which the distribution of a security fix requires that users get the new application version from the application store.

Providing robust authentication and authorization

Worklight provides the following features for authentication and authorization:

- ▶ Authentication integration framework

Worklight provides a server-side architecture that integrates with a back-end authentication infrastructure that is based on Java Authentication and Authorization Service (JAAS) with authentication realms and a client-side framework or asynchronous login requests on session expiration.

- ▶ Data protection realm

In Worklight, resources are protected by authentication realms. When a user attempts to access a protected resource, Worklight checks whether the user is already authenticated according to the process that is defined for the realm of the resource. If the user is not authenticated, Worklight triggers the challenge-response process of obtaining the client credentials and verifying them as defined in the realm.

- ▶ Device provisioning

Worklight offers the device provisioning feature to validate device identities. Device IDs are used to identify unique devices with the Worklight Server. A certificate must be created that is handled by an external trusted authority, which enhances security by signing the key pair.

Streamlining corporate security processes

A Worklight hybrid mobile app consists of web resources (HTML5, JavaScript, and CSS3) surrounded by a Worklight provided native mobile platform shell. This shell provides access to the native mobile device capabilities, such as the camera, a common user interface, the Worklight authentication framework, and security configuration.

Worklight apps can use customized shells that provide common components used by all apps in your enterprise, allowing access to only specific features of mobile devices, and enforcing the security requirements of the mobile apps in your organization. This approach helps companies ensure that apps using Worklight are trusted entities that adhere to corporate security policies, and therefore speed up the approval process.

Worklight also supports the concept of using the mobile device as a trust factor, where certificates provisioned to the device by a trusted third party (for example, a mobile device management (MDM) solution) are accepted as part of the authentication process.

5.3.2 What is new in IBM Worklight v6.2 security

This section lists the main security enhancements included in IBM Worklight v6.2.

Secured simple data sharing

Starting with IBM Worklight v6.2, you can securely share lightweight information among a family of applications on a single device.

Direct update improvements

Starting with IBM Worklight v6.2, direct update is improved in the following ways:

- ▶ Direct update is now part of the Worklight security framework. Apps developed in Worklight Studio v6.2 and later are defined as a security realm. Apps developed in earlier versions of Worklight Studio use the mechanism provided with v6.1.
- ▶ Support is extended to include Windows Phone 8.
- ▶ Direct update requests can be cached on a content delivery network server.
- ▶ Direct update process and interface can be customized.

Enhanced application authenticity

Starting with IBM Worklight v6.2, application authenticity is enhanced for iOS and Android.

Restricting database user permissions

When the databases are operational, you can create a database user with restricted privileges. You use this database user to perform database underlying operations from the Worklight Console. The user credentials appear in the application server configuration.

Integration with IBM Trusteer

Starting with v6.2, IBM Worklight supports full integration with the IBM Trusteer Mobile SDK, for Android and iOS applications.

5.4 Extending Worklight security with IBM Security Access Manager

Worklight mobile apps can be enhanced with authentication and authorization features not available in the core product. For example, you can use the IBM Security Access Manager features such as risk-based access (RBA), context-based access (CBA), strong authentication (one-time password), and identity aware applications (OAuth) to further enhance security.

The Worklight mobile app must be configured to respond to an authentication challenge from IBM Security Access Manager for Web, providing valid IBM Security Access Manager credentials or OAuth tokens. When using RBA, the Worklight mobile application must be configured to handle a step-up authentication using one-time password (OTP), which can be displayed in a web view or built in the Worklight mobile application.

5.5 IBM Security Access Manager for Mobile overview

IBM Security Access Manager for Mobile provides mobile access security protection addressing mobile security challenges by proactively enforcing access policies for web environments and mobile collaboration channels. It extends upon the web reverse proxy, web application firewall, load balancer, and session management features of IBM Security Access Manager for Web, which is used as the Policy Enforcement Point (PEP) for IBM Security Access Manager for Mobile authorization decisions.

IBM Security Access Manager for Mobile enables secure access to mobile and web applications with single sign-on and session management. It improves identity assurance with built-in and flexible authentication schemes such as one-time password (OTP). It enforces context-aware authorization, supports device finger printing, geographic location awareness, and IP reputation techniques, which enable risk-based access (RBA) and context-based access (CBA) to determine and score risk levels using user attributes and near real-time context.

IBM Security Access Manager for Mobile provides a central, policy-based user authentication and authorization system to guard against security threats to centrally monitor and control user access to mobile applications and devices. Identity assurance is improved with built-in and flexible authentication schemes including passwords, HTTP header based, IP address, Kerberos, OAuth, X.509 certificate as well as allowing for custom methods. Authentication policies can be defined to require multi-factor authentication schemes that require users to prove their identities in more than one way and implement strong authentication.

A self-service user interface is provided for device registration and access revocation, eliminating the need for user-identity and password-based login and combined with IBM Security Access Manager for Web for full-featured Access Management and application protection.

IBM Security Access Manager for Mobile offers built-in support for IBM Worklight applications to implement session management including strong authentication for client apps and context-based access for Worklight adapters such as transfer amounts in a banking application. Worklight challenge handlers are provided for traditional user name and password authentication as well as OAuth enabling Worklight developed apps to drag and drop IBM Security Access Manager authentication into their projects with little effort and accelerating development. Single sign-on from IBM Security Access Manager to Worklight is

supported using HTTP header and LTPA ensuring that features such as device single sign-on can be implemented when securing Worklight with IBM Security Access Manager.

For integration documentation and sample apps, see *IBM Security Access Manager for IBM Worklight* at the following site:

<https://ibm.biz/isamworklight>

5.6 Planning for OAuth using IBM Security Access Manager for Mobile

Mobile applications that provide basic authentication typically require users to enter their primary credentials when they log in. This method is not ideal on a touch-enabled mobile device because the user must enter long user IDs and passwords every time the application is started or when an inactivity timeout occurs.

This requirement introduces several attack vectors, such as weak passwords, denial of service through account lockout as a result of multiple incorrect logon attempts, and so on. One solution is to store the user ID and password on the device. Although it improves the user experience, it can introduce even more attack vectors. The OAuth standard offers an alternative, secure solution that removes most attack vectors and improves the usability of the application. With OAuth, the mobile application:

- ▶ Never stores or requires entry of the primary credential on the device.
- ▶ Does not require a long password (or no password), but provides an application launch with optional PIN protection.
- ▶ Provides user or administrator-managed device revocation.

Using IBM Security Access Manager, organizations can implement a mobile OAuth solution that increases user satisfaction, manages Access Management, and increases overall security.

The OAuth 2.0 specification includes a number of flows, which define how a user is identified to the OAuth server in order to receive an OAuth authorization grant. Each flow involves different security characteristics. Once the grant has been established, the user is issued an access token and optional refresh token, which are used for authorization in place of their credentials when accessing protected resources in place of user credentials.

5.6.1 OAuth patterns

Using the OAuth pattern, no personal information about the user is ever stored on the device. The tokens stored on the device can be invalidated temporarily or permanently to prevent access from that device and to a particular application. Adding an optional personal identification number (PIN) during authentication with a refresh token can also strengthen security.

The two flows most applicable for mobile application authorization are *authorization code* and *resource owner password credentials*. Mobile app developers must implement the logic required using HTTP request and responses to use the OAuth pattern in their applications.

Ready to run Worklight sample applications for integration with IBM Security Access Manager for Mobile using OAuth are available from *IBM Security Access Manager for IBM Worklight* at:

<https://ibm.biz/isamworklight>

Authorization code

Instead of users supplying a user name and password, a defined OAuth registration process is followed, with an enrollment step completed on a separate, trusted, and secure system. In a banking scenario, a user typically logs in to their Internet banking portal on their traditional browser system, such as a notebook. Within the web application is a self-service link to mobile banking enrollment in which the user is able to enroll a device.

The IBM Security Access Manager for Mobile OAuth registration page can be integrated into existing web applications and skinned or themed as required. The enrollment is available at: https://<isamforweb>/mga/sps/oauth/oauth20/authorize?client_id=<clientid>&response_type=code

The user obtains a time-sensitive registration code, called an *authorization code*, from this system. The authorization code is valid for a single use only, is time-limited, and must be entered in the device and validated before it expires. The code entry can be manual, using the device keyboard, or it can take advantage of device integration, such as scanning a quick-response (QR) code.

Figure 5-2 on page 94 shows the authorization code flow in the mobile app screens:

1. When users start the mobile app, a registration window opens, on which they submit their registration code (the authorization code). If the validation is successful, they are authenticated by IBM Security Access Manager and then single signed-on to Worklight.
As part of the successful validation, the user is issued an access token to use for the remainder of the session and refresh token for re-authentication upon expiration of the access token. The refresh token is stored securely on the device using the encryption features of the Worklight client JSONStore.
2. During the next launch of the mobile app, users need only to enter their PIN (if set) to exchange the refresh token for a new access token and new refresh token, which overwrites the existing one on the device. The application becomes identity aware without the need to enter the user credentials.

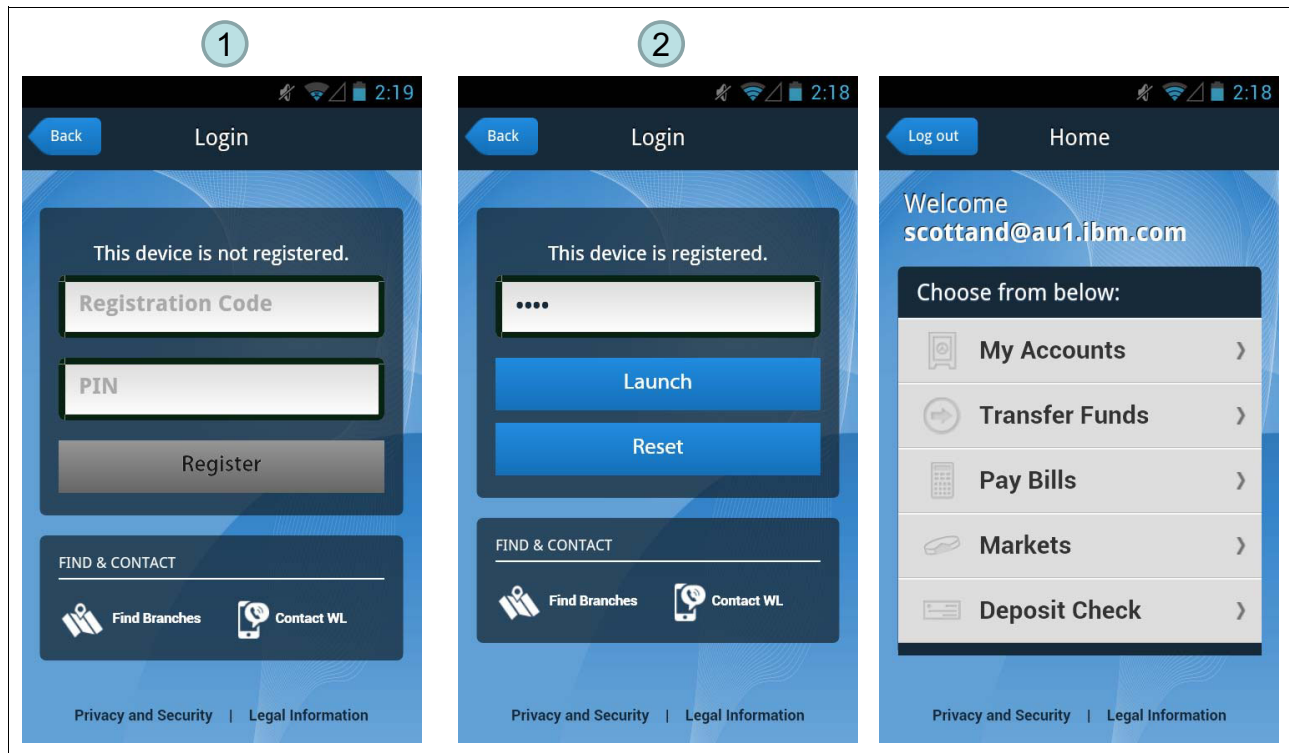


Figure 5-2 Authorization code flow mobile app windows

Resource owner password credentials

Whenever access to a separate, trusted, and secure system is not possible, or not desirable depending on the app function, the *resource owner password credentials* authorization flow method is used. When the mobile app is started, the app checks if it has been registered.

If this is the first time the application is launched, the user is prompted for their credentials (step 1 in Figure 5-3 on page 95) and optionally, on another screen using another flow, they can set a PIN for secure access in subsequent launches (step 2 in Figure 5-3 on page 95). This flow of user name and password performs the registration. Once successful, the user is issued with the access token to use for the remainder of the session and a refresh token for re-authentication upon expiration of the access token. The refresh token is stored securely on the device using the encryption features of the Worklight client JSONStore.

During the next launch of the mobile app, users need only to enter their PIN (if set) to exchange the refresh token for a new access token and new refresh token, which overwrites the existing one on the device.

The application becomes identity aware without the need to enter user credentials.

Using the resource owner password credentials requires trusting the legitimacy of the mobile app and your credentials will not be stored or exposed.

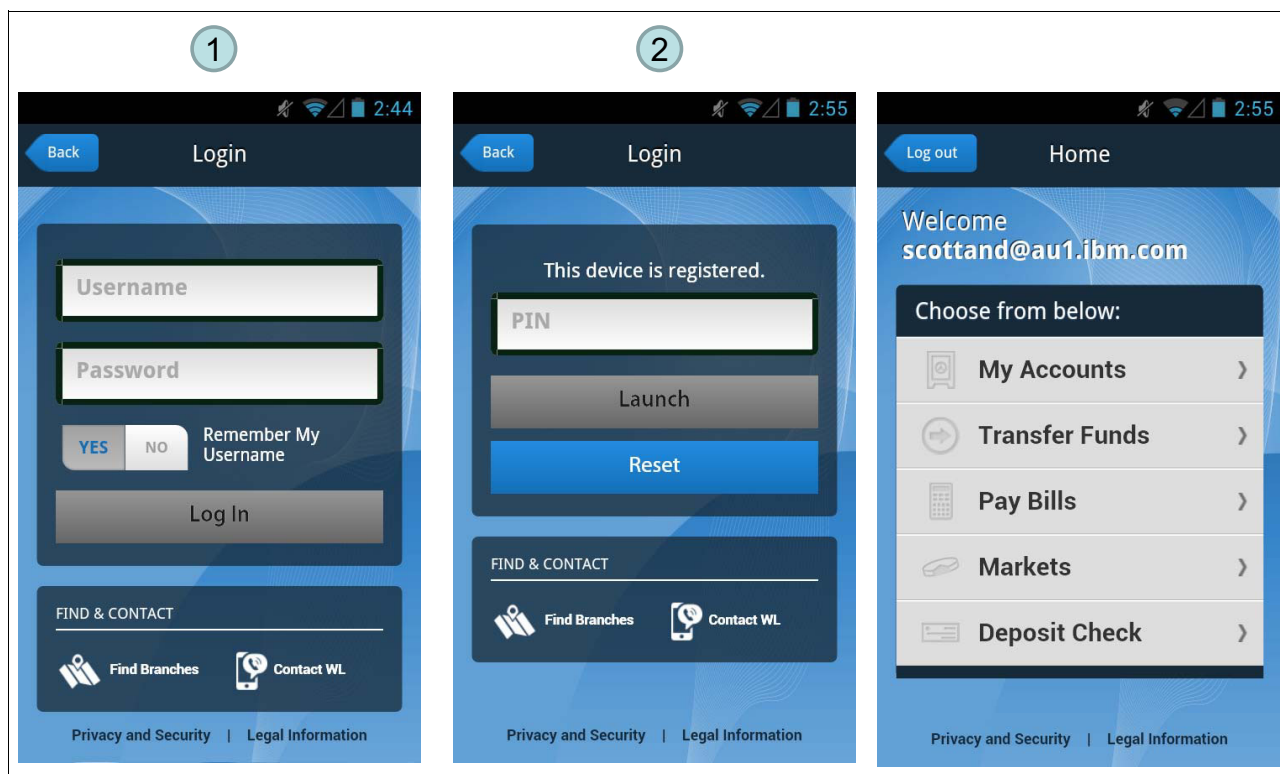


Figure 5-3 Resource owner password credentials flow mobile app windows

5.6.2 PIN protection on refresh tokens

Depending on authentication, authorization, or purely identity requirements, application developers may implement subtle differences in their patterns for enabling identity aware applications. For example:

- ▶ For an app that provides personalization per session using a third-party identity provider but does not store or retrieve any sensitive information, a long lived access token, without a refresh token may be used.
- ▶ For a mobile app where each launch of the app should start a new session but have access to profile information from a back-end server, an access token and refresh token without PIN may be used.
- ▶ For a mobile app where the user is required to authenticate each time that the app is launched, an access token and refresh token with PIN protection may be used.

A refresh token is used by the mobile app when an access token expires. The mobile app exchanges the refresh token for a new access token and a new refresh token to be used in future refresh flows. The refresh token flow can be secured with a user-defined PIN, which allows the temporary or permanent revocation of access for a particular OAuth grant.

When a mobile app does not store the access token, and the session lifetime of an access token is short lived, using a refresh token with PIN protection ensures that users are required to revalidate their identity with the PIN each time the application is loaded.

The pattern chosen for the access token lifetime and refresh token with PIN protection is a security decision for the application owner and each pattern can be implemented by using IBM Security Access Manager for Mobile.

5.7 Integrating Mobile and self-service requirements into IBM Security Access Manager for Mobile

An application programming interface (API) protection policy must be created in IBM Security Access Manager for Mobile to control the behavior of how resources are accessed. The API protection uses the OAuth 2.0 protocol. To configure the API protection, you must create a definition and a client and then attach the API protection definition to a resource protected by IBM Security Access Manager for Web.

You must complete configuration on both IBM Security Access Manager for Mobile and IBM Security Access Manager for Web to use IBM Security Access Manager as a point of contact for OAuth 2.0 decisions. This configuration must be completed before starting the OAuth configuration tasks.

Note: The objective of this section is to show the interfaces used by developers to integrate mobile apps and web self-service capabilities into an IBM Security Access Manager for Mobile use case. This section is *not* intended to outline the service configuration.

For information about how to implement OAuth service extensions to implement OAuth flow business logic within IBM Security Access Manager for Mobile, see the IBM developerWorks® article *Configure common use cases for IBM Security Access Manager for Mobile* at the following URL:


<http://www.ibm.com/developerworks/library/se-isam-mobileusecase/index.html>

Complete the instructions in *Using the isamcfg tool* in the IBM Security Access Manager for Mobile IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSELE6_8.0.0.5/com.ibm.isam.doc_8.0.0.5/config/concept/con_isamcfg.html

5.7.1 Create API protection definition

To create an API protection definition, perform the following steps in the IBM Security Access Manager for Mobile Local Management Interface:

1. Select **Secure Mobile Settings** → **API Protection**.
2. Click **Create Definition** .
3. Provide an appropriate name in the Name field, for example **Worklight Bank**.
4. Click **Grant Types** and select at least one grant type, for example **Authorization code** and **Resource owner username password**.
5. For extra security, select **Enforce single-use authorization grant**.
6. For convenience of Mobile use cases, specify four characters for **Authorization code length**.
7. For troubleshooting or latency errors, select **Enable multiple refresh tokens for fault tolerance**.
8. Select **Enable PIN policy**.
9. For convenience of Mobile use cases, select the PIN length as 4.
10. For convenience of Mobile use cases, select Token character set as **Numeric**, **Alphanumeric uppercase**, or **Alphabetic uppercase**. These choices help eliminate

confusion between uppercase i and lowercase l using certain fonts. Alternatively, modify the character set and remove any letter formats that can be confused.

11. Optionally, set the **Trusted Clients and Consent** prompt.

12. Click **Save**.

13. Deploy the pending changes.

Figure 5-4 on page 98 shows the IBM Security Access Manager for Mobile API protection definition configuration.

IBM Security Access Manager

isam800

Home

Appliance Dashboard

Monitor

Analysis and Diagnostics

Secure

Web Settings

Secure

Mobile Settings

Manage

System Settings

API Protection

Definitions

Resources

Clients

Advanced

Save

Cancel

Name:

Worklight Bank

Description:

Grant Types

☒ Authorization code
 ☒ Resource owner username password
 ☐ Client credentials
 ☐ Implicit

Token Management

Access token lifetime (seconds):

3,600

Access token length:

20

☐ Enforce single-use authorization grant

Authorization code lifetime (seconds):

300

Authorization code length:

4

☒ Issue refresh token

Maximum authorization grant lifetime (seconds):

604,800

Refresh token length:

40

☒ Enforce single access token per authorization grant

☐ Enable multiple refresh tokens for fault tolerance

☒ Enable PIN policy

PIN length:

4

Token character set:

0123456789

Select character set


Trusted Clients and Consent

Figure 5-4 IBM Security Access Manager for Mobile API protection definition

5.7.2 Create OAuth client

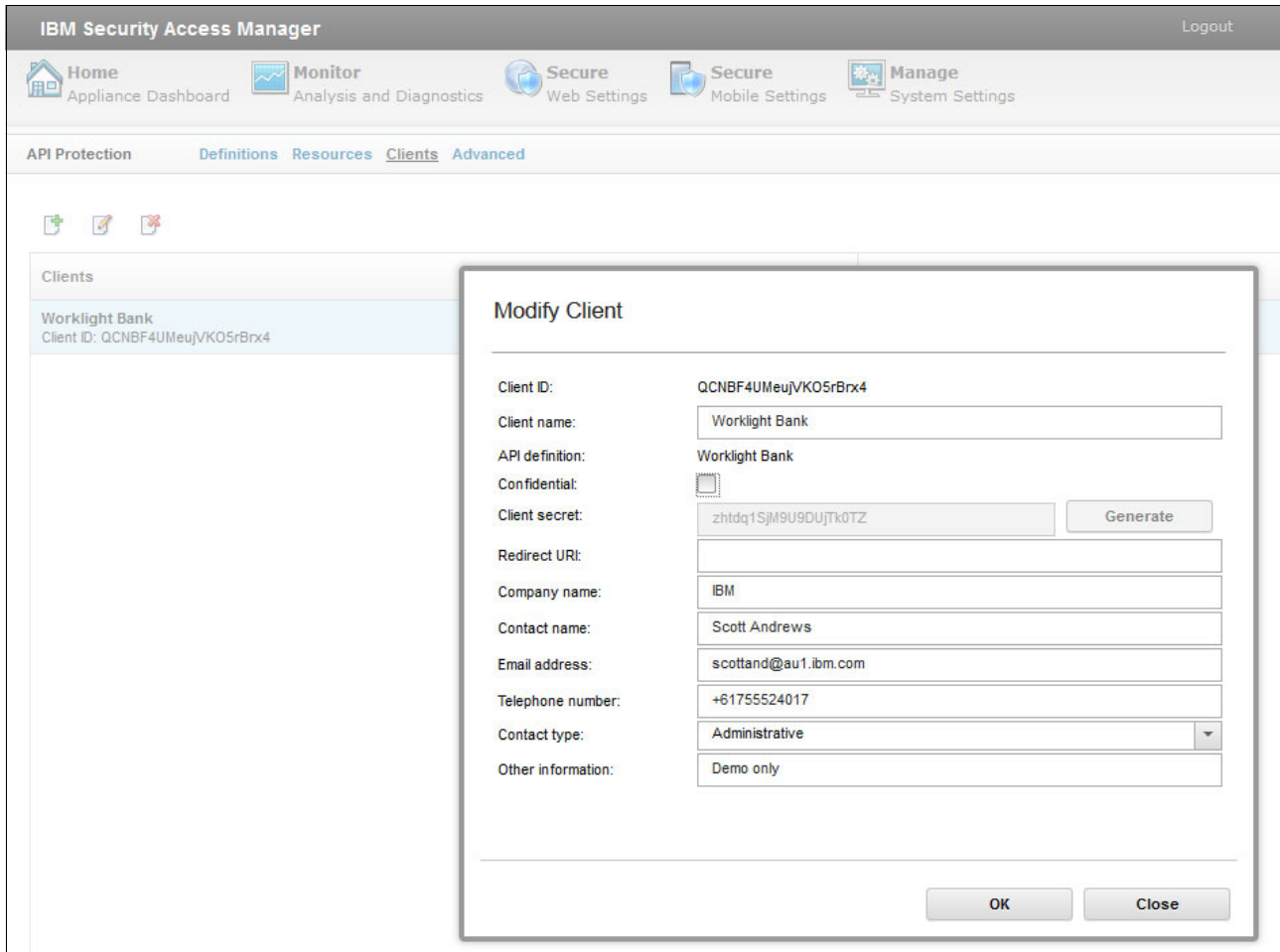
To create an API protection client, perform the following steps in the IBM Security Access Manager for Mobile Local Management Interface:

1. Select **Secure Mobile Settings** → **API Protection**.
2. Click **Clients**.

3. Click **New Client.** .
4. Provide an appropriate Name, for example **Worklight Bank.**
5. Select the API protection definition previously created. For example, **Worklight Bank.**
6. Clear **Confidential.**
7. Provide administrative details of the OAuth client.
8. Click **OK.**
9. Deploy the pending changes.

Note: The Client ID generated after creating the API protection client is the ID that must be coded into any mobile app using this OAuth client. For ease of identification, it is recommended that the client ID is overwritten with a meaningful name during the API protection client creation in IBM Security Access Manager for Mobile.

Figure 5-5 shows the IBM Security Access Manager for Mobile API protection client configuration.



The screenshot displays the IBM Security Access Manager for Mobile interface. The top navigation bar includes links for Home, Monitor, Secure, and Manage. The 'Clients' tab is selected under the 'API Protection' section. A 'Modify Client' dialog box is open, showing the configuration for a client named 'Worklight Bank'. The client ID is 'QCNBF4UMeujVK05rBrx4'. The 'Confidential' checkbox is unchecked. The 'Client secret' field contains 'zhldq1SjM9U9DUjTk0TZ' and a 'Generate' button is next to it. The 'Redirect URI' field is empty. The 'Company name' is 'IBM', 'Contact name' is 'Scott Andrews', 'Email address' is 'scottand@au1.ibm.com', 'Telephone number' is '+61755524017', 'Contact type' is 'Administrative', and 'Other information' is 'Demo only'. The dialog has 'OK' and 'Close' buttons at the bottom right.

Client ID:	QCNBF4UMeujVK05rBrx4
Client name:	Worklight Bank
API definition:	Worklight Bank
Confidential:	<input type="checkbox"/>
Client secret:	zhldq1SjM9U9DUjTk0TZ Generate
Redirect URI:	
Company name:	IBM
Contact name:	Scott Andrews
Email address:	scottand@au1.ibm.com
Telephone number:	+61755524017
Contact type:	Administrative
Other information:	Demo only

Figure 5-5 IBM Security Access Manager for Mobile API protection client

5.7.3 Attach API protection policy to endpoints

The API protection policy must be attached to any resource protected by IBM Security Access Manager for Web that will be accessed by mobile apps using the OAuth client. Perform the following steps in the IBM Security Access Manager for Mobile Local Management Interface:


1. Select **Secure Mobile Settings** → **API Protection**.
2. Click **Resources**.
3. Select the IBM Security Access Manager for Web instance, for example **wldemo-default**.
4. Select the resource to attach the policy to, or, if necessary, add the resource. For example, **/authenticate**.
5. Click the attach icon  **Attach**.
6. In the **Attach Policies** window:
 - a. Select the **API Protection** option.
 - b. Select the API protection definition created previously.
 - c. Click **OK**.
7. Back in the Resources window, click **Publish**.

Figure 5-6 on page 101 shows the IBM Security Access Manager for Mobile API protection policy attachment.

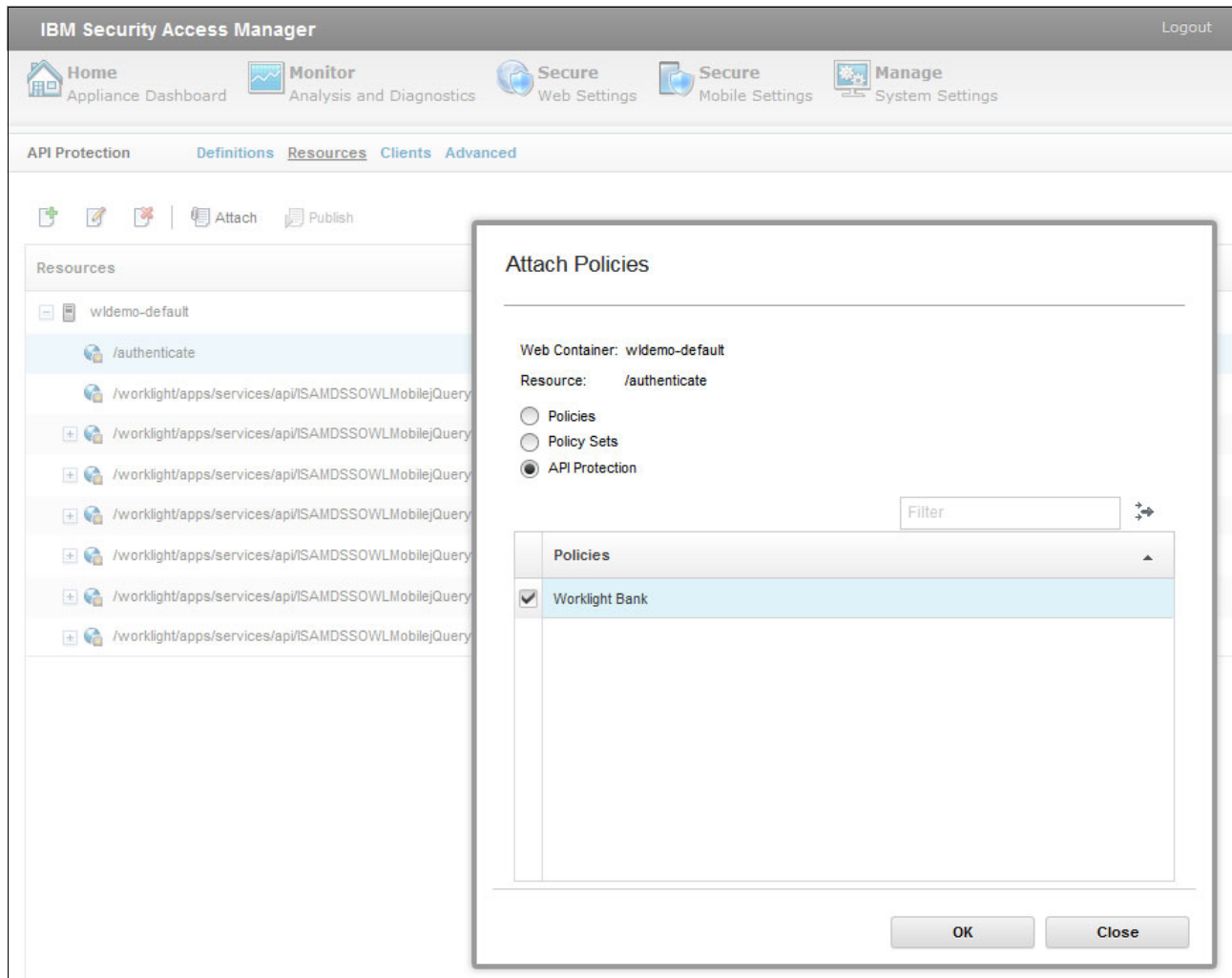


Figure 5-6 IBM Security Access Manager for Mobile API protection policy attachment

5.8 Example OAuth flow with PIN validation using IBM Security Access Manager for Mobile

This section discusses an example of a mobile application that uses the resource owner password credentials OAuth flow. For added security, a one-time password (OTP) is required to verify the account owner and to set the PIN for the refresh token.

The combination of OAuth and OTP provides an identity aware application and uses strong authentication.

5.8.1 Application registration

Mobile app registration processes provide an opportunity for app designers to establish a binding between an app and the device the app is being installed in. This binding can be used to mitigate threats of rogue apps being used to impersonate users installed on other devices. App registration can include device fingerprinting processes. Many financial institutions are using traditional user name and password registration processes to drive to a two-factor

authentication solution (a PIN and stored device token) for B2C mobile app access. This process should be augmented with reliable device fingerprinting technologies (such as IBM Trusteer), so that subsequent access can consider this context as part of transactional authorization processes.

The following sections show some high-level flows for registration.

Registration using existing credentials (user name and password)

This use case outlines the requirement for exchanging an existing user name and password combination for a token (stored securely on the device) and a PIN code. This use case is becoming more common in finance industry mobile app use cases and has been mandated by clients across the world.

Figure 5-7 shows a set of windows that takes a user through a step-by-step scenario for registering a PIN.

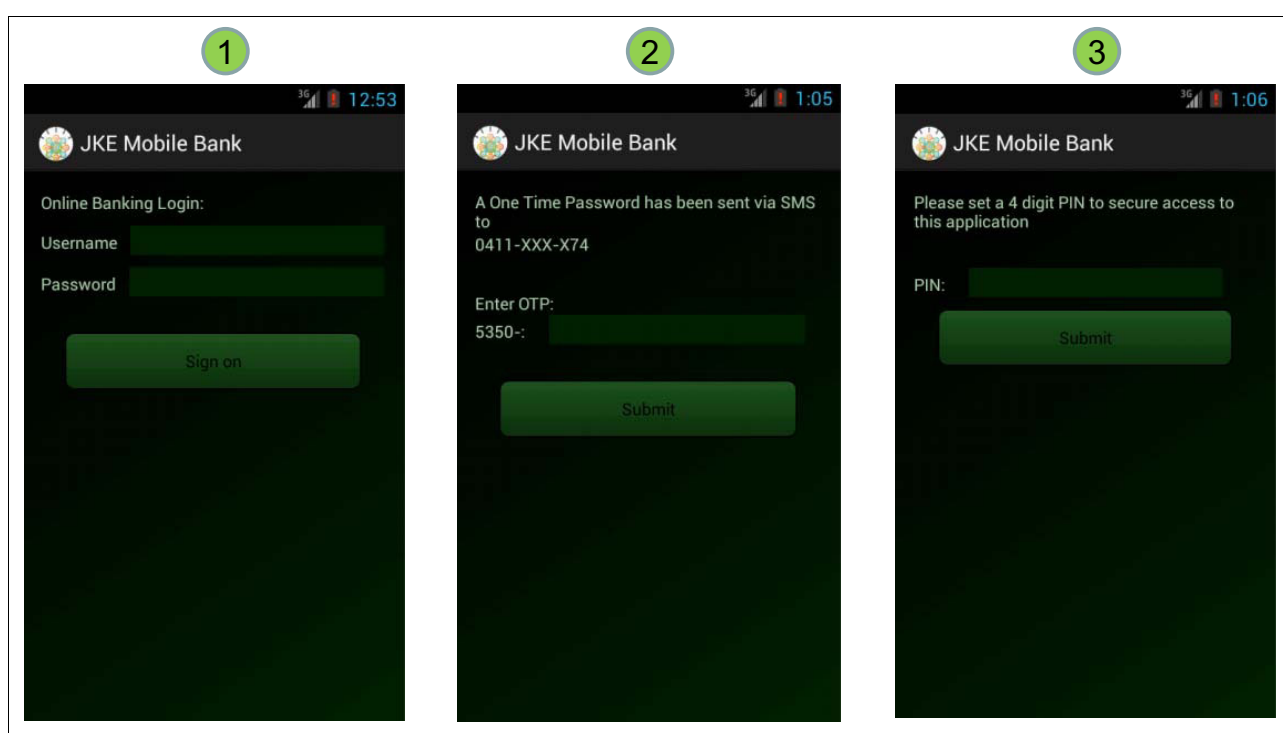


Figure 5-7 Application registration process

The registration policy in this case mandates the following set of operations:

1. The user must enter existing credentials (user name and password).
2. The server must send an out of band SMS to the user with a one-time password. The mobile phone number must be already present in the server. The user must enter the one-time password. This process proves server authenticity as only the legitimate server knows the user's mobile phone number.
3. The user sets the PIN number that they have received for future access. Optionally, the user's device can be fingerprinted, with the fingerprint information persisted along with the registered PIN. This approach provides the ability for not only the PIN to be checked, but also that the fingerprint matches the fingerprint persisted during registration.

Note that this set of operations are typically one time processes for each device that the user owns. Some customers insist on a new PIN on each device, others are content with one PIN across all devices. This is a policy decision only.

In a native app implementation, the developer does not want to interact with the server using traditional web processes but rather have security conform to the API model. It is critical that security controls can be applied to the use of APIs.

The diagram in Figure 5-8 shows the on-the-wire formatted requests required for developing a solution with IBM Security Access Manager for Mobile. For simplicity, the registration process shown does not include device fingerprinting processes. However, IBM Security Access Manager for Mobile provides the ability to register a fingerprint. For more information, see *Attribute collection service* in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSZSXU_6.2.2.7/com.ibm.tivoli.fim.doc_6227/rbaAdminACS.html

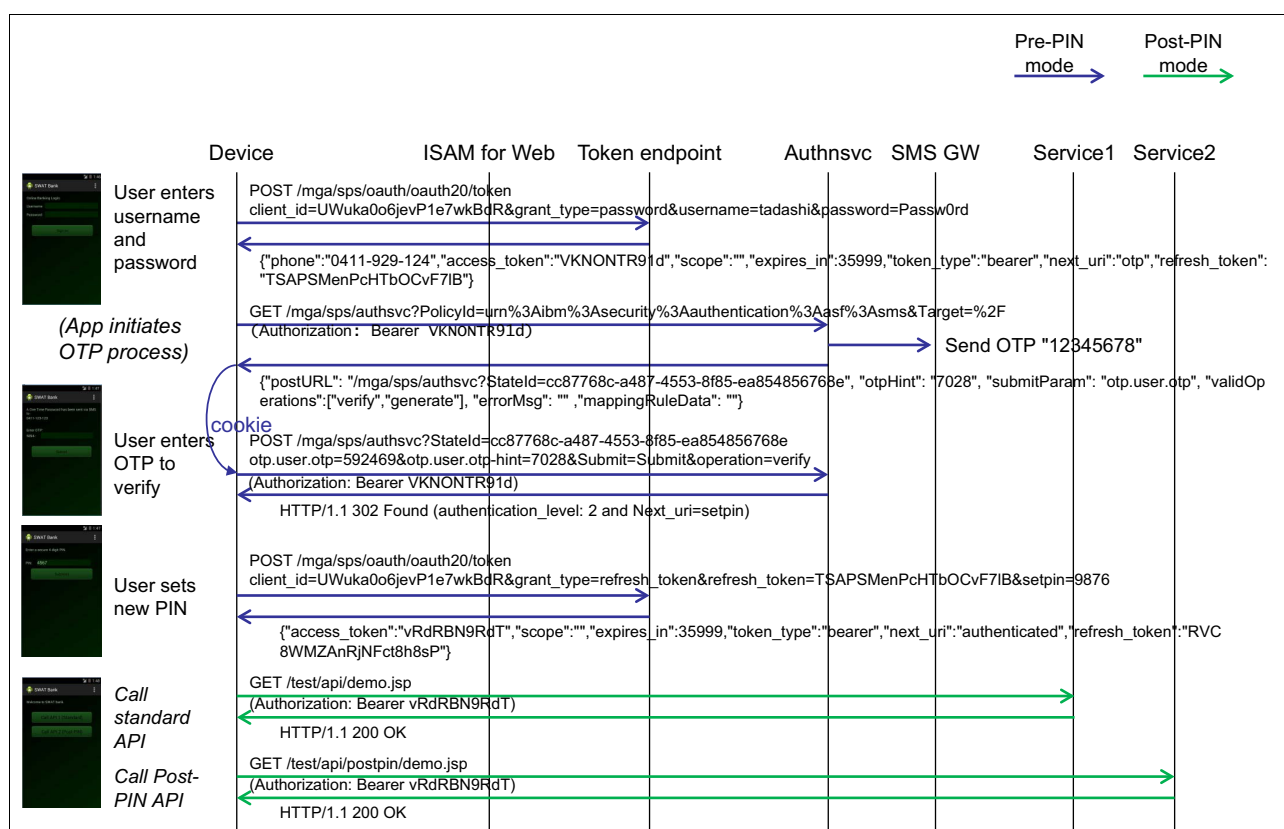


Figure 5-8 Accessing business APIs using access token: Authentication sequence 1

The sequence diagram refers to the following components:

- ▶ **Pre-PIN mode:** Refers to the operations that occur before the users set their PIN number.
- ▶ **Post-PIN mode:** Refers to the operations after the app has been registered.
- ▶ **Device:** The device in which the app is being installed.
- ▶ **IBM Security Access Manager for Web:** The policy enforcement point for API access.
- ▶ **Token endpoint:** The IBM Security Access Manager for Mobile endpoint for OAuth token operations.
- ▶ **AuthSvc:** The IBM Security Access Manager for Mobile endpoint for context-based access control.

- ▶ SMS GW: A Short Message Service (SMS) gateway.
- ▶ Service1 and Service 2: Business API service endpoints.

5.8.2 Two factor authentication step up

In the example described in “Registration using existing credentials (user name and password)” on page 102, a second factor one-time password sequence was initiated as part of the app registration through the app itself, specifically programmed as part of the registration workflow.

However, in accessing business APIs, the enforcement point itself is responsible for performing authorization (and therefore client obligations) of a client in accessing an API. This authorization decision may trigger additional authentication challenges to occur. IBM Security Access Manager for Mobile provides an authentication framework that can be used to implement stronger authentication mechanisms as part of such challenges. In the example described in “Registration using existing credentials (user name and password)” on page 102, IBM Security Access Manager for Mobile is used to drive a second factor one-time password flow through an SMS message to the authenticated mobile device of the user.

5.8.3 Business API using access token/refresh token

Having completed registration, the OAuth standard allows for a long lived, high entropy token (refresh token) to be used for retrieving an OAuth access token. It is common for financial industry organizations to want to provide low risk operations to users, without the user providing any PIN number. In Figure 5-9 on page 105, this approach is referred to as Pre-PIN mode, and it is shown with the blue lines. If an app is in possession of such an access token, that app is only permitted to access this low risk transaction endpoint (such as showing the account balance of a single account).

Post-PIN mode allows a user to access a broader set of APIs, and mandates that the refresh token is presented with the user’s PIN in order to receive an access token for calling the business APIs. Of course, an app in possession of such a token is authorized to a broader set of APIs via IBM Security Access Manager for Mobile policy.

Note: The flows that are shown in Figure 5-8 on page 103 and Figure 5-9 on page 105 are native flows. These flows can be embedded into a Worklight app as part of a native process.

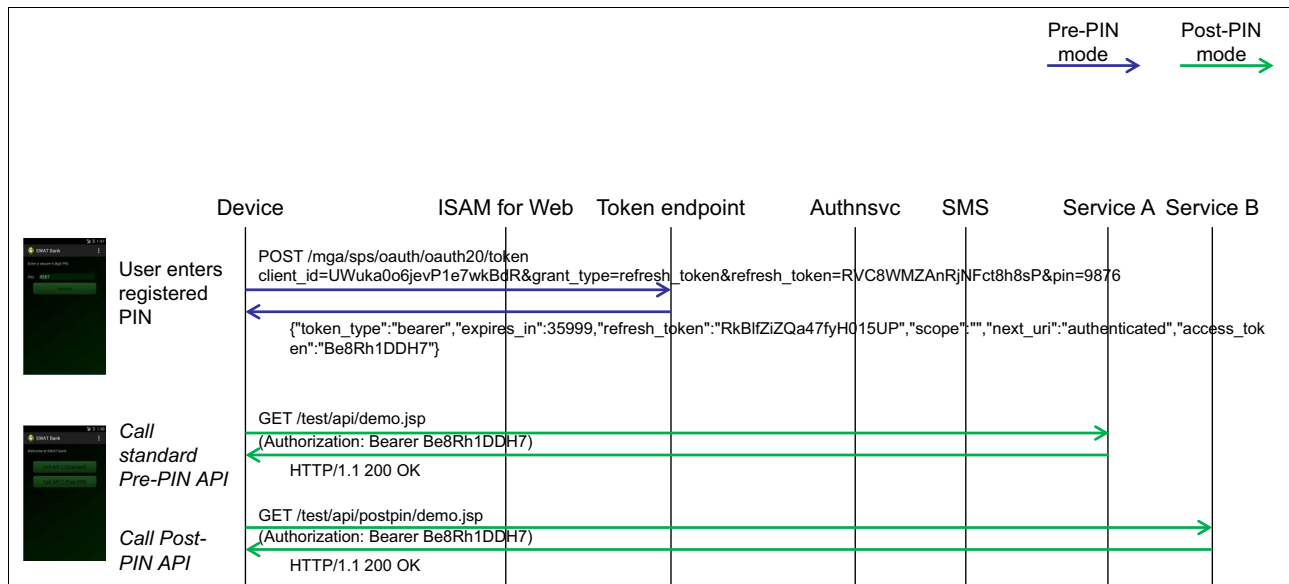


Figure 5-9 Accessing business APIs using access token: Authentication sequence 2

5.8.4 Administrator device management

Following user enrollment and device registration, IBM Security Access Manager for Mobile administrative users can retrieve a list of users with their registered devices.

Perform the following steps in the IBM Security Access Manager for Mobile Local Management Interface:

1. Select **Secure Mobile Settings** → **Device**.
2. Enter a user ID or wildcard pattern to search for. For all users, enter *****.
3. Click **Search**.
4. From the User Results panel, select the user ID to view the device registrations for.
5. In the Registered Device Fingerprints panel, select the Device name to view.
6. Optionally, click the **Device Fingerprint Attributes** icon to view the attribute data collected about the registered device.

Note: The attributes collected depend on those defined in the active risk profile at the time of registration.

Figure 5-10 on page 106 shows the IBM Security Access Manager for Mobile device registration search.

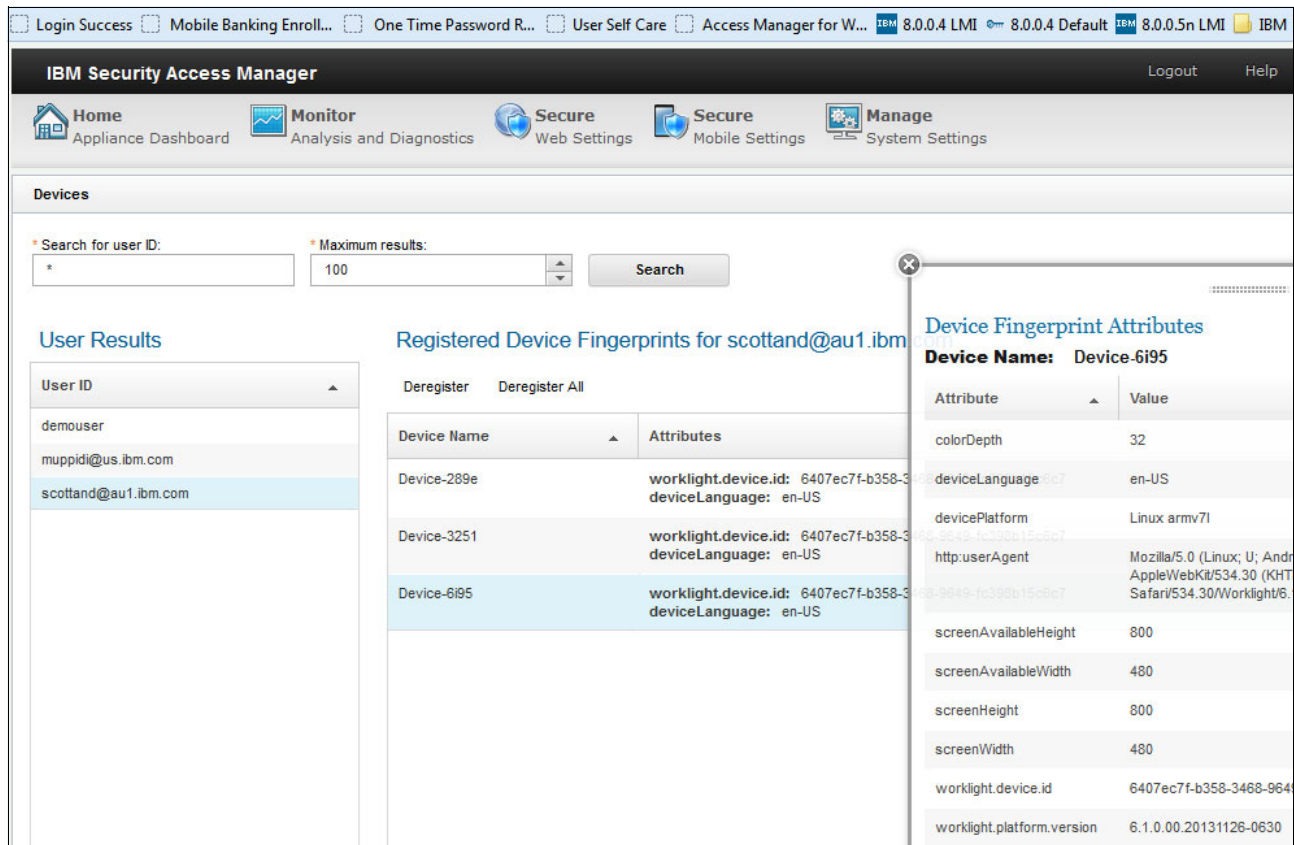


Figure 5-10 IBM Security Access Manager for Mobile device registration search

5.8.5 User device management

IBM Security Access Manager for Mobile provides a user self-service capability that can be integrated into existing help desk applications. This UI can be customized by changing the skin and theme as required. This capability allows users to manage their device registrations and OAuth grants.

The self-service feature is available at the following link:

https://<isamforweb>/mga/sps/mga/user/mgmt/html/device/device_selection.html

This link is typically embedding within an existing web application such as an Internet banking portal.

Figure 5-11 on page 107 shows the IBM Security Access Manager for Mobile user self-service device management UI.

IBM			
Device Selection			
Name	Enabled	Last Activity	
Device-3251	<input checked="" type="checkbox"/>	2014-09-22T07:37:39Z	Remove
Device-6i95	<input checked="" type="checkbox"/>	2014-08-14T23:59:15Z	Remove
Device-289e	<input checked="" type="checkbox"/>	2014-10-20T01:08:44Z	Remove
Authorization Grant Selection			
Id	Enabled	Client Id	
uuid2b1723f5-0149-17d9-a69e-da2c97b166a0	<input checked="" type="checkbox"/>	QCNBF4UMeujVKO5rBrx4	Remove
uuid2b18aeef-0149-15ff-ac83-da2c97b166a0	<input checked="" type="checkbox"/>	QCNBF4UMeujVKO5rBrx4	Remove

Figure 5-11 IBM Security Access Manager for Mobile self-service device management

Manage device access

After registering a device, it may be necessary to temporarily or permanently revoke access for a device, for example, if the device is misplaced, lent to, or being used by a family member, or it is lost or stolen.

Rather than calling a help desk, users can manage their own devices with the self-service capability.

Perform the following steps to manage the device access:

1. Open the self-service feature, likely from within a trusted web application, such as an Internet banking portal.
2. Locate the device name that was specified during registration.
3. The device can be either temporarily disabled using the Enabled check box or permanently disabled using the Remove link, requiring the device to be reregistered if access is required later.
4. Click the device to view the Device Attributes collected during registration (see Figure 5-12 on page 108).
5. Optionally, rename the device for easier identification. For example, My iPhone.
6. The device can also be disabled or removed from the Device Attributes view.

Device Attributes

Attributes for Device-6i95

Name	Value
http.userAgent	Mozilla/5.0 (Linux; U; Android 4.1.2; en-us; Nexus S Build/JZO54K) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30/Worklight /6.1.0.00.20131126-0630
screenHeight	800
screenWidth	480
screenAvailableHeight	800
screenAvailableWidth	480
colorDepth	32
deviceLanguage	en-US
devicePlatform	Linux armv7l
worklight.device.id	6407ec7f-b358-3468-9649-fc398b15c6c7
worklight.platform.version	6.1.0.00.20131126-0630

☒ Enabled

Rename Device

New device name:

Figure 5-12 IBM Security Access Manager for Mobile user self-service device attributes

Manage OAuth grants

Each mobile app typically uses its own OAuth grant and IBM Security Access Manager for Mobile policy definition. Using this pattern, users can self-manage the temporary or permanent revocation of OAuth grants on a per application basis.

For ease of identification, it is recommended to overwrite the client ID with a meaningful name during the API protection client creation in IBM Security Access Manager for Mobile. This is the same ID that is coded into the application and can be displayed to the user in a help screen of the mobile app to assist with the self-management capabilities.

Rather than calling a help desk, users can manage their own application registration in the self-service feature by performing the following steps:

1. Open the self-service feature, likely from within a trusted web application, such as an Internet banking portal.
2. Locate the client ID that is displayed in the help screen of the mobile app (if available).
3. The mobile app OAuth grant can be either temporarily disabled using the Enabled check box or permanently disabled using the Remove link, requiring the user to reregister if access timeout is required later.
4. Click the ID corresponding to the client ID to view the Authorization Grant used by the mobile app to access the protected resource using OAuth.
5. The grant can also be disabled or removed from the Authorization Grant view (see Figure 5-13 on page 109).

Authorization Grant

Tokens for uuid2b1723f5-0149-17d9-a69e-da2c97b166a0

Token String	Type	SubType	Date Created	Lifetime (seconds)	Last Activity	Scope
QaGS4JgkRFuy3Uhg2T9G	access_token	bearer	2014-10-20T01:06:31Z	3600	2014-10-20T01:06:31Z	
65K8s4fCF02LISn4Ae1Z6kTbY8CQxfWtyf4VxSYI	authorization_grant	refresh_token	2014-10-20T01:06:31Z	604785	2014-10-20T01:06:31Z	

Attributes for uuid2b1723f5-0149-17d9-a69e-da2c97b166a0

Name	Value
<input checked="" type="checkbox"/> Enabled	

[Remove Authorization Grant](#)

Figure 5-13 IBM Security Access Manager for Mobile user self-service authorization grant

5.9 Conclusion

IBM Security Access Manager for Web and IBM Security Access Manager for Mobile provide powerful, flexible, and scalable authentication and authorization scenarios for traditional web applications as well as for mobile apps using standards-based technologies.

IBM Worklight app developers can include IBM Security Access Manager challenge handlers into their existing apps to add identity awareness (OAuth), strong authentication (one-time password), and context-based access on Worklight adapter transactions to strengthen the security position of the mobile apps and protect the corporate network with intelligent fraud detection and prevention.



Scenario 1: Getting started

This chapter describes the solution design and implementation for a scenario that represents a cable TV installation service company. The company in this scenario receives installation requests from external customers and assigns those requests to field technicians. The field technicians use a mobile application to manage the work orders.

In the context of mobilized business processes, this scenario illustrates how to use IBM Worklight, IBM Business Process Manager, and IBM Bluemix to develop the solution for a fictitious company.

This chapter describes the basic Worklight and IBM BPM features organizations typically use when they start the journey to mobile-enable processes. This chapter includes implementation details for the features and mobile app described in the scenario.

The Worklight features described in this scenario are:

- ▶ Adapter-based authentication
- ▶ Geolocation
- ▶ Offline storage
- ▶ Push notifications
- ▶ Worklight adapters for IBM Business Process Manager (BPM) and data integration

The IBM BPM features shown in this scenario are:

- ▶ Business process definition
- ▶ Java integration service
- ▶ IBM Bluemix integration

6.1 Scenario 1 requirements and use case description

The scenario 1 use case is centered around how to mobilize business processes managed by the IBM BPM server. The requirements for this use case are:

- ▶ The IBM BPM server must be able to receive a new customer service request and manage the process until the work is completed.
- ▶ The field technicians must be notified when new work orders are available.
- ▶ The field technicians must be able to securely update the work order status when they are working offline and network connectivity is not available.
- ▶ The cable TV service company must be able to track the location of the technicians in the field.

6.1.1 Customer use case description

The following steps describe the customer interaction flow:

1. The customer requests a new cable TV installation over the phone.
2. The call center operator starts a New Order Installation business process (NOI-BP) from the Process Portal and enters the customer details.
3. The NOI-BP creates a full customer profile in the customer master data management system (this step is simulated by calling a Bluemix service).
4. The NOI-BP checks the scheduler for available field technicians (this step is simulated by calling a Bluemix service).
5. The NOI-BP sends a notification message to the field service team using Worklight push notification service triggering an installation work order business process (WO-BP), and assigns the WO-BP to a field technician team.
6. The WO-BP sends an SMS notification to the customer when the work order is complete. This step is not implemented in this scenario; it is added to the flow for completeness.
7. To finish the work order, the field technician uses the Field Service mobile application to update the status variable `tw.local.workOrderReport.status` in a user task in the WO-BP.
8. The WO-BP completes and returns back to the NOI-BP that called it. If the status is set to `Complete`, the NOI-BP updates the fulfillment system and notifies the customer via an SMS message (both simulations which are not implemented in this example) and then the business process ends.
9. If the status is set to anything other than `Complete`, a user task called `Follow-up with customer` begins. When this user task is complete, the business process ends.

6.1.2 Field technician use case

The following steps describe the field technician interaction flow:

1. The members of the field technician team receive a push notification regarding new work orders available.
2. The members of the field technician team retrieve the list of available work orders using the field service mobile application.

3. A field technician selects and accepts the work order using the field service mobile application.
4. The field technician updates the status of the work order while the installation is performed using the field service mobile application.

6.1.3 Overall architecture

The solution architecture consists of the following components:

- ▶ A Worklight hybrid mobile application that enables the field technician to manage work orders.
- ▶ The Worklight Server to support the mobile application including the integration with the IBM BPM server.
- ▶ The IBM BPM server to capture the customer work order and manage the overall process.
- ▶ IBM Bluemix as the platform to implement various supporting processes including fulfillment, order scheduler, and customer master data management (MDM) functionality.

Figure 6-1 shows a high-level architecture overview diagram for scenario 1. Starting from the left, the diagram shows a field technician using the Field Service app which interacts with the Worklight Server. The Worklight Server has adapters that call both the IBM BPM REST API and the location service exposed by Bluemix. The Worklight Server receives notification requests from IBM BPM and submits those requests to the push notification vendors (for example, Google and Apple). The IBM BPM server also calls Customer MDM, Scheduler, and Fulfillment REST services in Bluemix.

This scenario is designed to use the architectural pattern described in 4.4.4, “Pattern 4: IBM BPM REST API” on page 77, also known as the *Headless BPM* pattern. This pattern does not leverage any user interface (UI) elements (coach forms) in the IBM BPM workflow. Instead, the UI components are externalized (in this case, in the Worklight application). The Worklight application externally starts an IBM BPM workflow and manages it until completion.

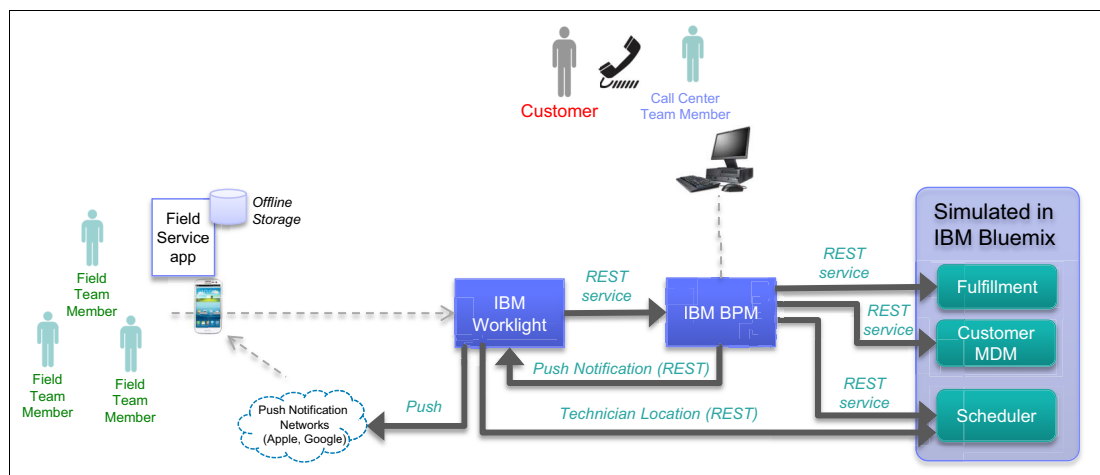


Figure 6-1 Scenario 1 architecture overview diagram

Figure 6-2 on page 114 shows a sequence diagram that has two sample flows: the order status update in IBM BPM, and the user location update in Bluemix.

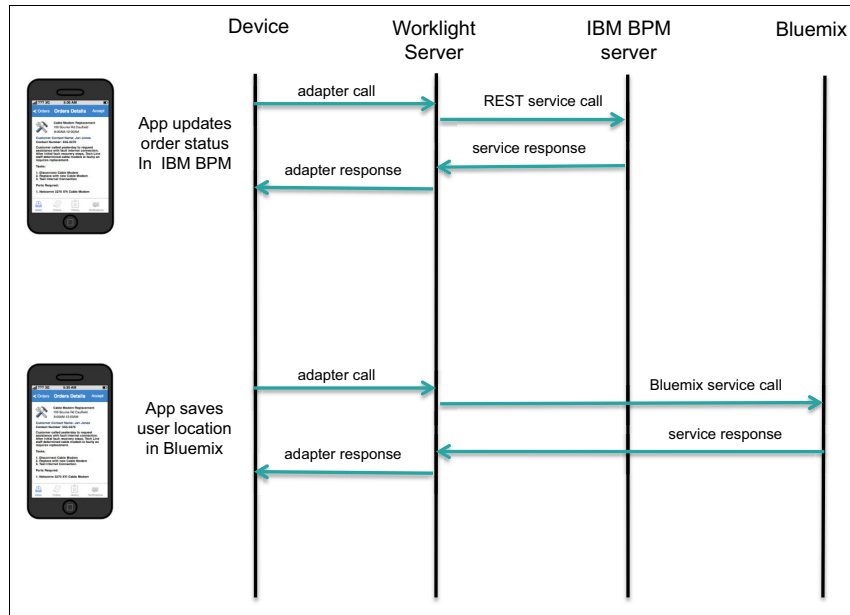


Figure 6-2 Scenario 1 sequence diagram

6.1.4 Additional considerations

In scenario 1, the integration between IBM Worklight and IBM BPM is achieved using IBM BPM REST API. There is an alternate integration approach using Java Messaging Service (JMS) technology.

Integrating Worklight and IBM BPM with JMS

Section 6.1.3, “Overall architecture” on page 113 explains that in scenario 1 IBM BPM REST APIs are used for Worklight and IBM BPM integration. Although REST APIs offer an easy and consistent integration approach, some scenarios are more suitable for an asynchronous integration approach. For example, some use cases may require assured delivery. Using JMS in such circumstances can ensure reliable delivery. Long running processes are also good candidates for JMS-based integration.

JMS adapter support in Worklight

The JMS adapter is one of the default adapters provided by IBM Worklight. It can be used to send and receive messages from a JMS-enabled messaging provider.

The adapter supports four basic APIs:

- ▶ `WL.Server.readSingleJMSMessage`: reads the next message from the queue.
- ▶ `WL.Server.readAllJMSMessages`: reads all the messages from the queue.
- ▶ `WL.Server.writeMessage`: writes a message to the queue.
- ▶ `WL.Server.requestReplyJMSMessage`: writes a message to the queue and then waits for a response on a different specified queue.

The JMS adapter can be configured to work with an external JMS provider by supplying the following information:

- ▶ `jmsConnection`
 - `connectionFactory`: name of `JMSConnectionFactory` in JNDI
 - User and password

- ▶ `namingConnection`
 - `url`: the JMS provider URL
 - `initialContextFactory`: JNDI provider initial context factory class name
 - User and password

JMS support in IBM BPM

IBM BPM processes can be integrated with JMS messages in two distinct ways:

- ▶ **Integration via SIBus (service integration bus):** To integrate with IBM BPM SIBus, Worklight can be hosted on the WebSphere Application Server Liberty Profile (Base or ND). The Worklight JMS adapter makes a JNDI call to a connection factory and the queue defined on the Liberty server. The Liberty server maps its connection to the SIBus connection factory and queue, so that when Worklight posts a message using the JMS adapter, it is sent to the BPM SIBus infrastructure.
- ▶ **Integration using Event Manager:** The Event Manager in the Process Server responds to external events. The Event Manager queues incoming messages and triggers the appropriate Undercover Agent (UCA). The UCA bridges external systems and business process definitions (BPDs) by initiating either a Start Message Event or an Intermediate Message Event that is contained within a BPD. The Event Manager listens on a predefined queue and expects a predefined XML structure.

6.2 Field service mobile application

In this scenario, the field technician uses a work order management mobile application. This app is a Worklight hybrid mobile application that leverages HTML5, CSS, and JavaScript. In a mobile hybrid application, these three web technologies work together as follows:

- ▶ HTML5 defines the content for every page in the mobile application.
- ▶ CSS dictates the look and feel (in other words, how to render this content).
- ▶ JavaScript provides the complete logic for user interaction and page navigation.

The mobile app runs on the device operating system inside a native shell. Therefore, enterprise developers can leverage popular web skills to develop full-featured Worklight mobile applications.

Assuming that an enterprise wants to develop a hybrid application to target Android, iOS, and Windows smartphone devices, the IBM Worklight Platform takes full advantage of this HTML5-CSS-JavaScript synergy. Worklight allows mobile developers to write the code that will be common to all platforms in a single place and then write the few platform-specific aspects separately.

Figure 6-3 on page 116 shows the different components of the Field Service hybrid mobile application.



Figure 6-3 Hybrid application components

The field service mobile application consists of the following pages:

- Login page: the field technician enters the login credentials in the UI shown in Figure 6-4.

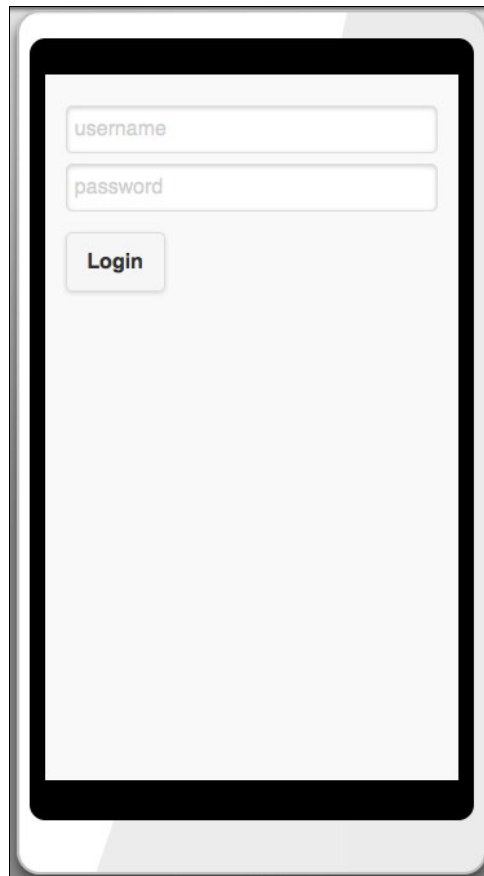


Figure 6-4 Field Service app: Login page

- **Assigned Orders:** the field technician can see the work orders available to the team on the UI shown in Figure 6-5. For implementation details, see “Assigned Orders page” on page 137.

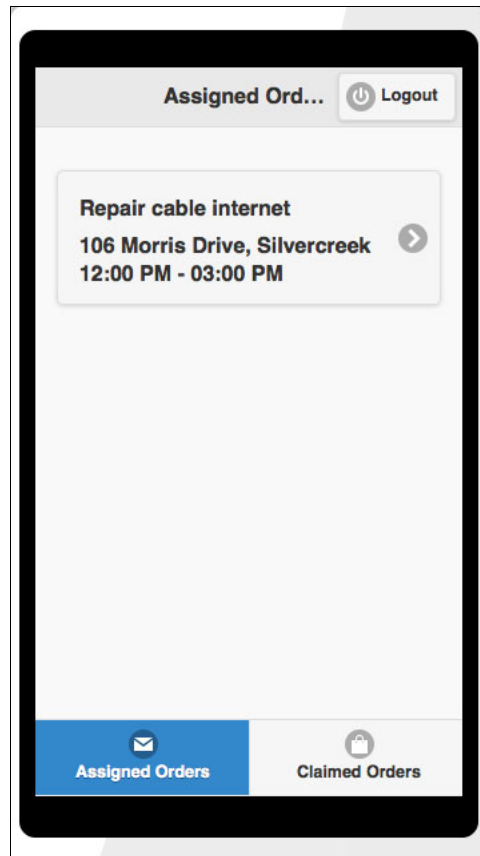


Figure 6-5 Field Service app: Assigned Orders page

- **Assigned Order Details:** the details of the available work order are displayed in the UI shown in Figure 6-6 on page 118. For implementation details, see “Assigned Order Details page” on page 138.

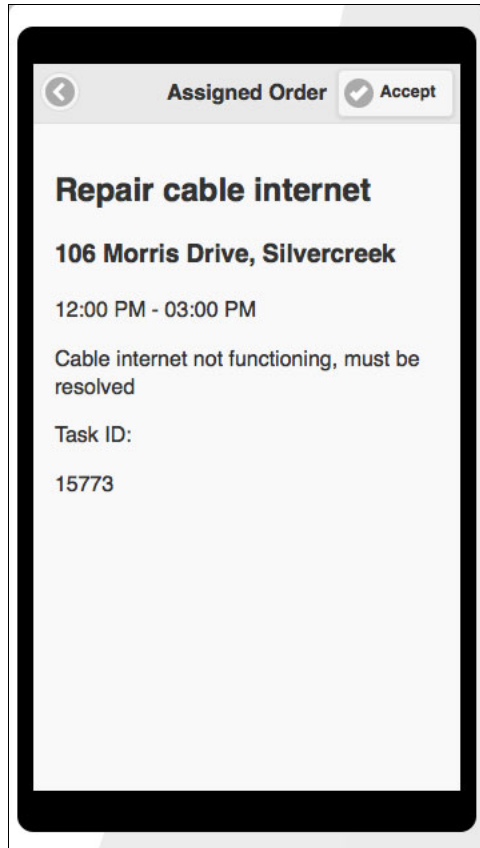


Figure 6-6 Field Service app: Assigned Order Details page

- **Order Accepted Confirmation:** the field technician accepts the selected order and receives a confirmation message, as shown in Figure 6-7 on page 119.

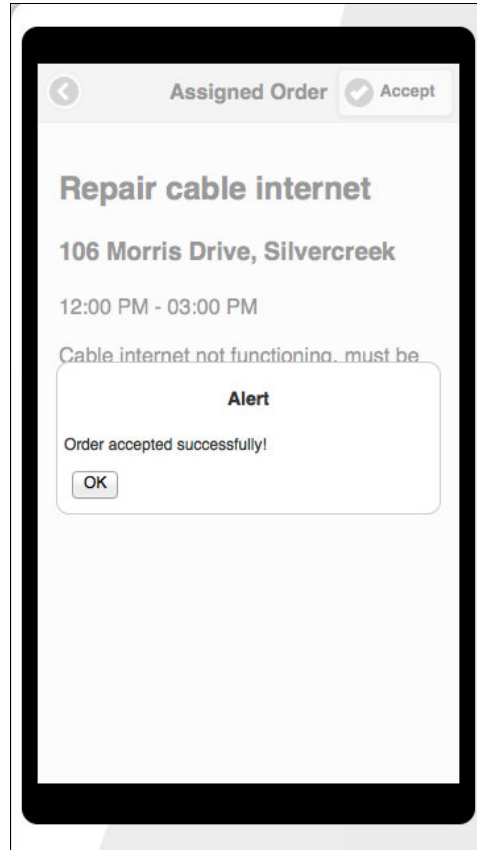


Figure 6-7 Field Service app: Order Accepted Confirmation page

- Claimed Orders: the list of orders accepted by the field technician is displayed as shown in Figure 6-8 on page 120. For implementation details, see “Claimed Orders page” on page 139.

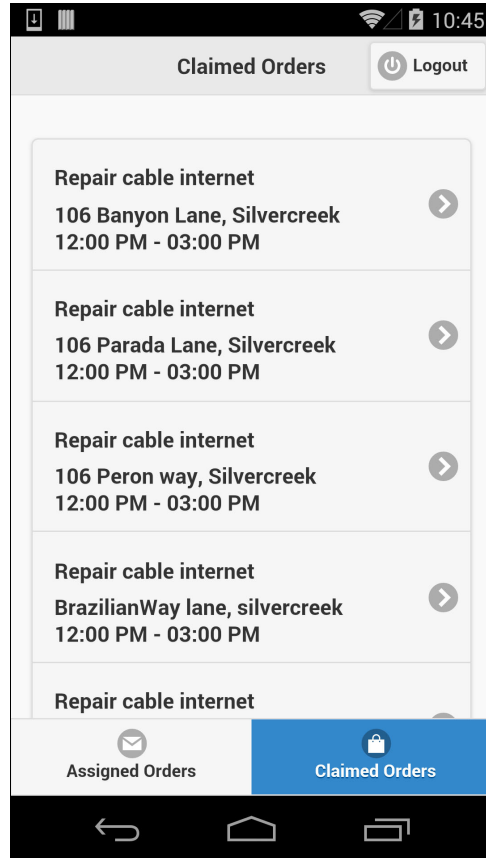


Figure 6-8 Field Service app: Claimed Orders page

- **Claimed Orders Details:** This page shows the details of the claimed order as shown in Figure 6-9 on page 121. From this page, the field technician can update the status of the order, and mark it as complete. The order details are filled dynamically. For implementation details, see “Claimed Order Details page” on page 139.

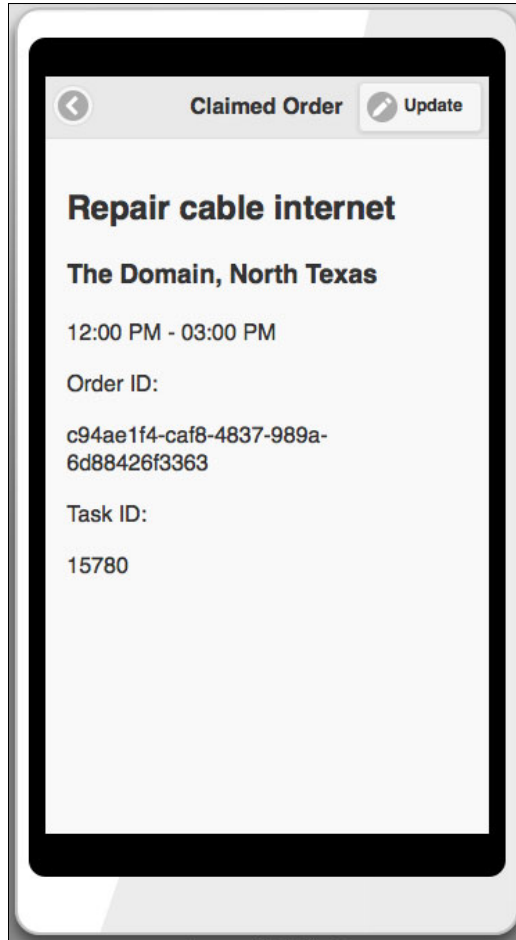


Figure 6-9 Field Service app: Claimed Orders Details page

6.3 Worklight features

This section describes the Worklight features shown in scenario 1:

- ▶ Worklight adapters for IBM BPM and data integration
- ▶ Adapter-based authentication
- ▶ Geolocation
- ▶ Offline storage
- ▶ Push notifications
- ▶ Worklight adapters for IBM BPM and data integration

6.3.1 Adapter-based authentication

This section describes the scenario 1 security configuration between Worklight and BPM.

Use case security requirements

Each field technician uses the personal credentials to log in to the Field Service app. The field technicians use their personal credentials (they never share credentials). The IBM BPM server is responsible for validating the login credentials.

Security implementation

Worklight can delegate the validation of the user credentials against a back-end system by using the adapter-based authentication configuration. When you use adapter-based authentication, the adapter is responsible for the collection of the user credentials and for propagating the credentials to the back-end system for authentication. The Worklight adapter serves as a conduit enabling the authentication to be delegated to IBM BPM.

Figure 6-10 shows the user authentication sequence of events.

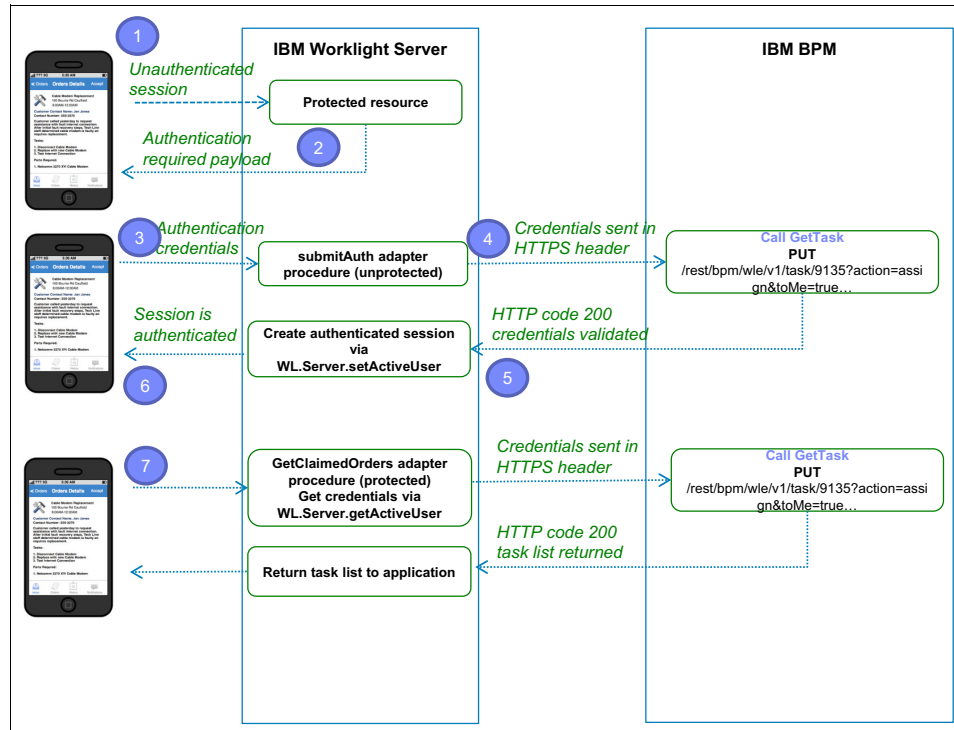


Figure 6-10 Adapter-based authentication logic flow

The flows depicted in Figure 6-10 are as follows:

1. When the application starts, it tries to access a protected resource on the Worklight Server.
2. The Worklight Server responds with an authentication-required security challenge.
3. The application receives the security challenge from the Worklight Server and calls the challenge handler function. The challenge handler function prompts the field technician for credentials and then sends the collected credentials to the submitAuthentication adapter procedure.
4. The submitAuthentication adapter procedure calls an IBM BPM REST API with the credentials in the HTTP header. Any IBM BPM REST API could be used to validate credentials as long as it is read-only. This call is done over HTTPS.
5. The submitAuthentication adapter receives response from the IBM BPM server with a 200 success code and then it establishes an authenticated session by calling the `WS.server.setActiveUser` function.
6. The application receives a user-authenticated confirmation response, along with the protected resource that was initially requested.
7. The next time that a protected adapter needs to call an IBM BPM REST service, the adapter gets the user credentials by calling the `WS.server.getActiveUser` function.

Additional considerations

In a production implementation, all adapter procedure calls to the back-end system should be over HTTPS.

In this implementation, the user ID and password are sent in the HTTP header following an HTTP basic authentication pattern. This implementation is particularly suitable when the Worklight Server is running on the Apache Tomcat application server.

If the Worklight Server runs on IBM WebSphere Application Server, a more robust approach is to implement Lightweight Third-Party Authentication (LTPA)-based authentication. LTPA is a security token type that is used by IBM WebSphere Application Server and other IBM products. LTPA can be used to send the credentials of an authenticated user to back-end services. It can also be used as a single sign-on (SSO) token between the user and multiple servers.

Using the LTPA token has some inherent advantages such as lower overhead, expanded identity information, and additional security because it is encrypted and signed. An LTPA-based implementation is shown in 7.4, “SSO configuration with LTPA implementation” on page 217.

6.3.2 Geolocation

The geolocation implementation leverages the Worklight location services APIs. These APIs enable the creation of differentiated services based on user location by collecting and feeding geolocation and Wi-Fi data to third-party systems.

Use case geolocation requirements

The cable TV installation service company wants to track the location of the field technicians during the work day. This location information is used for various purposes including validation of reported mileage, optimization of field technician locations based on where the work is done, and so on.

Geolocation implementation

Geolocation is a powerful differentiator and benefit provided by mobile apps. However, because geolocation coordinates must be constantly polled to understand where a mobile device is located, the resulting stream of geographic information can be difficult to manage without exhausting resources such as battery and network. Worklight includes location services that handle multiple geo modalities such as GPS, Wi-Fi sampling, and interpolation. You can set policies for acquiring geolocation data and transmitting it to the server to optimize battery and network usage.

In the implementation shown in this scenario, the Field Service app reports the location to the Worklight Server in a low accuracy fashion in order to conserve battery. Figure 6-11 on page 124 shows the sequence of events in the geolocation use case implementation.

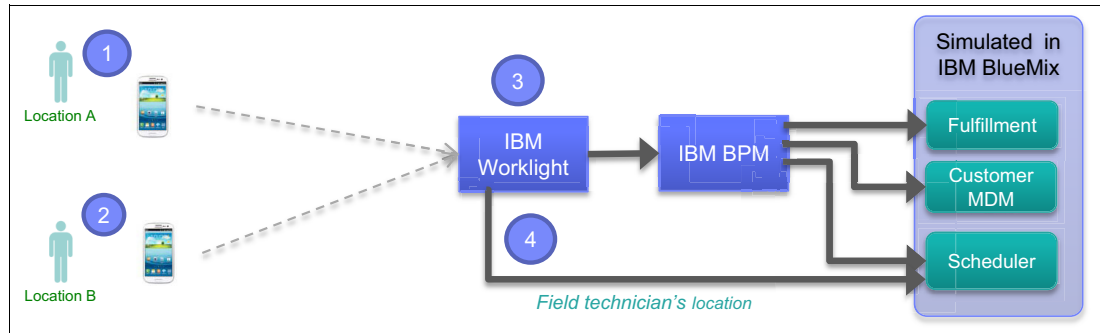


Figure 6-11 Geolocation implementation flow

1. The Field Service app acquires geolocation data from a device by using the `WL.Device.startAcquisition` API.
2. The `positionChange` trigger is activated if the position of the device changes significantly, and location change events can then be sent to the Worklight Server.
3. The Worklight Server handles these events by a configured event handler called `technicianMoved`.
4. When a new location change event arrives to the Worklight Server, the handler calls the `notifyPosition` adapter procedure, which sends the new location of the field technician to the work order scheduler simulated process in IBM Bluemix.

6.3.3 Offline storage

When you create an offline-enabled mobile application, it is useful to store information about the mobile device that can help preserve its functionality in offline mode. This information typically comes from the back-end system, and you must consider data synchronization with the back-end system as part of the application architecture.

IBM Worklight includes a feature called JSONStore for data exchange and storage. This feature adds the capability to store JSON documents in Worklight applications. It provides the ability to create, read, update, and delete data records from a data source. Each operation is queued when operating offline. When a connection is available, the operation is transferred to the server and each operation is then performed against the source data.

Figure 6-12 on page 125 shows a Worklight mobile application using the Worklight JSONStore API to interact with the embedded JSONStore database which can be fully encrypted. The JSONStore API supports full synchronization with the back-end system through a Worklight adapter. The data that is changed in the back-end is synchronized with the JSONStore through adapter calls. Changes on the data in the JSONStore while the device is offline are updated to the back-end when a connection becomes available through Worklight adapter calls.

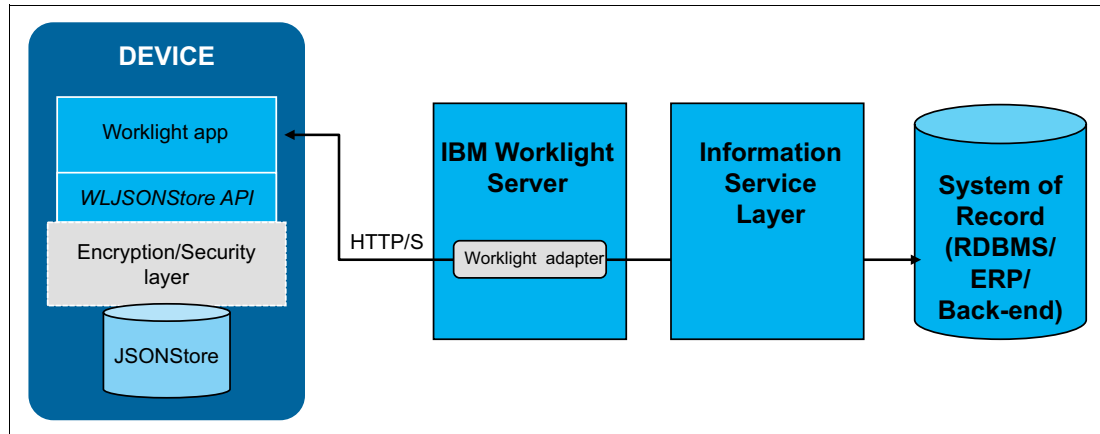


Figure 6-12 Worklight JSONStore overview

Use case offline storage requirements

The field technician must be able to update order status while being offline. The Field Service app submits the order updates when Internet access is restored.

Offline storage implementation

A work order is accepted by a field technician in the Field Service app by calling a Worklight adapter directly. During this interaction all the claimed orders are retrieved from the IBM BPM Server. This approach allows the field technician to maintain an up-to-date copy of all their work orders locally on the device. The retrieved work orders are stored in the local JSONStore on the device. The local JSONStore has a collection of the orders accepted by the field technician. Every future update to the work order is done on the JSONStore, and updated afterward on the IBM BPM Server when network connectivity exists.

Note: Another approach is to save a copy of the orders in the JSONStore once it is accepted by the user. Then, it is not required to load the claimed orders from IBM BPM server when the user accepts an order.

However, this example demonstrates the use case of having a local replica on the device of a certain back-end data. This snapshot is fully synchronized with the back-end system before the user starts working on it (pulled from the back-end system). The snapshot is then synchronized with the back-end system when the user finishes working on it (pushed to the back-end system).

The IBM Worklight runtime framework generates networking events, to which the application listens and captures changes in connectivity. The events are generated only on change of connectivity state. The `WL.Events.WORKLIGHT_IS_CONNECTED` event is generated when the application connects to the Worklight Server.

The Worklight application attempts to connect to the Worklight Server whenever the application is launched. In addition, the `WL.Client.setHeartBeatInterval(600)` method is used to set the interval of the heartbeat signal to the specified number of seconds. In other words, the Field Service app tries to connect to the server every 10 minutes (600 / 60) which triggers the networking events discussed previously.

Note: In the sample provided, network connectivity checks occur both before loading claimed orders from the IBM BPM Server and after updating the order status in the JSONStore to notify the IBM BPM Server of the latest changes. It will be a good exercise for the reader to create the code that listens for Worklight connectivity events, and whenever a network connection exists, prompts users to choose if they prefer to update the back-end system with the locally stored changes.

6.3.4 Push notifications

With push notifications, mobile applications can send information to mobile devices, even when the application is not being used. IBM Worklight includes a unified notification framework that provides a consistent mechanism for such push notifications. Push notifications provide the ability to send messages to the mobile device from a server. The most common form of notification is Short Message Service (SMS). Notifications are received regardless of whether the application is running.

The Worklight adapter framework allows developers to implement *event sources*. Event source is a push notification channel to which mobile applications can register. Event source is used to generate notification events that the IBM Worklight client framework can subscribe to, for example, push notifications. To send a notification, it must be retrieved from a back-end system first. Event source can either poll notifications from the back-end system, or wait for the back-end system to explicitly push a new notification.

Some characteristics of an event source are:

- ▶ An event source is defined within a Worklight adapter.
- ▶ To start receiving push notifications, an application must first subscribe to a push notification event source.
- ▶ The user must approve the push notification subscription.
- ▶ When the user approves the subscription, the device registers with an Apple, Google, or Microsoft push server to obtain a token that is used to identify the device and sends a subscription request to the Worklight Server.
- ▶ An event source is declared in the Worklight adapter that is used by the application for push notification services.

This operation is performed automatically by the IBM Worklight framework.

In addition to push notifications based on user subscription to an event source, IBM Worklight V6.2 implements an infrastructure with improved scalability, which brings the following features:

- ▶ Tag-based notification: In the descriptor file of the Worklight application, you add tags that define topics of interest, such as customer categories, and the application can subscribe to those tags to receive notification.
- ▶ Broadcast notification: Any push-enabled application is also enabled for broadcast.
- ▶ JavaScript and REST APIs:
 - New JavaScript APIs to manage tag subscriptions and to send notifications.
 - New REST APIs for the management of devices and subscriptions.

Use case push notification requirements

When a new work order is assigned to a field technician team, every member in the team must be notified by a push notification message on the mobile device.

Push notification implementation

The PushAdapter is an HTTP adapter that exposes a procedure called `submitNotification`, which is invoked by the IBM BPM server when new orders are available for a particular field technician team.

Tag-based notification for field technician team targeting is used to demonstrate push notifications in this scenario. The Field Service app subscribes to a push notification tag that is used for group notifications. The `<tags>` element in the `application-descriptor.xml` file is used. The IBM BPM server can then notify the applications that are targeted by a particular tag.

6.3.5 IBM BPM integration using adapters

Adapters are the server-side code of Worklight applications. Adapters connect to enterprise applications, deliver data to and from mobile applications, and perform any necessary server-side logic on this data.

Use case IBM BPM integration requirements

The Field Service app pulls the work orders assigned to the field technician team from the IBM BPM server. The work order status updates must be sent to the IBM BPM server.

IBM BPM integration implementation

The IBM BPM server exposes the process management features via an IBM BPM REST API. Worklight can access these services using adapters, which are the way that Worklight integrates with back-end systems. Three files define each Worklight adapter:

- ▶ An XML file that declares connectivity parameters, such as host name and port number.
- ▶ A JavaScript file that describes the procedures and connection logic.
- ▶ An optional XSLT file used for filtering data.

Figure 6-13 on page 128 shows the sequence of events for a typical integration interaction:

1. The field technician using the Field Service app retrieves the work orders that are available for assignment. The `getAssignedOrders` adapter procedure is called, which calls an IBM BPM REST API to retrieve a list of work orders.
2. After an order assignment confirmation, the `getAssignedOrderDetails` adapter procedure is invoked which calls an IBM BPM REST API that retrieves the work order business data.
3. The field technician browses the work orders available and accepts one order. The `acceptOrder` adapter procedure is called, invoking an IBM BPM REST API to assign the order to the field technician.
4. The field technician retrieves the list of the orders claimed using the `getClaimedOrders` adapter procedure, invoking an IBM BPM REST API that retrieves the work order business data.
5. The field technician updates the work order status. The `updateOrder` adapter procedure is called, which invokes an IBM BPM REST API to propagate the status change.

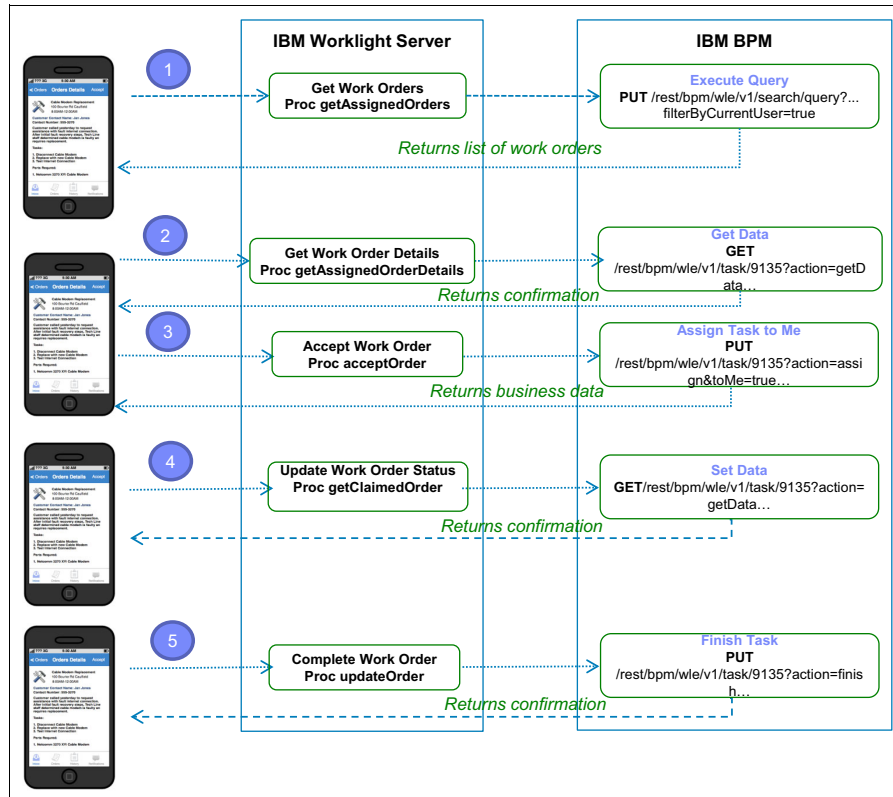


Figure 6-13 Worklight adapters and IBM BPM integration flow

6.4 Worklight implementation details

This section explains the Worklight implementation details for scenario 1. The implementation details include:

- ▶ Creating the adapters for the integration between Worklight and IBM BPM.
- ▶ Creating the UI for the Field Service app.
- ▶ Implementing the authentication mechanism for the mobile application.

Note: The implementation sections of this scenario assume that the reader has IBM Worklight development skills. For information about developing Worklight applications, see the IBM Knowledge Center, *Developing Worklight applications* at the following link:

http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.dev.doc/topics/t_devwlaps.html

See Appendix A, “Samples included with this book” on page 307 for information about the samples provided with the IBM Redbooks publication.

6.4.1 Worklight project

jQuery Mobile is the framework that is used in this chapter to create the UI pages. jQuery Mobile relies on the jQuery library (the utility library) as a dependency, which is included by default in Worklight projects. You only need to download and include the jQuery Mobile library.

To download the latest version of JQuery Mobile:

1. Download the latest version of the JQuery Mobile code from the jQuery mobile project website at <http://jquerymobile.com/download>

The version used in this sample is 1.4.3. Ensure that you use a 1.4.x release.

Note: The JQuery Mobile version used to develop the examples included in this chapter was 1.4.3. It is expected that the example will work on any JQuery Mobile version 1.4.x.

2. Extract the downloaded compressed file.

To create a Worklight project with the support of JQuery Mobile:

1. From Worklight Studio, click **File** → **New** → **Worklight Project**.
2. Enter the name of the project, **WL_BPM_Project** in this sample, and select **Hybrid Application** as the type of your app.
3. Click **Next**.
4. Enter the application name, **FieldService** in this sample.
5. To add your JQuery Mobile files you downloaded, click **Configure Java Script Libraries**.
6. Select **Add JQuery Mobile**.
7. Select the folder where you extracted the JQuery Mobile downloaded compressed file.
8. From the displayed files, select the following:
 - a. images folder
 - b. jquery.mobile-1.4.3.js
 - c. jquery.mobile-1.4.3.css
9. Click **Finish**.
10. Click **Finish** to close the JavaScript Libraries modal dialog.
11. Click **Finish** again to close the New Worklight Project dialog.
12. If the message Some pages may not render correctly when using Internet Explorer as the embedded browser is displayed click **OK**.
13. Worklight Studio might prompt you to switch to the Design perspective. Click **Yes**.

The JQuery Mobile files are copied to the directory

WL_BPM_Project/apps/FieldService/common/jqueryMobile and the lines shown in Example 6-1 are inserted in your WL_BPM_Project/apps/FieldService/common/index.html file to include the required JQuery Mobile files.

Example 6-1 JQuery Mobile include

```
<link href="jqueryMobile/jquery.mobile-1.4.3.css" rel="stylesheet">
<link rel="stylesheet" href="css/main.css">
<script>
    window.$ = window.jQuery = WLJQ;
</script>
<script src="jqueryMobile/jquery.mobile-1.4.3.js"></script>
```

6.4.2 Creating the adapters

There are four HTTP adapters needed for the implementation of scenario 1; they are:

- ▶ AuthenticationAdapter
- ▶ OrderAdapter
- ▶ PushAdapter
- ▶ LocationAdapter

Note: For an overview of Worklight adapters, see *Overview of Worklight adapters*, at:

http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.dev.doc/devref/c_overview_of_ibm_worklight_adap.html

For step-by-step instructions on creating Worklight adapters, see *Creating a Worklight adapter*, at the following link:

http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.dev.doc/devref/t_creating_a_new_ibm_worklight_a.html

AuthenticationAdapter

This adapter is an HTTP adapter created as part of the security authentication framework. This adapter integrates with the IBM BPM REST APIs to authenticate the user. If it is a valid user, the current active user identity is set to the authenticated credentials and the authentication is successful, otherwise the user is invalid and the user is not authenticated.

For information about IBM Worklight security framework, see the IBM Redbooks publication *Securing Your Mobile Business with IBM Worklight*, SG24-8179.

OrderAdapter

This adapter is an HTTP Adapter created to integrate with the IBM BPM REST APIs to get, accept, update, and finish orders.

Adapter XML file OrderAdapter.xml

The <connectivity> element defines the mechanism by which the adapter connects to the back-end application. Example 6-2 shows the connectivity element in the OrderAdapter.xml file, where the connection to the IBM BPM server is defined. The connection must be established to be able to invoke the IBM BPM REST APIs on it.

In file WL_BPM_Project/adapters/OrderAdapter/OrderAdapter.xml, note the <ConnectionPolicy> element sub-elements:

- ▶ Protocol: http
- ▶ Domain: IBM BPM server host name or host address, 10.12.6.27 in this example
- ▶ Port: IBM BPM server, 9080 in this example

Example 6-2 <connectivity> element and sub-elements: Connection configuration to IBM BPM server

```
<connectivity>
<connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
  <protocol>http</protocol>
  <domain>10.x.x.x</domain>
  <port>9080</port>
</connectionPolicy>
  <loadConstraints maxConcurrentConnectionsPerNode="2" />
</connectivity>
```

The <procedure> element in the adapter XML file defines a process for accessing a service exposed by a back-end application, which is an IBM BPM process in this scenario. The service can retrieve data from the back-end system or perform a transaction at the back-end system.

Example 6-3 shows procedures exposed by the adapter. There are five procedures exposed:

- ▶ acceptOrder
- ▶ getAssignedOrders
- ▶ getClaimedOrders
- ▶ getAssignedOrderDetails
- ▶ updateOrder

All procedures have the same security test applied to them, the *FieldServiceSecurityTest*. For information about Security Test, see *Worklight security overview* in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.deploy.doc/admin/c_security_overview.html?lang=en

Example 6-3 <procedure> element: Procedures in OrderAdapter.xml

```
<procedure name="acceptOrder"
securityTest="FieldServiceSecurityTest"></procedure>
<procedure name="getAssignedOrders" securityTest="FieldServiceSecurityTest">
</procedure>
<procedure name="getClaimedOrders" securityTest="FieldServiceSecurityTest">
</procedure>
<procedure name="getAssignedOrderDetails"
securityTest="FieldServiceSecurityTest"></procedure>
<procedure name="updateOrder" securityTest="FieldServiceSecurityTest">
</procedure>
```

JavaScript implementation of the procedures: OrderAdapter-impl.js

The OrderAdapter-impl.js file includes the JavaScript implementation of the procedures exposed by the adapter and also some internal helper functions used by the procedures:

- ▶ acceptOrder procedure

Example 6-4 shows the sample code for the acceptOrder procedure. It is called when the user accepts the order from the mobile app.

The Assign Task to Me IBM BPM REST API exposed for this procedure is as follows:

'/rest/bpm/wle/v1/task/'+taskId+'?action=assign&toMe=true&parts=none'

where, taskId is sent from the UI; it is the ID of the task to be accepted by the user.

To invoke the IBM BPM REST API from the adapter, the following properties are set in the input JSON passed to the WL.Server.invokeHttp method:

- The method is PUT.
- The returnedContentType is JSON.
- The path is the IBM BPM REST API.
- The header is set with the current authenticated user.

Example 6-4 acceptOrder procedure in OrderAdapter-impl.js

```
//accept an assigned order
function acceptOrder(taskId) {
```

```

WL.Logger.warn("acceptOrder: enter " + taskId);

var input = {
    method : 'put',
    returnedContentType : 'json',
    path :
'/rest/bpm/wle/v1/task/'+taskId+'?action=assign&toMe=true&parts=none',
    headers : {
        Authorization : WL.Server.getActiveUser().credentials
    }
};

WL.Logger.warn("acceptOrder: exit");

return WL.Server.invokeHttp(input);
}

```

► **getAssignedOrders procedure**

This procedure is called by the UI to get all the orders assigned to the current user's group (assigned but not accepted, that is not *claimed* yet). Example 6-5 on page 133 shows the sample code for the `getAssignedOrders` procedure.

The Execute Query IBM BPM REST API exposed for that procedure is as follows:

'rest/bpm/wle/v1/search/query?organization=byInstance&run=true&shared=false&filterByCurrentUser=true&columns=orderId%2Ctitle%2CtimeRequired%2CstreetAddress%2Csuburb'

The `filterByCurrentUser=true` parameter ensures the data returned is filtered by the current user.

In order to invoke the IBM BPM REST API from the adapter, the following properties are set in the input JSON passed to the `WL.Server.invokeHttp` method:

- The method is PUT
- The returnedContentType is JSON
- The path is the IBM BPM REST API
- The header is set with the current authenticated user

After invoking the IBM BPM REST API, the returned orders are filtered to return the *not* closed orders *only*. Also the `taskAssignedTo.type` is checked in the returned JSON to return only orders that are assigned to a Group, which means that no field technician has claimed (accepted) them yet:

- If it is Group, the order is not accepted yet and it is added to the `ordersList` returned by the procedure.
- If it is User, it is ignored.

Only the properties needed by the UI are added to `ordersList` (not every property returned by the IBM BPM REST API is needed). Custom objects `orders` are added to `ordersList` with the needed properties.

A utility function `getOrdersByType` is created to retrieve either the assigned orders or claimed orders from IBM BPM, based on either the User or Group parameter being passed in with the request.

```
//retrieve assigned orders: orders for a group
function getAssignedOrders() {

    WL.Logger.warn("getAssignedOrders: enter");
    WL.Logger.warn("User: " + WL.Server.getActiveUser().userId);

    var orders = getOrdersByType("Group");
    WL.Logger.warn("Orders" + JSON.stringify(orders));

    return orders;
}

//utility function to get orders by type
//group type for assigned orders
//user type for claimed orders
function getOrdersByType(type){
    WL.Logger.warn("type : " + type);

    var input = {
        method : 'put',
        path:
'rest/bpm/wle/v1/search/query?organization=byInstance&run=true&shared=false&filter
ByCurrentUser=true&columns=orderId%2Ctitle%2CtimeRequired%2CstreetAddress%2Csuburb
',
        headers : {
            Authorization : WL.Server.getActiveUser().credentials
        }
    };

    var result = WL.Server.invokeHttp(input);

    WL.Logger.warn("result : " + JSON.stringify(result));

    var ordersList = [];

    //populate returned array
    for(var i=0; i<result.data.data.length; i++){
        var datetime = getTimeDetails(result.data.data[i].timeRequired);
        var order= {"taskId": result.data.data[i].taskId,
            "title": result.data.data[i].title,
            "orderId": result.data.data[i].orderId,
            "streetAddress": result.data.data[i].streetAddress,
            "suburb": result.data.data[i].suburb,
            "datetime": datetime,
            "businessStatus": "NA",
            "taskStatus": result.data.data[i].taskStatus,
            "taskAssignedTo": {
                "type": result.data.data[i].taskAssignedTo.type,
                "who": result.data.data[i].taskAssignedTo.who
            }
        };

        //don't return finished orders
```

```

        if(result.data.data[i].taskAssignedTo.type == type &&
result.data.data[i].taskStatus != "Closed"){
            ordersList.push(order);
        }
    }

    vardata = {"orders" : ordersList};

    return data;
}

//time is returned as numbers that map to the time shifts
function getTimeDetails(timRequired){
    var datetime = "";
    if(timRequired == 0){
        datetime = "09:00 AM - 12:00 PM";
    }
    else if(timRequired == 1){
        datetime = "12:00 PM - 03:00 PM";
    }
    else if(timRequired == 2){
        datetime = "03:00 PM - 05:00 PM";
    }
    else if(timRequired == 3){
        datetime = "05:00 PM - 07:00 PM";
    }
    else{
        datetime = "null";
    }
    return datetime;
}

```

► **getClaimedOrders procedure**

This procedure is called by the UI to get all the orders accepted by the current user. Example 6-6 shows the sample code for the getOrder procedure.

It has the same logic as in the function getAssignedOrders, but it filters the returned orders to get only the ones with taskAssignedTo.type equals User. It calls the utility function getOrdersByType.

Example 6-6 getClaimedOrders procedure in OrderAdapter-imp.js

```

//returns list of accepted orders
function getClaimedOrders() {

    WL.Logger.warn("getOrder: enter");
    WL.Logger.warn("User: " + WL.Server.getActiveUser().userId);

    var orders = getOrdersByType("User");
    WL.Logger.warn("Orders" + JSON.stringify(orders));
    return orders;
}

```

► `getAssignedOrderDetails` procedure

This procedure is called by the UI to get the details of the currently selected order for the current user. Example 6-7 shows the sample code for the `getAssignedOrderDetails`.

The Get Data IBM BPM REST API exposed for that procedure is as follows:

`'/rest/bpm/wle/v1/task/'+taskId+'?action=getData&fields=workOrder'`

where, `taskId` is the currently selected task, sent from the UI.

In order to invoke the IBM BPM REST API from the adapter, the following properties are set in the input JSON passed to the `WL.Server.invokeHttp` method:

- The method is GET.
- The `returnedContentType` is JSON.
- The path is the IBM BPM REST API.
- The header is set with the current authenticated user.

In the returned order, only the properties needed by the UI are added (not all the properties returned by the IBM BPM REST API).

Example 6-7 `getAssignedOrderDetails` procedure in `OrderAdapter-impl.js`

```
//get details of the assigned order
function getAssignedOrderDetails(taskId) {

    var input = {
        method : 'get',
        returnedContentType : 'json',
        path : '/rest/bpm/wle/v1/task/' + taskId
            + '?action=getData&fields=workOrder',
        headers : {
            Authorization : WL.Server.getActiveUser().credentials
        }
    };

    var workOrderResult = WL.Server.invokeHttp(input);
    WL.Logger.warn("workOrderResult: " + JSON.stringify(workOrderResult));

    var order = {};

    if (workOrderResult.data.resultMap.workOrder === null) {///if work order is
    null, populate with nulls

        order = {
            "taskId" : taskId,
            "title" : "null",
            "orderId" : "null",
            "streetAddress" : "null",
            "suburb" : "null",
            "postCode" : "null",
            "datetime" : "null",
            "description" : "null",
            "tasks" : "null"
        };
    } else {

        order = {
```

```

        "taskId" : taskId,
        "title" : workOrderResult.data.resultMap.workOrder.title,
        "orderId" : workOrderResult.data.resultMap.workOrder.orderId,
        "streetAddress" :
workOrderResult.data.resultMap.workOrder.location.streetAddress,
        "suburb" : workOrderResult.data.resultMap.workOrder.location.suburb,
        "postCode" : workOrderResult.data.resultMap.workOrder.location.postCode,
        "datetime" :
getTimeDetails(workOrderResult.data.resultMap.workOrder.timeRequired),
        "description" : workOrderResult.data.resultMap.workOrder.description,
        "tasks" : workOrderResult.data.resultMap.workOrder.tasks
    };
}

return {
    "order" : order
};
}

```

► updateOrder procedure

This procedure is called by the UI to change the status of the order to one of the following values: In My Way, On Site, or Complete:

- If the status passed by the UI is In My Way, or On Site, the procedure invokes the Set Data IBM BPM REST API as follows:

```

'/rest/bpm/wle/v1/task/'+taskId+'?action='+action+'&params=%7B%22workOrderReport%22%3A%7B%22orderId%22%3A%22'+orderId+'%22%2C%22status%22%3A%22'+status+'%22%2C%22summary%22%3A%22%22%7D%7D&parts=none'

```

where, action is set to setData, taskId is the currently selected task, orderId is the currently selected order ID, and status is the new status to be changed to.

The status here is the business-related status, which is modified as part of the workOrderReport JSON as in the URL.

- If the status passed by the UI is Complete, the procedure calls the Finish Task IBM BPM REST API, as follows:

```

'/rest/bpm/wle/v1/task/'+taskId+'?action='+action+'&params=%7B%22workOrderReport%22%3A%7B%22orderId%22%3A%22'+orderId+'%22%2C%22status%22%3A%22'+status+'%22%2C%22summary%22%3A%22%22%7D%7D&parts=none'

```

where, action is set to finish, taskId is the currently selected task, orderId is the currently selected order ID, and status is the new status to be changed to.

In order to invoke the IBM BPM REST API from the adapter, the following properties are set in the input JSON passed to the WL.Server.invokeHttp method:

- The method is PUT.
- The returnedContentType is JSON.
- The path is the IBM BPM REST API.
- The header is set with the current authenticated user.

Example 6-8 on page 137 shows the sample code for the updateOrder procedure.

Example 6-8 updateOrder procedure in OrderAdapter-imp.js

```
// updating the status of the work order
function updateOrder(taskId,orderId, status) {

    var action;

    if(status === "complete"){
        action = "finish";
    }else{
        action = "setData";
    }

    status = encodeURIComponent(status.trim());

    var input = {
        method : 'put',
        returnedContentType : 'json',
        path :
        '/rest/bpm/wle/v1/task/'+taskId+'?action='+action+'&params=%7B%22workOrderReport%22%3A%7B%22orderId%22%3A%22'+orderId+'%22%2C%22status%22%3A%22'+status+'%22%2C%22summary%22%3A%22%22%7D%7D&parts=none',

        //basic authentication with BPM
        headers : {
            Authorization : WL.Server.getActiveUser().credentials
        }
    };

    var result = WL.Server.invokeHttp(input);
    return result;
}
```

PushAdapter

This adapter is an HTTP adapter used for the server-side implementation of the Tag-based push notification framework. This adapter is described in details in section 6.4.7, “Push notifications support” on page 154.

LocationAdapter

This adapter is an HTTP adapter used as an event handler for the events transmitted by the geolocation acquisition from the client-side. This adapter is described in details in section 6.4.8, “Geolocation support” on page 157.

6.4.3 User Interface

jQuery Mobile is the framework that is used to create the UI pages. As a typical jQuery Mobile application, pages are implemented as sections in a single HTML file, WL_BPM_Project/apps/FieldService/common/index.html in this scenario, and their implementation is described in the following sections.

Assigned Orders page

This is the page that shows the orders assigned to the field technician, who is the user of the Field Service app. The field technician can select to accept (claim) this order, execute it, or

ignore it. The UI is shown in Figure 6-5 on page 117. Example 6-9 shows the Inbox page HTML code in index.html.

Example 6-9 Assigned Orders page HTML code

```

<div data-role="page" id="page-assigned-orders" data-theme="a">
  <div data-role="content">
    <div id="orders-header" data-role="header" data-position="fixed">
      <h1>Assigned Orders</h1>
      <a id="button-logout" href="#" data-icon="power"
class="ui-btn-right">Logout</a>
    </div>
    <ul data-role="listview" id="assigned-orders" data-inset="true">
    </ul>
  </div>
  <div data-role="footer" data-position="fixed">
    <div data-role="navbar">
      <ul>
        <li><a href="#page-assigned-orders" data-icon="mail"
class="ui-btn-active ui-state-persist">Assigned Orders</a></li>
        <li><a href="#page-claimed-orders" data-icon="shop">Claimed
Orders</a></li>
      </ul>
    </div>
  </div>
</div>

```

This page is referred to as the page-assigned-orders page in the sample. It includes the following:

- ▶ Header at the top of the page, including title and Logout button.
- ▶ Listing of the assigned orders using JQuery Mobile list view represented in the element with ID assigned-orders. The list is populated dynamically using Java Script.
- ▶ At the bottom of the page, the tab control that is used to navigate between the main pages. The tab control is implemented using the navbar view in JQuery Mobile. It has links to the main pages: Assigned Orders and Claimed Orders.

Assigned Order Details page

Example 6-10 shows the HTML code for this page. It is coded in the <div page-assigned-order-details> element. It has a listing for the order details, and the button to accept this order in the header at the top. The UI is shown in Figure 6-6 on page 118.

Example 6-10 Assigned Order Details page HTML code

```

<div data-role="page" id="page-assigned-order-details" data-theme="a">
  <div id="orders-header" data-role="header" data-position="fixed">
    <a id="button-back" href="#page-assigned-orders" class="ui-btn-left
ui-btn ui-icon-carat-l ui-btn-icon-notext ui-corner-all"></a>
    <h1>Assigned Order</h1>
    <a id="button-accept" href="#" data-icon="check"
class="ui-btn-right">Accept</a>
  </div>
  <div data-role="content">
    <h2 id="order-title"></h2>

```

```

        <h3 id="order-address"></h3>
        <p id="order-datetime"></p>
        <p id="order-description"></p>
        <span>Task ID:</span><p id="order-task-id"></p>
    </div>
</div>

```

Claimed Orders page

This page is referred to as the Claimed Orders page in the sample, since it displays the orders accepted and claimed by the logged-in field technician. The listing of the orders is filled dynamically. The UI is shown in Figure 6-8 on page 120. Example 6-11 shows the HTML code for this page.

Example 6-11 Claimed Orders page HTML code

```

<div data-role="page" id="page-claimed-orders">
    <div id="orders-header" data-role="header" data-position="fixed">
        <h1>Claimed Orders</h1>
        <a id="button-logout" href="#" data-icon="power"
class="ui-btn-right">Logout</a>
    </div>
    <div data-role="content">
        <ul data-role="listview" id="claimed-orders" data-inset="true">
        </ul>
    </div>
    <div data-role="footer" data-position="fixed">
        <div data-role="navbar">
            <ul>
                <li><a href="#page-assigned-orders" data-icon="mail">Assigned
Orders</a></li>
                <li><a href="#page-claimed-orders" data-icon="shop"
class="ui-btn-active ui-state-persist">Claimed Orders</a></li>
            </ul>
        </div>
    </div>
</div>

```

Claimed Order Details page

This page shows the details of the claimed order. From this page, the field technician can update the status of the order, and mark it as complete. The order details are filled dynamically. Example 6-12 shows the HTML code for this page.

Example 6-12 Claimed order details page HTML code

```

<div data-role="page" id="page-claimed-order-details" data-theme="a">

    <div id="orders-header" data-role="header" data-position="fixed">
        <a id="button-back" href="#page-claimed-orders" class="ui-btn-left ui-btn
ui-icon-carat-l ui-btn-icon-notext ui-corner-all"></a>
        <h1>Claimed Order</h1>
        <a id="button-update" href="#popup-update-order" data-rel="popup"
data-transition="pop" data-icon="edit" class="ui-btn-right">Update</a>
    </div>

```

```

<div data-role="content">
  <h2 id="order-title"></h2>
  <h3 id="order-addresse"></h3>
  <p id="order-datetime"></p>
  <p id="order-description"></p>
  <span>Order ID:</span><p id="order-order-id"></p>
  <span>Task ID:</span><p id="order-task-id"></p>
  <p id="order-json-id" hidden ></p>
</div>
<div data-role="popup" id="popup-update-order" data-corners="true"
data-dismissible="true" data-shadow="true" data-position-to="window"
class="ui-content" style="max-width:500pt;">
  <a href="#" data-rel="back" data-role="button" data-theme="a"
data-icon="delete" data-iconpos="notext" class="ui-btn-right">Close</a>
  <p>Select a status update:</p>
  <div>
    <a href="javascript:updateShownOrder('onmyway')" class="ui-btn
ui-corner-all ui-btn-inline ui-icon-location ui-btn-icon-right" style="width:
70%;">On My Way</a>
  </div>
  <div>
    <a href="javascript:updateShownOrder('onsite')" class="ui-btn
ui-corner-all ui-btn-inline ui-icon-alert ui-btn-icon-right" style="width:
70%;">On Site</a>
  </div>
  <div>
    <a href="javascript:updateShownOrder('complete')" class="ui-btn
ui-corner-all ui-btn-inline ui-icon-check ui-btn-icon-right" style="width:
70%;">Complete</a>
  </div>
</div>
</div>

```

When loading the order details, the following elements in the page Document Object Model (DOM) store the data needed to communicate with the adapter and the JSONStore:

- ▶ `order-order-id`: Stores order ID.
- ▶ `order-task-id`: Stores task ID
- ▶ `order-json-id`: Stores ID of the document representing this order in the JSONStore, which is a hidden element.

When the user updates the order, these values are read and used in the request to the adapter or to the JSONStore.

6.4.4 Security and authentication

The Worklight security framework is used to secure the application and the adapter calls on Worklight Server. Adapter-based authentication is the authentication mechanism used in this scenario. The following sections describe the code and configuration for the sample.

For information about IBM Worklight security framework, see the IBM Redbooks publication *Securing Your Mobile Business with IBM Worklight*, SG24-8179 and *Worklight security framework* in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.dev.doc/dev/r_security_framework.html

Server-side configuration

This section describes the artifacts that are deployed on the Worklight Server.

Authentication configuration

The `authenticationConfig.xml` file, in the `WL_BPM_Project/server/conf` folder, is used to configure how the Field Service app authenticates users. The following elements in the `authenticationConfig.xml` file are relevant to the configuration in this sample:

- **Security Test:** A custom security test, with user validation (see Example 6-13).

Example 6-13 Security test definition for adapter-based authentication

```
<customSecurityTest name="FieldServiceSecurityTest">
  <test isInternalUserID="true" realm="FieldServiceAuthRealm"/>
</customSecurityTest>
```

- **Login Module:** Since adapter-based authentication is used, credentials are validated in the adapter, not in the login module. The `NonValidationLoginModule` class is used (see Example 6-14).

Example 6-14 NonValidation login module definition for adapter-based authentication

```
<loginModule name="AuthLoginModule">
  <className>com.worklight.core.auth.ext.NonValidatingLoginModule</className>
</loginModule>
```

- **Security Realm:** It is the `<realms>` element that attaches the Login Module and the Authenticator. The authenticator defines how to collect credentials. The Login Module configuration defines how to validate these credentials.
- The `AdapterAuthenticator` class is used in this example. In the parameters of the authenticator class, the login and logout functions in the adapter are provided. Note that `AuthenticationAdapter` is the name of the adapter used in validating the credentials.

Example 6-15 Definition of security realm in authenticationConfig.xml

```
<realm loginModule="AuthLoginModule" name="FieldServiceAuthRealm">
  <className>com.worklight.integration.auth.AdapterAuthenticator</className>
  <parameter name="login-function"
value="AuthenticationAdapter.onAuthRequired"/>
  <parameter name="logout-function"
value="AuthenticationAdapter.onLogout"/>
</realm>
```

AuthenticationAdapter

This adapter is used to connect to the IBM BPM server and validate the credentials.

The adapter has three main functions:

- **onAuthRequired:** This function returns the required result in the response that tells the challenge handler on the client-side whether an authentication is needed or not (see Example 6-16 on page 142).

Example 6-16 Function called when authentication is required on the server side

```
//returns the value in the response that tells the client that an
//authentication is needed
function onAuthRequired(headers, errorMessage){
    errorMessage = errorMessage ? errorMessage : null;

    return {
        authRequired: true,
        errorMessage: errorMessage
    };
}
```

- ▶ **submitAuthentication:** This function validates the credentials. If they are valid, the credentials are stored as the active user in the Worklight Server session using the `setActiveUser` API (see Example 6-17).

The REST URL `rest/bpm/wle/v1/user/current` on the IBM BPM server-side is used to validate the credentials. Credentials are base64 encoded before sending them in the authorization header.

The API `setActiveUser()` in the Worklight Server is used to set the authenticated user identity as the active user in the session. This identity is retrieved and sent to the IBM BPM back-end in subsequent IBM BPM REST calls.

Example 6-17 submitAuthentication function to validate credentials on the server side

```
//do authentication
function submitAuthentication(username, password){
    WL.Logger.warn("submitAuthentication: enter");

    var result = {};

    var authHeader = 'Basic ' + base64_encode(username + ':' + password);

    var input = {
        method : 'get',
        path : "rest/bpm/wle/v1/user/current",
        headers : {
            Authorization : authHeader
        }
    };
    var response = WL.Server.invokeHttp(input);

    if (response.statusCode == 200) {

        WL.Logger.warn("submitAuthentication: server invocation succeeded");

        var userIdentity = {
            userId : username,
            credentials : authHeader
        };

        WL.Server.setActiveUser("FieldServiceAuthRealm", userIdentity);

        result = {
```

```

        authRequired : false
    };
} else {

    WL.Logger.warn("submitAuthentication: server invocation failed");
    result = onAuthRequired(null, response.statusCode + ": " +
response.statusReason);
}

    WL.Logger.warn("submitAuthentication: return");
    return result;
}

```

- **onLogout:** When the logout button is pressed, this function is called by the framework to set the active user to null (see Example 6-18).

Example 6-18 Logout function on the server side

```

function onLogout(){
    WL.Server.setActiveUser("FieldServiceAuthRealm", null);
    WL.Logger.debug("Logged out");
}

```

Client-side configuration

This section describes the Worklight artifacts and configurations on the client side.

Challenge handler

You must declare a challenge handler in the application to handle challenges from the form-based configured realm. The challenge handler code is in the `authenticationRealmChallengeProcessor.js` file. Notice the following sections in the code:

- **isCustomResponse:** This function is called each time that a response is received from the server. It is used to detect whether the response contains data that is related to this challenge handler. It must return either `true` or `false`.

If this function returns `true`, the `handleChallenge()` function is called by the framework to check whether authentication is needed. Authentication is needed if no authentication was done at all or done but with invalid credentials (see Example 6-19). It checks whether `authRequired` was returned in the JSON coming from the server. Note that `authRequired` was returned in the result of `onAuthRequired()` function in the adapter on the server side.

Example 6-19 Checking response on the client side

```

challengeHandler.isCustomResponse = function(response) {
    if (!response || !response.responseJSON || response.responseText === null) {
        return false;
    }
    if (typeof(response.responseJSON.authRequired) !== 'undefined'){
        return true;
    } else {
        return false;
    }
};

```

- **handleChallenge:** This function is used to perform required actions, such as hide application screens and show the login screen.

It checks the response and if authentication is still required, it displays the login page. If authentication is not required, it displays the default page, which is the Assigned Orders page, and the framework is notified that authentication is successful by calling `challengeHandler.submitSuccess()`. See Example 6-20.

Example 6-20 Handling challenge on the client side

```
challengeHandler.handleChallenge = function(response){
    var authRequired = response.responseJSON.authRequired;

    if (authRequired == true){
        //switch to login page
        $('#message-invalid-login').hide();
        $.mobile.navigate('#page-login');
        $('#input-password').val('');
    } else if (authRequired == false){
        //switch to main page
        $.mobile.navigate("#page-assigned-orders");
        challengeHandler.submitSuccess();
    }
};
```

- The Java Script event registration of the Login button. When the Login button is clicked, the framework is invoked to do the validation process, and consequently calls the authentication adapter. By calling `challengeHandler.submitAdapterAuthentication()`, the specified adapter is used for authentication in the framework. See Example 6-21.

Example 6-21 Binding authentication to the login button

```
$("#link-login").bind('click', function () {
    var username = $("#input-username").val();
    var password = $("#input-password").val();

    var invocationData = {
        adapter : "AuthenticationAdapter",
        procedure : "submitAuthentication",
        parameters : [ username, password ]
    };

    challengeHandler.submitAdapterAuthentication(invocationData, {});
});
```

6.4.5 Calling adapters directly

In the Assigned Orders page, calls to get the orders assigned to the field technicians group, and the subsequent steps to accept the order are implemented by calling directly the relevant adapters, not through JSONStore as in the claimed orders calls (Claimed Orders page).

Retrieving assigned orders

Assigned orders are retrieved by an adapter call in Worklight. When successful, the UI is updated (see Example 6-22 on page 145).

Example 6-22 Loading assigned orders from adapter

```
//load assigned orders
function loadAssignedOrders(){
    var invocationData = {
        adapter : "OrderAdapter",
        procedure: "getAssignedOrders",
        parameters: []
    };

    WL.Client.invokeProcedure(invocationData, {
        onSuccess: function(response){
            //update UI when invocation is successful
            updateAssignedOrdersUI(response.invocationResult.orders);
        },
        onFailure: function(response){
            WL.Logger.warn(JSON.stringify(response.invocationResult));
        }
    });
}
```

After a successful call, the UI is updated using the typical JQuery selectors. The function to go to the details page is rendered as part of the <a href > element of the list item (see Example 6-23).

Example 6-23 Updating Assigned Orders page UI

```
//update assigned orders page
function updateAssignedOrdersUI(orders){

    $("#assigned-orders").empty();

    for ( var i = 0; i < orders.length; i++) {
        var li = document.createElement("li");
        li.innerHTML =
            "<a href='javascript:goToAssignedOrderDetailsPage(" +
orders[i].taskId + ")'>" +
            "<h2>" + orders[i].title + "</h2>" +
            "<div>" + orders[i].streetAddress + ", " + orders[i].suburb +
"</div>" +
            "<div>" + orders[i].datetime + "</div>" +
            "</a>";

        $("#assigned-orders").append(li);
    };

    //refresh JQuery Mobile control
    $("#assigned-orders").listview("refresh");
}
```

Showing the Details (Assigned Order Details) page

As in loading the list of orders, loading the details of an order follows the same approach by calling the adapter directly (see Example 6-24 on page 146).

Example 6-24 Retrieving order details from adapter

```
//load details for assigned order
function loadAssignedOrderDetails(taskId){
    var invocationData = {
        adapter : "OrderAdapter",
        procedure: "getAssignedOrderDetails",
        parameters: [taskId]
    };

    WL.Client.invokeProcedure(invocationData, {
        onSuccess: function(response){

            //when invocation is successful, switch to details page and update UI
            $.mobile.navigate("#page-assigned-order-details");
            updateAssignedOrderDetailsUI(response.invocationResult.order);
        },
        onFailure: function (response){
            WL.Logger.warn(JSON.stringify(response.invocationResult));
        }
    });
}

//update UI in assigned order details page
function updateAssignedOrderDetailsUI(order){

    $("#page-assigned-order-details #order-title").text(order.title);

    $("#page-assigned-order-details #order-address").text(order.streetAddress + ",
" + order.suburb);

    $("#page-assigned-order-details #order-datetime").text(order.datetime);

    $("#page-assigned-order-details #order-description").text(order.description);

    $("#page-assigned-order-details #order-task-id").text(order.taskId);
}
```

Accepting an order

To accept an order, the adapter is called directly, then a dialog box is shown using the Worklight client API `WL.SimpleDialog` (see Example 6-25).

Each time an order is accepted, the list of claimed orders are loaded from the IBM BPM server, and updated to the JSON Store. With this approach, there is always an updated snapshot of the claimed orders in the JSON Store available whenever the user goes offline. The JSON Store is initialized before using it.

Example 6-25 Accepting an order through adapters

```
//accept an assigned order
function acceptAssignedOrder(taskId){
    var invocationData = {
        adapter : "OrderAdapter",
        procedure: "acceptOrder",
```

```

        parameters: [taskId]
    };

    WL.Client.invokeProcedure(invocationData).then( function(response){
        //show confirmation dialog
        if(response.status == 200){
            WL.SimpleDialog.show("Alert", "Order accepted successfully!", [{text:
"OK", handler: null}], null);
        }

    }).then( function(response){

        return initJSONStore();

    }).then( function(response){

        return loadClaimedOrders();

    }).fail( function(errorObject){

        WL.Logger.warn(JSON.stringify(errorObject));

    });
}

```

6.4.6 Supporting offline storage

This section describes the implementation details for the offline storage use case.

Adding the JSONStore feature

Worklight includes client-side JavaScript APIs to be used in the mobile application code to create the JSONStore and operate on it. You need to add the JSONStore feature to the Worklight application to be able to use Worklight JSON Store client APIs.

Perform the following steps:

1. In the Worklight project, open the file:
WL_BPM_Project/apps/FieldService/application-descriptor.xml.
2. Make sure that the Design tab is active in the file editor.
3. Select **Optional Features**, and click **Add**.
4. Select **JSONStore**, and click **OK**.
5. Save your changes.

JSONStore initialization

Since the Claimed Orders page reads from the JSONStore, initialization is done when displaying this page. According to the JQuery promise shown, when the initialization is successful, the function to load the claimed orders is called (see Example 6-26).

Example 6-26 Location of JSON Store initialization

```

//load claimed orders when switching to claimed orders page
$( "#page-claimed-orders" ).on( "pageshow", function( event ) {
    $.when( initJSONStore() ).then(function(){

```

```

        loadClaimedOrders();
    })
    .fail(function(response){
        WL.Logger.warn(JSON.stringify(response));
    });
});

```

In Example 6-27, the code to initialize the JSONStore is shown. Note the following:

- ▶ One collection that represents the orders is created.
- ▶ All attributes in the order are represented as search fields (like columns) in the collection.
- ▶ OrderAdapter is attached to this collection for synchronization with IBM BPM.
- ▶ In this example, only `getClaimedOrders()` is used to retrieve all the orders accepted by the logged in user, and `updateOrder()` to update the order status.
- ▶ Note the promise created in the statement `jQuery.Deferred()` in Example 6-27. The objective is to control the sequence of calls in an asynchronous manner. Calls inside the function `initJSONStore()` follow this sequence:
 - a. When the call to `WL.JSONStore.init(collections, options)` is successful, `dfd.resolve()` is called.
 - b. The function `initJSONStore()` signals as completed with success when `dfd.resolve()` is called, and with failure when `dfd.reject()` is called.
 - c. Then, the promise is returned at the end of the function.
 - d. Using the promise made, the function `initJSONStore()` acts as if it is one block or thread, even when the call to `WL.JSONStore.init()` inside it is asynchronous, running in a different thread.
 - e. Therefore, the function waiting for `initJSONStore()` to complete can resume execution.
 - f. In this case, `loadClaimedOrders()` was waiting for `initJSONStore()` to complete (see Example 6-26 on page 147).
 - g. By using promises, it is guaranteed that `loadClaimedOrders()` will not be called before `initJSONStore()` and the asynchronous calls inside it are complete.

Example 6-27 JSONStore initialization

```

function initJSONStore(){
    var collections = {};

    //Define the 'people' collection and list the search fields
    collections['orders'] = {
        searchFields : {orderId: 'string', taskId: 'number' },
    };

    var options = {};

    //creating the promise
    var dfd = new jQuery.Deferred();

    WL.JSONStore.init(collections, options)
        .then(function(){
            dfd.resolve();
        })
        .fail(function(errorObject){
            WL.Logger.warn(JSON.stringify(errorObject));
        });
}

```

```

        dfd.reject();
    });

    return dfd.promise();
}

```

Note: Note that the JSON Store was initialized when accepting an order. In general, the call to the API `WL.JSONStore.init()` can be done before any operation with the JSONStore to ensure that it is initialized. Initialization of a previously initialized JSONStore does nothing, following a Singleton pattern.

Loading data

For loading data the code runs as follows:

- Check for network connectivity using Worklight APIs. If network connectivity exists, do the following:
 - a. Clear data in orders collection using the function `clear()`. Avoid using the `remove()` API since it requires reinitialization of the JSONStore.
 - b. When the clear operation is complete, connect to the adapter to load the orders and update the JSONStore.

Note: The `load()` and `push()` functions that were used in JSONStore synchronization with the adapters are deprecated in Worklight 6.2.

- c. The JSONStore has the latest snapshot of the orders at the back-end system. The function `findAll()` is called to retrieve these orders and documents from the JSONStore.
 - d. When `findAll()` is complete, the UI is updated.
- If network connectivity does not exist, the `findAll()` function is called directly to load records from the JSONStore, without loading them first from the IBM BPM server (see Example 6-28).

Example 6-28 Loading claimed orders, from IBM BPM server, then from JSONStore

```

//load claimed orders
function loadClaimedOrders(){

    //create accessor for JSON Store collection
    var accessor = WL.JSONStore.get('orders');

    //check on network status
    WL.Device.getNetworkInfo(function (networkInfo) {

        WL.Logger.warn(JSON.stringify(networkInfo));

        if(networkInfo.isNetworkConnected == 'true'){
            //there is network connectivity, load from handlers

            //clear store first
            accessor.clear().then(function () {

                //call adapter to get accepted orders
                var invocationData = {

```

```

        adapter : 'OrderAdapter',
        procedure : 'getClaimedOrders',
        parameters : [],
        compressResponse: true
    };
    return WL.Client.invokeProcedure(invocationData);

}).then(function(response){

    var data = response.invocationResult.orders;

    var changeOptions = {

        // Search fields in the input data that make a document unique
        replaceCriteria : ['taskId'],

        // Data that does not exist in the Collection will be added,
default false.
        addNew : true,

        // Mark data as dirty (true = yes, false = no), default false.
        markDirty : false
    };

    //add orders to JSON Store
    return accessor.change(data, changeOptions);

}).then(function (amountOfDataChanged) {

    //Load orders from JSON Store
    return accessor.findAll();

}).then(function (response) {

    //Update Ui with the accepted orders
    return updateClaimedOrdersUI(response);

}).fail(function (errorObject) {
    WL.Logger.warn(JSON.stringify(errorObject));
});
}else{
    WL.SimpleDialog.show("Alert", "No internet connection. Working offline!",
[ {text: "OK", handler: null}], null);

    //There is no network connectivity, load directly from JSON Store

    accessor.findAll().then(function (response) {

        //Update UI
        return updateClaimedOrdersUI(response);

    }).fail(function (errorObject) {
        WL.Logger.warn(JSON.stringify(errorObject));
    });
};
};

```

```
    });  
}
```

jQuery promises are used in this code. A detailed description is as follows:

1. Check for network connectivity is done using the API `WL.Device.getNetworkInfo()`. This is an asynchronous function. When it is complete, a check is done whether a network connection exists or not using the attribute `isNetworkConnected`.
2. If network connection exists:
 - a. Collection is cleared using the API `clear()`.
 - b. The adapter is called to get all the claimed orders using the API `WL.Client.invokeProcedure()`.
 - c. When the adapter call is complete, the data in the JSONStore is updated using the API `change()` with the following parameters:
 - `addNew:true` To add any new records. In this case all records will be treated as new since the collection was cleared.
 - `markDirty:false` Since all the records are just retrieved from the adapter and are in sync with the back-end system, they should not be marked as dirty.
 - d. When the `change()` call is successful, the JSONStore is updated. The API `findAll()` is used to read all the orders from the JSONStore and update the UI.
3. If no network connection exists, data is loaded directly from the JSONStore using the API `findAll()`.

Example 6-29 shows the update to the list of claimed orders.

Example 6-29 Updating claimed orders list UI

```
//Update claimed orders page UI (claimed orders)  
function updateClaimedOrdersUI(orders){  
  
    $("#claimed-orders").empty();  
    WL.Logger.warn("orders = "+JSON.stringify(orders));  
    for ( var i = 0; i < orders.length; i++) {  
        var li = document.createElement("li");  
        li.innerHTML =  
            "<a href='javascript:goToClaimedOrderDetailsPage(\"" +  
orders[i].json.orderId + "\", " + orders[i].json.taskId + ")'>" +  
            "<h2>" + orders[i].json.title + "</h2>" +  
            "<div>" + orders[i].json.streetAddress + ", " +  
orders[i].json.suburb + "</div>" +  
            "<div>" + orders[i].json.datetime + "</div>" +  
            "</a>";  
  
        $("#claimed-orders").append(li);  
  
    };  
  
    $("#claimed-orders").listview("refresh");  
}
```

Pushing data

Pushing data from the mobile device to the adapter and consequently to the IBM BPM server happens only when the field technician updates the status of the order. The general flow is as follows:

1. The document of the order is retrieved from the JSONStore.
2. Order status is updated in the retrieved order JSON variable.
3. Using the updated JSON document, the old document in the JSONStore is replaced with the new one.
4. The document is marked as dirty.
5. Dirty documents are pushed to the adapter using the adapter function `updateOrder()`.
6. If no network connection exists, the call to the adapter fails, but the changed order is stored in JSONStore to be retrieved later.

Example 6-30 shows the code in the function `updateShownOrder()` for updating the displayed order.

Example 6-30 Updating an order

```
//update status of an order, process is updated in JSON Store first then updated
in BPM
function updateShownOrder(status){
    var orderId = $("#page-claimed-order-details #order-order-id").text();

    var taskId = $("#page-claimed-order-details #order-task-id").text();

    var jsonId = Number($("#page-claimed-order-details #order-json-id").text());

    // Documents will be located with their '_id' field
    // and replaced with the data in the 'json' field.

    var collectionName = 'orders';

    var options = {

        limit: 20,

        offset: 0,

        filter: ['_id', 'json'],

    };

    var query = {_id: jsonId};

    var options = {
        exact: true,
        limit: 1
    };

    WL.JSONStore.get(collectionName).find(query, options).then(function (orders) {

        orders[0].json.businessStatus = status;
```



```

var options = {
    // Mark data as dirty (true = yes, false = no), default true.
    markDirty: true
};

//replace the old record with the new updated one
return WL.JSONStore.get(collectionName).replace(orders, options);

}).then(function () {

    //returns all dirty documents
    return WL.JSONStore.get(collectionName).getAllDirty();

}).then(function (arrayOfDirtyDocuments) {

    //call adapters to update BPM with the list of changed orders
    dirtyDocs = arrayOfDirtyDocuments;

    var invocationData = {
        adapter : 'OrderAdapter',
        procedure : 'updateOrder',
        parameters : [taskId, orderId, status],
        compressResponse: true
    };
    return WL.Client.invokeProcedure(invocationData);

}).then(function (response) {

    if(response.status == 200){
        WL.SimpleDialog.show("Alert", "Operation executed successfully!", [{text:
"OK", handler: null}], null);
        //since backend is updated, these changed orders are not dirty anymore
        return WL.JSONStore.get(collectionName).markClean(dirtyDocs);
    }
})

.then(function (numberOfDocumentsMarkedClean) {
    WL.Logger.warn(numberOfDocumentsMarkedClean);

}).fail(function (errorObject) {
    WL.Logger.warn(JSON.stringify(errorObject));
    WL.SimpleDialog.show("Alert", "Unable to update order, order updated
locally!", [{text: "OK", handler: null}], null);
});
}

```

Use the following steps to update the JSONStore:

1. The document representing the order is retrieved from the JSONStore using the find() API.
2. The status of order is updated in the retrieved document variable.

3. The old document in the JSONStore is replaced with the new one using the `replace()` API.
4. Document is marked as dirty. Note that the flag `markDirty:true` causes the `replace()` API to mark the document automatically as dirty.

Once the replace operation is successful, the following sequence takes place to push the data to the adapter:

1. All dirty documents are retrieved from the JSONStore using the `getAllDirty()` API.
2. Function `updateOrder()` in the adapter is invoked, sending the dirty document information (the updated order) as parameters.
3. Once the adapter call is successful, the JSONStore is again synchronized with the back-end system, and the document is marked as clean using the `markClean()` API.

6.4.7 Push notifications support

In this scenario, Tag-based Push Notification is the push notification type used. Tag-based is used to notify devices subscribed to a specific tag. IBM BPM calls the PushAdapter with the parameters needed, to send notification to the field technicians group subscribed to a specific tag to notify them about the arrival of a new order. When the field technicians receive the notification on their devices, they can go to the Field Service app inbox to check the new order.

The tag-based notification implementation includes:

- ▶ Adding the tags to the `application-descriptor.xml` file.
- ▶ Subscribing to a tag.
- ▶ Creating the server-side push notification adapter.
- ▶ Configuring the client certificate.

Adding tags to the application-descriptor.xml file

In the `application-descriptor.xml` file, add a `<tags>` element to specify the tags that you want for the application. Example 6-31 shows two tags that have been added: `FSTeamTag1` and `FSTeamTag2`

Example 6-31 Tags in application-descriptor.xml

```
<tags>
  <tag>
    <name>FSTeamTag1</name>
    <description>Field Service Team 1</description>
  </tag>
  <tag>
    <name>FSTeamTag2</name>
    <description>Field Service Team 2</description>
  </tag>
</tags>
```

Subscribing to a tag

To receive notifications that are targeted to a particular tag, the application subscribes to the tags that have been defined. To set up tag subscriptions, use methods of the `WL.Client.Push` class. The procedure is as follows:

- ▶ When the application first connects to the Worklight Server from the device, the device registers with a push service mediator and obtains a device token. This process is done automatically by IBM Worklight.
- ▶ When the token is obtained, the `onReadyToSubscribe` callback method that is defined in the application is notified that a device is ready to subscribe to push notifications.
- ▶ After the `onReadyToSubscribe` callback is notified, the application subscribes to a tag by calling the `subscribeTag` method.

Example 6-32 shows that the tag `FSTeamTag1` is used as the tag to be subscribed to. Note that this tag must be defined in the `application-descriptor.xml` file as described in “Adding tags to the application-descriptor.xml file” on page 154.

A callback function needs to be defined to handle the received notifications. This call back function is registered after the subscription to notifications is successful in the function `doSubscribeTagSuccess`. The implementation in Example 6-32 shows two alerts that display the contents of the notifications.

Example 6-32 Subscribing to a tag from the client side, main.js

```
...
if (WL.Client.Push) {
    WL.Client.Push.onReadyToSubscribe = function() {
        WL.Logger.warn("onReadyToSubscribe");

        doSubscribeToTag("FSTeamTag1");
    };
}

function doSubscribeToTag(tagName){
    WL.Client.Push.subscribeTag(tagName , {
        onSuccess: doSubscribeTagSuccess,
        onFailure: doSubscribeTagFailure
    });
}

function doSubscribeTagSuccess() {
    WL.Logger.warn("Tag Push Notification Subscribe Success");

    WL.Client.Push.onMessage = function (props, payload) {
        alert("Provider notification data: " + JSON.stringify(props));
        alert("Application notification data: " + JSON.stringify(payload));
    }
}

function doSubscribeTagFailure() {
    WL.Logger.warn("Tag Push Notification Subscribe Failure");
}
...
```

Creating the server-side push notification adapter

In order for IBM BPM to send notifications to the users subscribed to a tag, an adapter must be created to send the notification using the method `WL.Server.sendMessage`.

The push notification adapter is an HTTP adapter; in this example it is called `PushAdapter`. Example 6-33 shows the `PushAdapter.xml` file where the `submitNotification` procedure is exposed with `securityTest="wl_unprotected"`, to call the adapter from IBM BPM without authentication.

Example 6-33 Exposing PushAdapter procedure in PushAdapter.xml

```
<procedure name="submitNotification" securityTest="wl_unprotected"/>
```

The implementation of the `submitNotification` procedure is shown in Example 6-34. To send tag-based notifications:

- ▶ Use the `WL.Server.sendMessage` method. The `appName` and `options` parameters are mandatory.
- ▶ Specify the `tagNames` as an array in the `options.target.tagNames` object. In Example 6-34, the `tagNames` sent in the procedure are in a loop to fill the `options.target.tagNames` array.
- ▶ Specify the message to be sent as text in `options.message.alert`.

Example 6-34 Sending tag-based notification implementation details, PushAdapter-impl.js

```
function submitNotification(appName, tagNames, message){
    var options = {};

    options.message = {};
    options.message.alert = message;
    options.target = {};
    options.target.tagNames = [];
    for(var i=0; i<tagNames.length; i++){
        options.target.tagNames.push(tagNames[i]);
    }
    var delayTimeOut = WL.Server.sendMessage(appName, options);
    WL.Logger.warn("notificationTimeout"+delayTimeOut);
}
```

Google Cloud Messaging (GCM) API keys configuration

The mobile application must have a server ID and key to receive push notification using Google Cloud Messaging platform.

Follow the instructions to create GCM API keys and add the keys to the `application-descriptor.xml` as described in *Setting up push notifications for Android* in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/api/content/n1/en/SSZH4A_6.2.0/com.ibm.worklight.dev.doc/devref/t_setting_up_push_notification_android.html

The `pushSender` element is where the key and `senderId` information are specified.

Example 6-35 on page 157 shows the push notification configuration for GCM.

Example 6-35 Environment configuration in application-descriptor.xml

```
<android version="1.0">
  <worklightSettings include="true"/>
  <pushSender key="KEY_XXXXX" senderId="SENDER_ID_XXXXX"/>
  <security>
    <encryptWebResources enabled="false"/>
    <testWebResourcesChecksum enabled="false" ignoreFileExtensions="png, jpg,
jpeg, gif, mp4, mp3"/>
    <publicSigningKey/>
    <packageName/>
  </security>
</android>
```

PushAdapter URL to be called by IBM BPM

Now, IBM BPM can invoke the `submitNotification` procedure, using the following URL, to be able to send notifications to the users subscribed to a specific Tag(s).

`http://<wl-server-ip>:<wl-server-port>/<project-context-root>/invoke?adapter=PushAdapter&procedure=submitNotification¶meters=['<appName>', '<[tagNames]>', '<message>']`

where:

- ▶ `wl-server-ip`: is the Worklight Server domain name or IP address.
- ▶ `wl-server-port`: is the Worklight Server port.
- ▶ `project-context-root`: is the context root of the Worklight project.
- ▶ `appName`: is the application ID.
- ▶ `tagNames`: are the tags array to send the notification to.
- ▶ `message`: is the message text to be sent.

For Example:

`http://10.12.6.27:9084/WL_BPM_Project/invoke?adapter=PushAdapter&procedure=submitNotification¶meters=["FieldService", ["FSTeamTag1", "FSTeamTag2"], "New+Order+Received"]`

6.4.8 Geolocation support

Geolocation is used in this scenario to capture the location of the field technicians and track their location during the work day. This location information is used for various purposes including validation of the mileage reported by the field technician, field technician group optimization based on where the work is done, and so on.

The implementation of this use case includes:

- ▶ Start the location acquisition when the technician logs in and define the policy and triggers for the acquisition.
- ▶ Create a server-side adapter to handle the event generated when the position changes.

Start the location acquisition

Starting the acquisition of the geolocation is performed in `main.js` in the function `wlCommonInit()` when the application starts as shown in Example 6-36 on page 158.

```
function wlCommonInit(){
...
...
...
    startGeoAcquisition();
...
...
...
...
}
```

The following properties for the policy are set:

- The trigger defined is the `PositionChange` trigger with a callback function that transmits an event object to the server by calling the `WL.Client.transmitEvent` sending the event object `{event : "technicianMoved" , userId: <logged in userId> }`.

The failure callback for the `startAcquisition` API is `alertOnGeoAcquisitionErr`, which alerts the user that there is an error with an error message.

Example 6-37 Starting the acquisition if geolocation: Implementation

158 Extending IBM Business Process Manager to the Mobile Enterprise with IBM Worklight

```

        minChangeDistance : 1000,
        callback : function(deviceContext) {
            WL.Client.transmitEvent({
                event : 'technicianMoved',
                userId : WL.Client
                    .getLoginName("FieldServiceAuthRealm")
            }, true);
        }
    }
}
};

WL.Device.startAcquisition({
    Geo : geoPolicy
}, triggers, {
    Geo : alertOnGeoAcquisitionErr
});
}, function(geoErr) {
    alertOnGeoAcquisitionErr(geoErr);
}, geoPolicy);
}

```

Adding permission to the Android project

For the application to be able to acquire location data from the device, certain permissions need to be added in the generated Android project. For Android, all permissions must be listed in the application metadata file; the user confirms approval on these required permissions when installing the application.

Perform the following steps:

1. Navigate to the generated Android environment project folder
/FieldService/android/native
2. Open the file **AndroidManifest.xml**
3. Go to the tab **AndroidManifest.xml** to open the text editor of the file.
4. Add the following two permissions to the end of uses-permission list:
`<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>`
`<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>`
5. The top section of the file including the permissions should look like the one shown in Example 6-38.

Example 6-38 Adding location permissions in AndroidManifest.xml

```

<?xml version="1.0" encoding="UTF-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.FieldService" android:versionCode="1" android:versionName="1.0">
    <uses-sdk android:minSdkVersion="10" android:targetSdkVersion="19"/>
    <supports-screens android:smallScreens="false" android:normalScreens="true"
android:largeScreens="false"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>

```

```

<!-- Push permissions -->
<permission android:name="com.FieldService.permission.C2D_MESSAGE"
android:protectionLevel="signature"/>
<uses-permission android:name="com.FieldService.permission.C2D_MESSAGE"/>
<uses-permission
android:name="com.google.android.c2dm.permission.RECEIVE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.USE_CREDENTIALS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>

```

Create the location changed event handler adapter

After starting the acquisition of the geolocation, the event is sent to the server-side adapter, LocationAdapter, that filters on technicianMoved in the sent event and reports the new position.

The LocationAdapter is an HTTP adapter with the following implementation details:

- ▶ **createEventHandler:**

The WL.Server.createEventHandler JavaScript server-side API must be called to create an event handler. It filters the events transmitted and listens on the event technicianMoved only. When the filtered event is transmitted, the function notifyPosition is called.

- ▶ **setEventHandlers:**

After the creation of the event handler, WL.Server.setEventHandlers is called with the name of the event handler sent as an argument to the method.

- ▶ **notifyPosition:**

This function handles the event. In this function, integration with IBM Bluemix takes place to store the coordinates of the user on IBM Bluemix data services. The details are described in 6.4.9, “Storing location information in IBM Bluemix” on page 160.

6.4.9 Storing location information in IBM Bluemix

IBM Bluemix services are exposed as REST services. The coordinates of the device are stored in IBM Bluemix with the logged in user name and the time stamp.

The LocationAdapter connects IBM Bluemix over HTTP. Example 6-39 shows the configuration of the adapter. Note that the domain is the domain name of the IBM Bluemix application, application_name.mybluemix.net.

Example 6-39 LocationAdapter.xml

```

<wl:adapter name="LocationAdapter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wl="http://www.worklight.com/integration"
  xmlns:http="http://www.worklight.com/integration/http">

  <displayName>LocationAdapter</displayName>
  <description>LocationAdapter</description>
  <connectivity>
    <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
      <protocol>http</protocol>
    
```



```

        <domain>w402customermdm.mybluemix.net</domain>
        <port>80</port>
        </connectionPolicy>
        <loadConstraints maxConcurrentConnectionsPerNode="2" />
    </connectivity>
</wl:adapter>

```

The user name, coordinates, and time stamp are extracted from the event by the function `notifyPosition()` and sent to IBM Bluemix. See Example 6-40.

Example 6-40 Storing location data in Bluemix when location changes

```

function notifyPosition(event) {
    var date = new Date(event.deviceContext.Geo.timestamp);

    var contentJSON = {
        userId:event.userId,
        longitude:event.deviceContext.Geo.coords.longitude,
        latitude:event.deviceContext.Geo.coords.latitude,
        date: date.getDate() + '/' + date.getMonth() + '/' + date.getYear(),
        time: date.getHours() + ':' + date.getMinutes()
    };

    var invocationData = {
        method : 'post',
        path
: '/w402customermdm/v1/apps/9db7d6f0-d779-4a3e-8b8f-1d626bb0c0f0/location',
        body:{
            contentType:'application/json',
            content:JSON.stringify(contentJSON)
        }
    };

    var response = WL.Server.invokeHttp(invocationData);
    WL.Logger.warn("Response: " + JSON.stringify(response));
    return response;
}

WL.Server.setEventHandlers([
    WL.Server.createEventHandler({ event : 'technicianMoved'}, notifyPosition),
]);

```

The variable `contentJSON` is filled with all the information extracted from the event. Date and time are extracted from the timestamp sent in the event. This JSON is sent as the body content in the post HTTP request directed to IBM Bluemix. Example 6-41 shows an example of the JSON structure of the location event.

Example 6-41 Location changed event structure

```

{
    "event":"technicianMoved",
    "userId":"bdaniel",
    "deviceContext":{
        "lastModified":1412289196018,
        "timezoneOffset":-120,
        "Geo":{
            "timestamp":1412289196018,

```

```

        "coords":{
            "speed":78.08,
            "altitude":8900.77,
            "longitude":2.2935178725093985,
            "latitude":48.85800178087322,
            "accuracy":48.9,
            "altitudeAccuracy":1,
            "heading":250
        }
    }
}

```

Example 6-42 shows an example of the JSON to be sent as the body content of the IBM Bluemix request.

Example 6-42 Example of JSON body content of a Bluemix request

```

{
    "userId":"bdaniel",
    "longitude":2.2935178725093985,
    "latitude":48.85800178087322,
    "date":"3/9/2014",
    "time":"0:33"
}

```

6.5 IBM BPM processes supporting this scenario

This section describes the IBM BPM process and features used in this scenario and the corresponding artifacts to implement them.

Figure 6-14 on page 163 shows the business process in this scenario. The business process comprises three swim lanes representing the process participants and a sequence of activities and tasks that form the New Order Installation process.

- The Call Centre swim lane contains the activities performed by the staff at the call center. These activities are human work flow tasks that require the interaction between the staff and the IBM BPM user interface, served by the Process Portal.

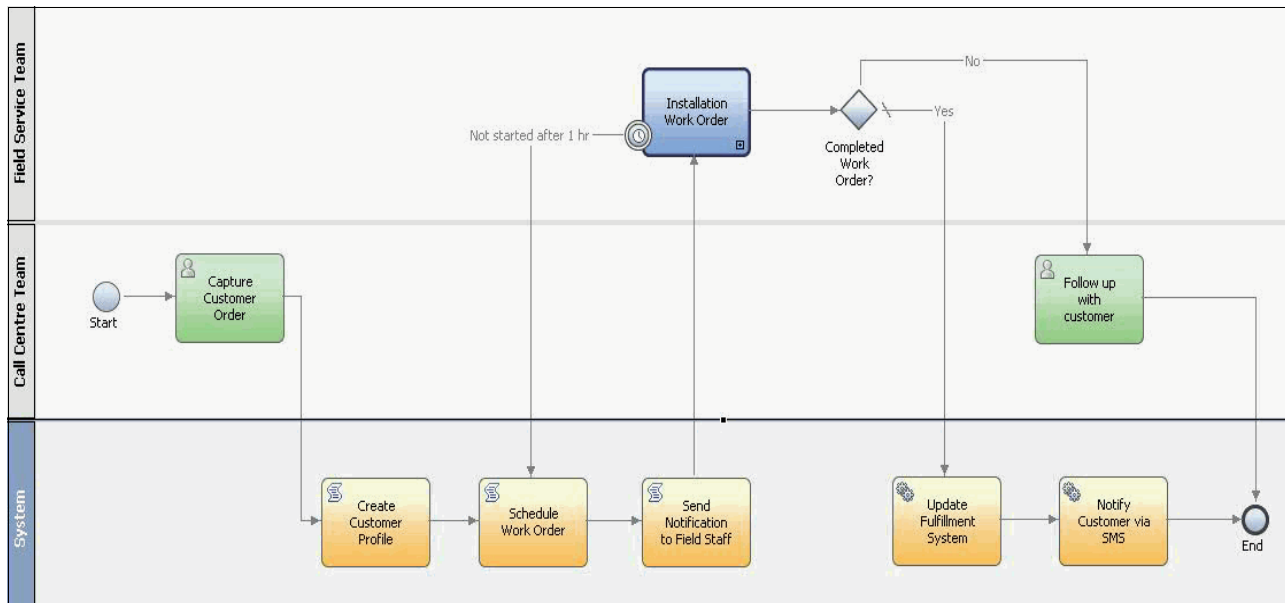


Figure 6-14 Scenario 1 business process

- The Field Service Team swim lane contains activities performed by field technicians. In this scenario there is a single activity that is implemented by a linked process, which is the Installation Work Order business process.

The Installation Work Order process shown in Figure 6-15 comprises a single human task, Perform Installation. A human task is typically used to drive a human interaction via the IBM BPM user interface in Process Portal. However in this scenario, the interaction with this task is from the mobile app via the IBM BPM REST API. As such, a coach form is not required to be created as part of the Perform Installation task.

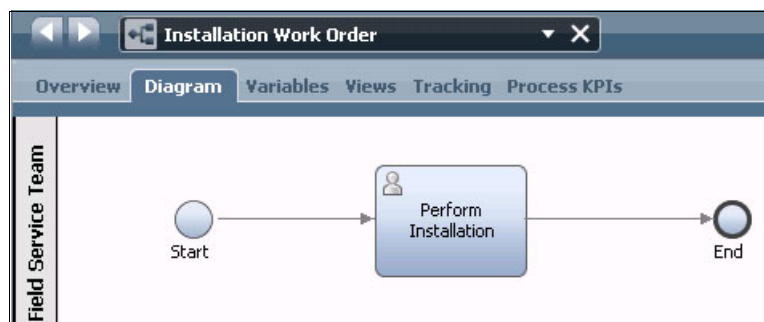


Figure 6-15 Installation Work Order linked process

- The System swim lane contains all the automated system tasks that include either interactions with the simulated back-end systems or automated processing.

In this scenario, the business process is triggered by the call center staff member upon receiving a call from a customer. The following steps describe the process:

1. The call center staff member receives a customer call with a request for a new installation.
2. The call center team member starts the New Order Installation process via the Process Portal. The process is started by clicking the New Order Installation link listed under the Launch section of the My Work page (see Figure 6-16 on page 164).

3. The first activity in the process is Capture Customer Order. As this task is assigned to the Call Centre Team swim lane, this task will appear under My Tasks for all call center staff members, as shown in Figure 6-16.

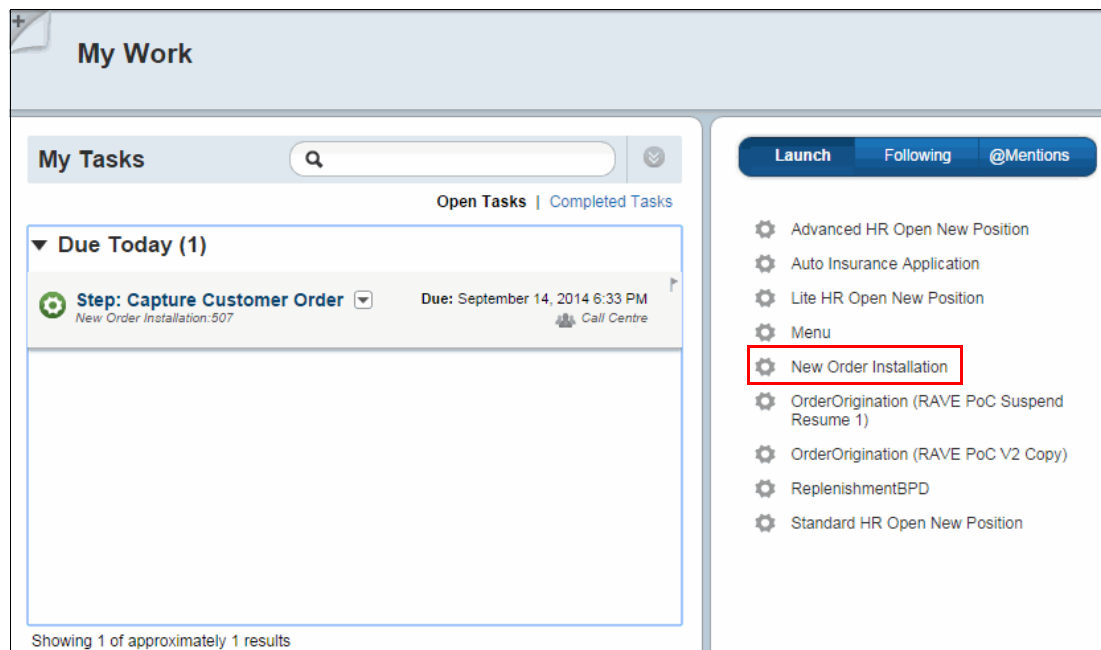


Figure 6-16 Process Portal - My Work - Capture Customer Order task

4. Clicking the **Capture Customer Order** entry under the Due Today section prompts the call center staff member for whether the task should be claimed.
5. Clicking the **Claim Task** button assigns the task to the user and presents the Capture Customer Order coach form within the human task.
6. The call center staff member enters the details of the customer order over the next sequence of coach forms shown in Figure 6-17. The Capture Customer Order is implemented as a client-side human service, new feature in IBM BPM 8.5.5, and includes a responsive design.

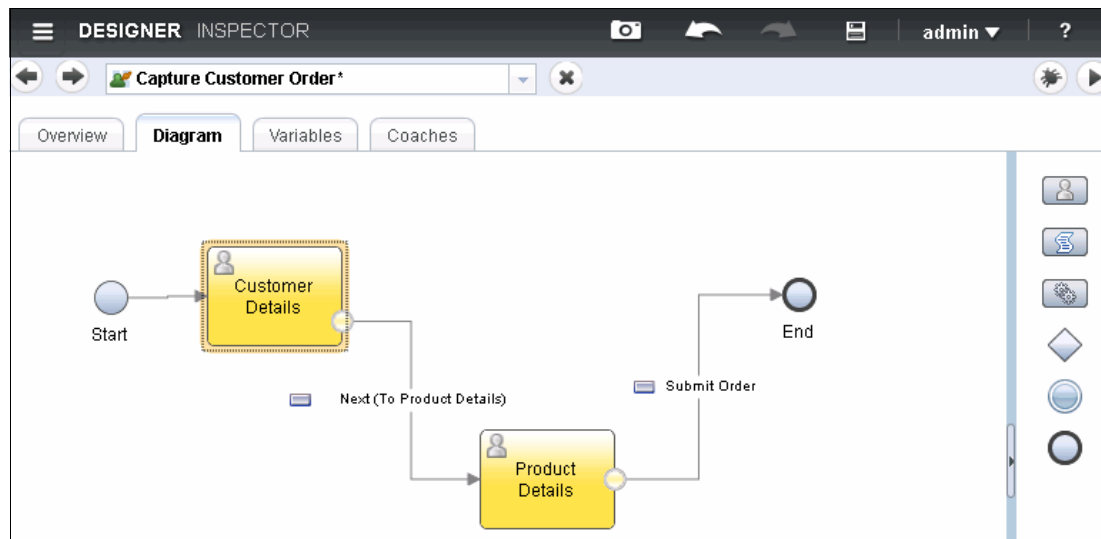


Figure 6-17 Capture Customer Order client-side human service

The Customer Order is captured using two windows, as shown in Figure 6-18 and Figure 6-19.

Figure 6-18 Capture Customer Order coach form (1 of 2)

Figure 6-19 Capture Customer Order coach form (2 of 2)

7. Once capturing the customer order is completed, the New Order Installation process proceeds to the next task, which is the Create Customer Profile activity. This activity is an automated system that represents the step of a customer profile being created in a Customer Master Data Management (MDM) system. For this scenario, the Customer MDM is simulated using the Bluemix Mobile Data service (MDS), NoSQL-based document repository accessible via an HTTP API.

8. The process continues to the Schedule Work Order activity that represents the step of calling a Work Order Management system to schedule and create an associated work order for the customer order.
9. Once a work order has been created and assigned to a team, the process then proceeds to the Send Notification to Field Staff activity that sends a push notification to the field technicians team via IBM Worklight. The Send Notification to Field Staff activity uses the IBM Worklight push notifications API to send the notification.
10. The process then moves to the Installation Work Order (IWO) activity that is a linked process. The IWO linked process starts automatically.
11. Within the IWO process, the process proceeds immediately to the Perform Installation activity where the process halts as it is a human task.
12. At this point, the Perform Installation task has been assigned to the field service team. From a business process perspective, the field service team sees this task as a Work Order in the Field Service app.
13. A team member may then choose to accept the Work Order, in which case, from an IBM BPM perspective, performs a task claim action.
14. As a field technician staff member of the field service team performs the work described by the Work Order, the field technician can update the status of the Work Order using the Field Service app.
15. The Field Service app sends update status requests to IBM Worklight, which in turn uses the IBM BPM REST API to update the Perform Installation task business data that holds the Work Order state.
16. The BPM process remains at the Perform Installation task until the field technician completes the Work Order within the Field Service app. While performing the work, the field technician can update the Field Service app status, which in turn updates the Perform Installation task business data.

Suspend/Resume pattern

While in this scenario only a single device is assumed to be used, the field technician could start a work order on one device, for example, a smartphone, and then switch to continue the activities of the work using an alternative device, for example, a tablet (*suspend*). As the status updates are persisted within the IBM BPM Perform Installation task when the Field Service app is launched on the alternative device (tablet), it can read the business data from the IBM BPM Perform Installation task and the field technician can *resume* work from that point.

17. The field technician could use a different device to retrieve the data from the active Work Order and continue updating it from there. This approach is often called *suspend/resume*. The IBM BPM process remains at the Perform Installation task until the field technician completes the Work Order within the Field Service app.
18. Once the Installation Work Order activity is completed by the field technician, the process checks the Work Order status to determine whether it was completed successfully or requires a follow-up with the customer.
19. If successful, the process moves to the next step where the fulfillment system is called. Once again Bluemix is used in this scenario to simulate a real fulfillment system.
20. Finally, the customer is sent a Short Message Service (SMS) to confirm their service has been provisioned. (This activity is not actually implemented in the scenario but is here for completeness of the example).

6.6 Building the business process

This section provides the details about how to build the business process described in 6.5, “IBM BPM processes supporting this scenario” on page 162.

Note: See Appendix A, “Samples included with this book” on page 307 for information about the samples provided with this IBM Redbooks publication.

6.6.1 Creating the business process definition

Now that we have walked through the business process, let us begin building it using IBM Process Designer.

Perform the following steps to build the business process definition:

1. Launch IBM Process Designer and create a new process application.
2. Within the Designer tab, create a new business process definition (BPD) and name it **New Order Installation**.
3. Rename the **Team** swim lane to **Call Centre Team**.
4. Select the **Call Centre Team** swim lane and under the **Properties** → **General** section, click **New** next to the Default Lane Team: attribute.
5. Create a new team that represents the call center staff and add some standard members to the team.
6. The Default Lane Team should now be set to **Call Centre**, as shown in Figure 6-20.



Figure 6-20 Call Centre Default Lane Team

Note: You may want the Capture Customer Order activity to be automatically assigned to the user that starts the process from the Process Portal. This is possible by setting the Assign To property to Lane and the User Distribution property to Last User in the User Task Assignments properties pane. When the user clicks the New Order Installation process link in the Process Portal Launch pane, the Capture Customer Order coach form appears automatically in the Process Portal work pane.

Figure 6-21 on page 168 shows the Task Assignments properties pane for the Capture Customer Order activity.

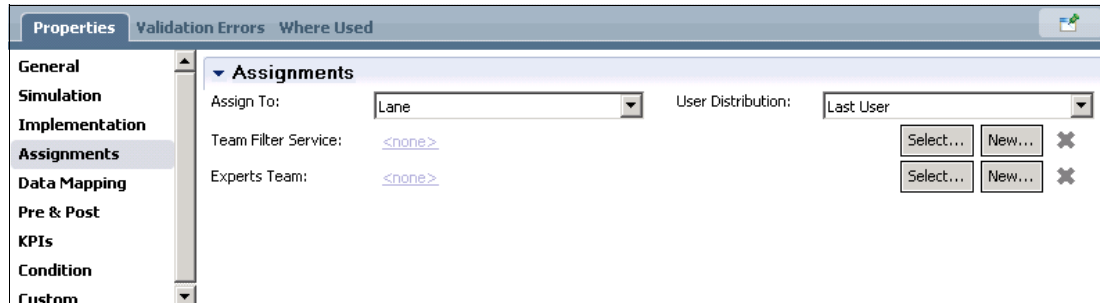


Figure 6-21 Task Assignments for Capture Customer Order

7. Add another swim lane above the Call Centre swim lane and name it **Field Service Team**.
8. As with the Call Centre team, add a Field Service Team and set the Field Service swim lane Default Lane Team to the Field Service Team. This is an important step as tasks created within this swim lane must be allocated to a team from which an individual team member will eventually claim the task to perform the work. Only one team member may claim and work with a task at any time.
9. Add the remaining process elements to their respective swim lanes as shown in Figure 6-22. Their implementation is discussed in Section 6.6.2, “Define the business objects and variables” on page 168.

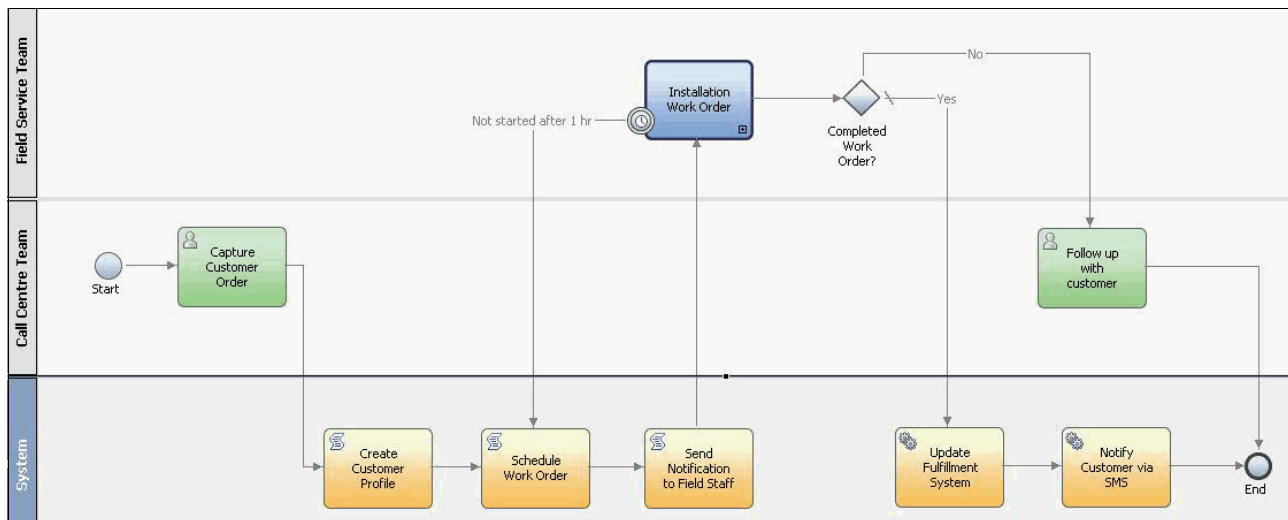


Figure 6-22 Scenario 1 business process definition (BPD)

6.6.2 Define the business objects and variables

Now that the key elements of the process have been laid out, it is time to define the variables that will be used within the BPD. Perform the following steps:

1. Define the business objects under Data in the Library section as described in Table 6-1.

Table 6-1 Business objects

Name	Parameter	Type	Comment
CustomerOrderBO	contractTerm	Integer	12 = 12 months 24 = 24 months
	dateOfInstall	String	

Name	Parameter	Type	Comment
	timeOfInstall	String	
	CustomerDetails	CustomerBO	
	ProductDetails	ProductBO	
CustomerBO	CustomerID	String	
	Firstname	String	
	Lastname	String	
	DateOfBirth	String	
	StreetAddress	String	
	Suburb	String	
	PostCode	String	
	MobileNumber	String	
	EmailAddress	String	
ProductBO	productCode	String	Example: CBL
	cableModem	Integer	0 = None 1 = Standard 2 = With Wi-Fi
	cableSpeed	Integer	0 = Standard 1 = High Speed
WorkOrderBO	orderId	String	
	title	String	
	description	String	
	tasks	String	
	dataRequired	String	
	timeRequired	String	
	location	LocationBO	
LocationBO	streetAddress	String	
	suburb	String	
	postCode	String	
WorkOrderReportBO	orderId	String	
	status	String	
	summary	String	

2. Add the private variables as defined in Table 6-2 on page 170.

Table 6-2 Private variables

Name	Type	Comment
customerOrder	CustomerOrderBO	Holds the customer order entered by the call center team member
workOrder	WorkOrderBO	Holds the work order received from the work order management system
workOrderReport	WorkOrderReportBO	Holds the current status and final completion details of the work completed by the Field Service team member when closing a work order

3. Right-click the **Capture Customer Order** activity and select **Activity Wizard**.
4. Select the **Create a client-side human service** option and then click **Next**.
5. Set the Business Process Variable Input and Output to **false** for everything except customerOrder Output, as in Figure 6-23.
6. Click **Finish**.

Setup parameters:		
Business Process Variable	Input	Output
customerOrder (CustomerOrderBO)	false	true
workOrder (WorkOrderBO)	false	false
workOrderReport (WorkOrderReportBO)	false	false

Figure 6-23 Customer Order Activity Wizard: Parameters

6.6.3 Implementing the Capture Customer Order coach forms

As part of the New Order Installation business process, the Call Centre team will use coach forms to capture the customer's order. This approach follows IBM BPM interaction pattern #1 IBM Process Portal defined in 4.4.1, "Pattern 1: IBM BPM Process Portal" on page 76. Perform the following steps to implement the Capture Customer Order coach forms:

1. In IBM Process Designer, double-click the **Capture Customer Order** activity. The Design Inspector opens in the default web browser.

Client-side human services is a new feature introduced in IBM BPM v8.5.5 that supports creating human services and coach forms using a standard browser.

2. Create the Human Task flow as shown in Figure 6-24.

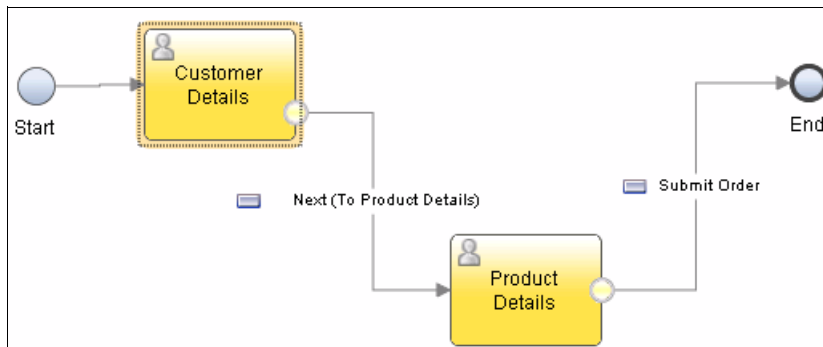


Figure 6-24 Capture Customer Order human service flow

3. Add the necessary fields to the Customer Details form.

Create the Customer Details container and drag the parameters from the CustomerDetails variable under the Variables section to the Customer Details container as shown in Figure 6-25.

The screenshot shows the Camunda Designer interface with the 'DESIGNER INSPECTOR' header. The main workspace displays a 'Coach' form titled 'Customer Details'. The form has a blue header bar with the title. Below the header, there are eight input fields, each with a label: 'Firstname', 'Lastname', 'Date Of Birth', 'Email Address', 'Mobile Number', 'Street Address', 'Suburb', and 'Post Code'. A blue 'Next' button is located at the bottom left of the form. On the left side, there is a sidebar with two items: 'Customer Details' (selected) and 'Product Details'. On the right side, there is a sidebar with a search bar 'enter filter text' and a 'Variables' section. The 'Variables' section lists several variables: 'contractTerm', 'dateOfInstall', 'timeOfInstall', 'CustomerData', 'CustomerId', 'Firstname', 'Lastname', 'DateOfBirth', 'StreetAddress', 'Suburb', 'PostCode', and 'MobileNumber'. The 'CustomerData' variable is expanded, showing its sub-variables.

Figure 6-25 Customer Details coach form

4. Add the necessary fields to the Product Details Coach form.

- Create two container sections: Service Requested and Order Details
- Drag over the matching parameters from the Variables section as shown in Figure 6-26 on page 172.

Figure 6-26 Product Details Coach form

5. Go back to the New Order Installation BPD and select the **Capture Customer Order** activity. Under the Properties section, select Data Mapping and set the Output Mapping variable map from **customerOrder** to **tw.local.customerOrder**, as shown in Figure 6-27.

Figure 6-27 Capture Customer Order activity data mapping

6.6.4 Implementing the Installation Work Order linked process

The Installation Work Order Activity provides a container for the task that will be used by field technicians to retrieve and update work order data. The Installation Work Order Activity is implemented as a linked process due to its potential reuse in other processes.

Note: When you use a linked process as the implementation for an activity, you can use an advanced option in the implementation properties to supply a predefined variable to dynamically call at runtime one of many linked processes, depending on your needs. For more information, see *Calling a linked process dynamically* in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSFPJS_8.5.5/com.ibm.wbpm.wle.editor.doc/topics/using_nested_processes_A.html?lang=en

The Installation Work Order linked process comprises a single activity implemented as a human task (see Figure 6-28 on page 174).

The New Order Installation process is implemented as a standard business process definition (BPD); therefore, it continues to flow through each activity until it completes.

For the field technicians to participate in this process, the process must provide an opportunity for the Field Service team to retrieve the list of work orders assigned to the team, view, claim, and update the work order as part of the process. To enable this capability, the activity presenting the field service work order is implemented as a human task. This activity puts the process in a waiting state while the field service team has an opportunity to retrieve the work order details, review it, claim it (accept the work order), update the business data (work order state), and then finish (complete the work order).

While a user task typically exposes a coach form for direct human interaction via a browser user interface, in this case the user interface is provided by the Field Service app and the interaction is via the IBM BPM REST API. By using a human task, coach form task assignments are enabled to allow work orders to be allocated to a team from which a team member can claim a task (work order). The coach form can also be implemented to allow browser-based access to the work order task in addition to the mobile app. If a coach form is added, when the process runs and arrives at the user task-based activity, it will block at the first coach form. The mobile app can still use the IBM BPM REST API to retrieve and update business data and finish the task even though the user task workflow does not complete its flow.

If task assignments are not necessary and browser-based human interaction are not required, the Perform Installation activity can be implemented as an External Activity.

To create the Installation Work Order linked process, use the following steps:

1. Right-click the **Installation Work Order** activity and select **Activity Wizard** from the context menu.
2. Select **Linked Process** and then click **Next**.
3. Set customerOrder Input: **false** Output: **false**, workOrder Input: **true** Output: **false**, workOrderReport Input: **false** Output: **true**
4. Click **Finish**.
5. The Installation Work Order only requires a single task that can be accessed via the REST API:
 - a. Drag an activity from the palette and configure it as a human task as shown in Figure 6-28 on page 174. A coach form does not need to be configured in this scenario as this activity will only be accessed via the IBM BPM REST API as per IBM BPM interaction pattern 4 IBM BPM REST API, as defined in 4.4.4, “Pattern 4: IBM BPM REST API” on page 77.

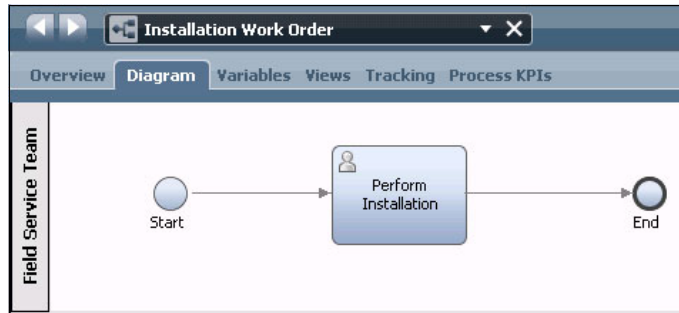


Figure 6-28 Installation Work Order linked process

- b. Set the input variable to **workOrder** and the output variable to **workOrderReport**. The Perform Installation flow is shown in Figure 6-29.

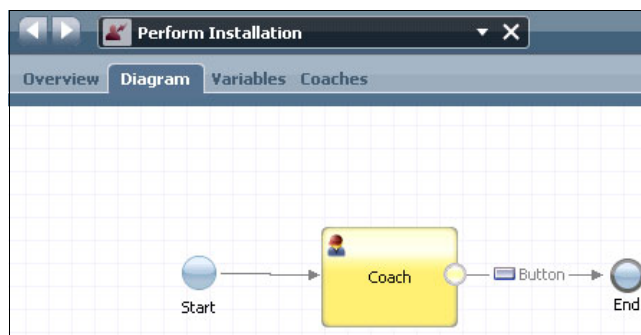


Figure 6-29 Perform Installation human task

Note: In this scenario, for this human task, the task assignment is done by Lane, whereby anyone who is a member of the field service team can claim this task.

Another alternative to assigning this task is to use a team assignment based on a team name returned from the work order scheduling system (not implemented in this scenario).

Figure 6-30 on page 175 shows the Assignment properties for the user task. Notice the following:

- ▶ A dynamically determined team name can be passed in a variable, in this case, `tw.local.workOrder.team`. This team name can be returned from the call to the work order scheduling system.
- ▶ A team filter service can be written to filter out certain users based on any criteria.
- ▶ A list of experts for this task can be dynamically determined by defining an *Experts* team.

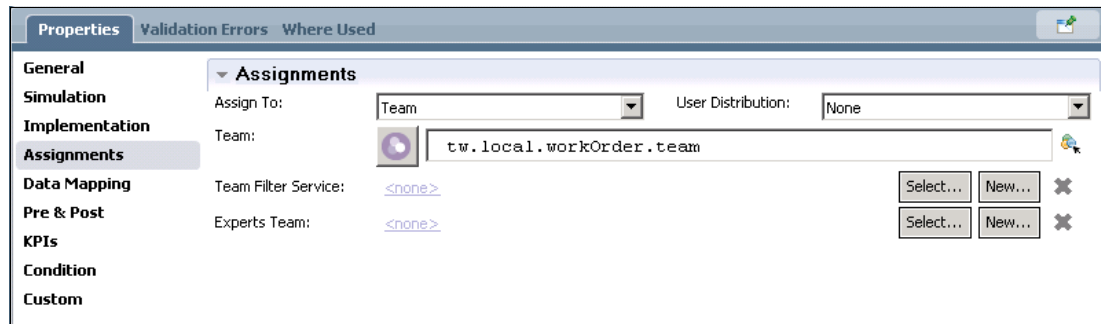


Figure 6-30 User task assignment properties

6.6.5 Implementing the Create Customer Profile system activity

The Create Customer Profile activity is an automated system activity that makes a call to an external customer Master Data Management (MDM) system to create a new instance of a customer. In this scenario, the customer MDM is represented by an IBM Bluemix Mobile Data service that accepts incoming JSON data objects.

Bluemix provides a cloud Platform as a Service (PaaS) environment that includes various application and data services. Bluemix Mobile Data service includes a REST API for performing create, read, update, delete operations on JSON objects within its repository. A typical MDM system returns a unique identifier for the customer profile that is created. This is simulated in this scenario by the Mobile Data service where it returns a unique identifier for the object being inserted.

There are other services within Bluemix that could be used to simulate an MDM service, such as Node.js. The Bluemix Mobile Data service provides a simple, easy to configure service without the need to code and lends itself to quick access and implementation.

Create the Bluemix Mobile Data service

Perform the following steps to create the Bluemix Mobile Data service:

1. Create an account in IBM Bluemix (<http://bluemix.net>). Bluemix offers a trial at no charge.
2. Under the Applications section, click the **CREATE AN APP** tile.
3. Bluemix offers a range of application templates to provide a starting point for development. Select the **Mobile Cloud** boilerplate as shown in Figure 6-31. This template comprises Node.js, Mobile Data service, and Mobile Security. Only the Mobile Data service is used in this scenario. For details about the API, see the Mobile Cloud Services REST API Documentation at the following link:

<https://mobile.ng.bluemix.net/mbaas-api>

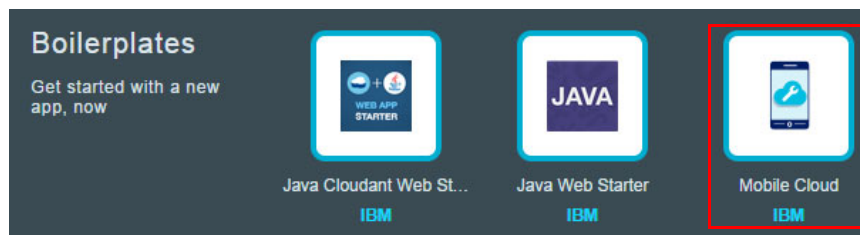


Figure 6-31 Bluemix application templates

4. Enter the name of the application, for example, **redbookapp8240** as shown in Figure 6-32.

Make a note of the host name; it must be unique because it will be accessible on the Internet.

Host mobile and web applications and accelerate development time of server side scripts by leveraging the mobile app template and SDK. Powered by SDK for Node.js™

[VIEW DOCS](#)

Pick a plan Monthly prices shown are for: [Australia](#)

Plan	Features	Price
✓ Default	Run one or more apps free for 30 days (375 GB-hours free, shared across Bring Your Buildpack and Ruby runtimes).	AU\$0.08 AUD/GB-Hour

Create an app:

Space: Redbook-W402

Name: redbook

Host: redbook

Domain:

Selected Plans:

SDK for Node.js™ Default

MAS Standard

Push Standard

MobileData Shared

CREATE

Figure 6-32 Bluemix Mobile Cloud boilerplate instance configuration window

5. Click **Create**.
6. Once the application is created, click **Mobile Options** and take note of the Mobile Application Identifier and Secret, as shown in Figure 6-33 on page 177. They will be used in section “Create the IBM BPM Bluemix integration service” on page 179.

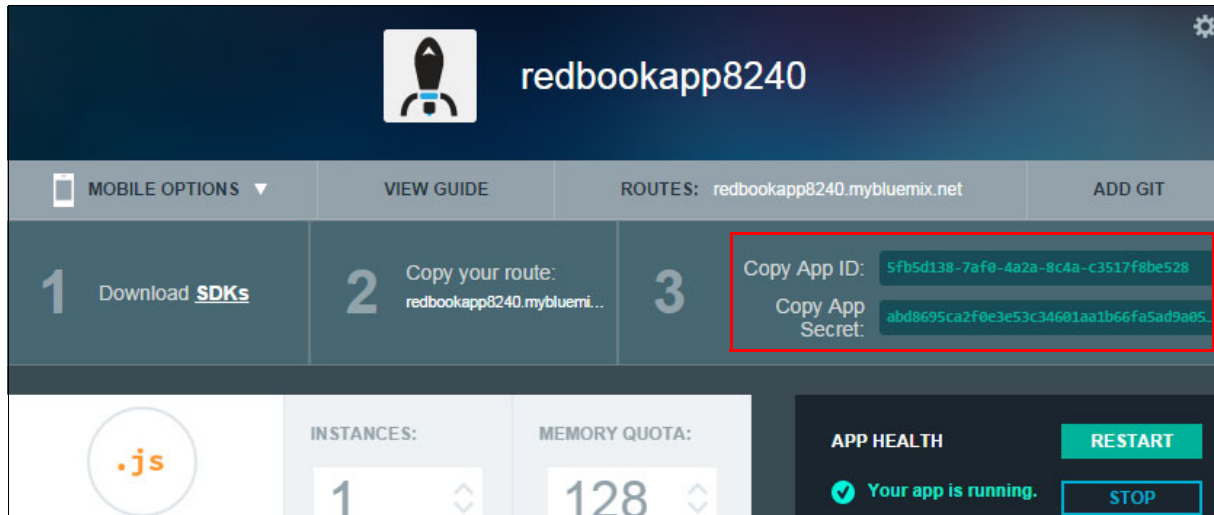


Figure 6-33 Mobile Cloud App ID and App Secret

7. Click the **Mobile Data Service** tile to open it.
8. Click the **Mobile Data** tab.
9. In a text editor, create the JSON file `CustomerProfile.json` as listed in Example 6-43. See “Samples compressed file content” on page 308 for information about how to locate this file in the samples.

Notice that the JSON file contains an array with one element. This is required by the Mobile Data service when importing a JSON file.

Example 6-43 `CustomerProfile.json`

```
[
  {
    "customerId": "239032095823532",
    "firstName": "John",
    "lastName": "Smith",
    "dateOfBirth": "01/01/2000",
    "streetAddress": "60 City Rd",
    "suburb": "Southbank",
    "postCode": "3008",
    "mobileNumber": "0444-444-44",
    "emailAddress": "john@smith.org"
  }
]
```

10. Save and drag the file onto the file import hotspot as shown in Figure 6-34 on page 178.

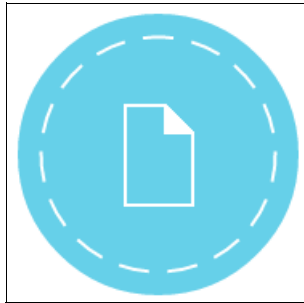


Figure 6-34 File Import hotspot

11. Dragging the file onto the dial triggers an import process that prompts for the creation of the class for the data being imported (see Figure 6-35).
12. Enter the Class Name **CustomerProfile** and click **Import**.

Import Data [Close]

You can import a wide variety of files, such as videos or JSON database files and then use the files with the services in your application.

Data File

CustomerProfile.json [Browse...]

Import to Class

Create a new Class [v]

Class Name

CustomerProfile

[Import] [Cancel]

Current Storage 37.22KB

Figure 6-35 Bluemix Mobile Data service Import Data form

13. Once the data has been imported, the Mobile Data service creates the class definition. After this step, you can select the record that you imported and deleted to avoid having the sample data returned once you deploy. The class definition remains even if you delete all the elements.
14. Repeat Step 9 on page 177 to Step 13 to create the Location class in the Mobile Data service using the file as listed in Example 6-44 on page 179. The Location class will be used by the Field Service app to report its GPS location via Worklight. Name the class Location in the Import Data form and use the JSON structure as shown in Example 6-44 on page 179. See “Samples compressed file content” on page 308 for information about how to locate this file in the samples.

```
[
  {
    "time": "String",
    "userId": "String",
    "longitude": "String",
    "latitude": "String",
    "date": "String"
  }
]
```

Create the IBM BPM Bluemix integration service

The IBM Business Process Manager product is available in two editions: Standard and Advanced. Although the Standard edition includes basic integration with SOAP-based services, it does not include integration with external REST APIs using JSON/HTTP. The IBM BPM Advanced edition supports additional integration capabilities using the underlying IBM Process Server engine and requires the use of IBM Integration Designer to implement integration adapters that can interface with external REST APIs. The Standard edition does not include the Advanced integration services component and so an interface to an external REST API must be implemented using a Java integration.

To create the Java integration that will be used to interface with Bluemix, a Java integrated development environment (IDE) is required. In this scenario, the Eclipse IDE for Java EE Developers is used to create the Java Integration module. It can be downloaded from the Eclipse website at <http://eclipse.org>.

The Bluemix Java integration adapter is implemented using a single Java class that leverages the Apache HTTP client library. It can be downloaded from *The Apache Software Foundation* at <http://hc.apache.org>.

Perform the following steps to implement the Java integration adapter that will be imported into IBM BPM:

1. Launch Eclipse.
2. Select **File** → **New** → **Java Project**.
3. Click **Next**.
4. Enter the project name: **BluemixMobileData**.
5. Click **Next**.
6. Select the **Libraries** tab.
7. Click **Add External JARs**.
8. Select the **httpclient-4.3.5.jar** and **httpcore-4.3.2.jar** files.
9. Click **Finish**.

The Eclipse IDE creates the Java project.

The next step is to write the Java class by performing the following steps:

1. In the Package Explorer view, right-click the **src** folder.
2. Select **Java Class**. The New Java Class dialog is displayed.
3. Enter the package name: **com.ibm.redbook.bluemix**.
4. Enter the class name: **BluemixMobileDataService**.

5. Click **Finish**.

6. Enter the source code as listed in Example 6-45.

The class includes a single public method, `submitCustomerProfile`, that will be called by the Create Customer Profile activity to submit the customer profile captured in the Capture Customer Order activity. The method takes a single parameter of type `String` that must be in JSON format. The method returns a `String` that is also in JSON format. Note the use of the Application ID and Secret. The Application ID is part of the URI and the Application Secret is set as an HTTP header.

Example 6-45 Bluemix Java integration service source code

```
package com.ibm.redbook.bluemix;

import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.apache.http.entity.StringEntity;

public class BluemixMobileDataService {

    public String submitCustomerProfile(String jsonString) throws Exception {

        String stringResponse;

        CloseableHttpClient httpClient = HttpClients.createDefault();

        try {

            HttpPost httpPost = new HttpPost("https://mobile.ng.bluemix.net:443/"
                + "data/rest/v1/apps/9db7d6f0-d779-4a3e-8b8f-1d626bb0c0f0/"
                + "uploads");
            httpPost.setHeader("IBM-Application-Secret", "f815370c6d53eaaa6a59c"
                + "e54c0569248ca125ed9");
            httpPost.setHeader("Content-type", "application/json");

            System.out.println("inputParameter = "+jsonString);
            httpPost.setEntity(new StringEntity(jsonString));

            CloseableHttpResponse response = httpClient.execute(httpPost);
            try {
                System.out.println(response.getStatusLine());
                HttpEntity entity = response.getEntity();

                stringResponse = EntityUtils.toString(entity);
                System.out.println("Content = "+ stringResponse);
                System.out.println("Parameter = " + jsonString);

                EntityUtils.consume(entity);
            } finally {
                response.close();
            }
        }
    }
}
```

```

    }

    } finally {
        System.out.println("Closing");
        httpClient.close();
    }

    System.out.println("stringResponse = " + stringResponse);
    return stringResponse;
}
}

```

After entering the code, export the project as a JAR file:

1. Right-click the project in Package Explorer and select **Export** from the context menu.
2. Select **JAR** file (type JAR in the filter to quickly find it).
3. Select an export destination.
4. Click **Finish**.

The JAR file can be imported into IBM BPM so that it may be used to call the Bluemix Mobile Data service by the New Order Installation business process.

Import the JAR file into IBM BPM:

1. In IBM Process Designer, click the + (plus) button next to the Files entry.
2. Select **Server File** from the context menu as shown in Figure 6-36.

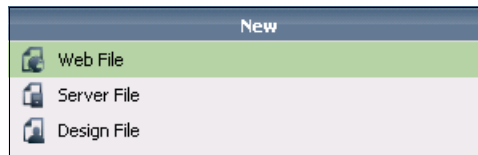


Figure 6-36 New File options

3. Select the exported JAR file.
4. Under the Common section of the Server File window ensure that the MIME Type is set to application/octet-stream.
5. The Apache HTTP Client libraries also must be added. Select the `httpClient` `library.jar` files and add them as Server Files with MIME Type set to application/octet-stream.
6. Click **Finish**.

Now that the JAR file has been imported, the Bluemix Mobile Data integration service can be created and configured to invoke the Java method. Perform the following steps:

1. Select **Implementation** in the Library section and click the + (plus) button.
2. Select **Integration Service**.
3. Name it **Bluemix Mobile Data Service**.
4. Click **Finish**.
5. Drag two server script tasks and a Java integration task onto the canvas, arrange, and name as shown in Figure 6-37 on page 182.

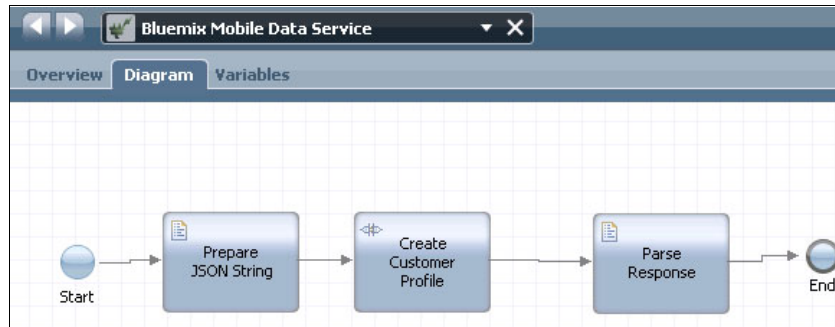


Figure 6-37 Mobile Data Integration Service implementation

6. Select the **Variables** tab and create the variables as shown in Figure 6-38.

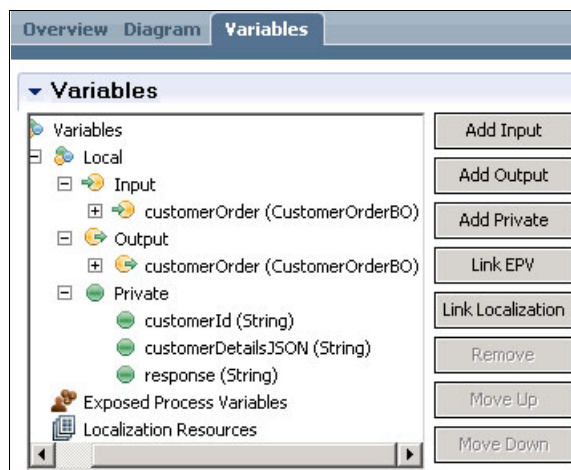


Figure 6-38 Mobile Data Integration Service variables

7. Select the **Prepare JSON String** task and under its Properties section select the **Implementation** tab.
8. Enter the JavaScript code as shown in Example 6-46. Note the use of the `json2.js` library to convert the `customerProfile` object into a JSON string.

Example 6-46 Prepare JSON String JavaScript source code

```

var customerDetails = tw.local.customerOrder.CustomerDetails;

var customerProfile =
[{"className":"CustomerProfile","attributes":{"firstName":customerDetails.Firstname,"lastName":customerDetails.Lastname,"dateOfBirth":customerDetails.DateOfBirth,"streetAddress":customerDetails.StreetAddress,"suburb":customerDetails.Suburb,"postCode":customerDetails.PostCode,"mobileNumber":customerDetails.MobileNumber,"emailAddress":customerDetails.EmailAddress}}]

var profileString = JSON.stringify(customerProfile);

tw.local.customerDetailsJSON = profileString;
  
```

9. Select the **Create Customer Profile** task.
10. Under the **Properties** section, select the **Definition** section.

11. Click **Select** to set the Java Class.
12. Select the **BluemixMobileDataService** class within the Bluemix Mobile Data JAR file.
13. Select the **submitCustomerProfile** method from the Method drop-down list as shown in Figure 6-39.

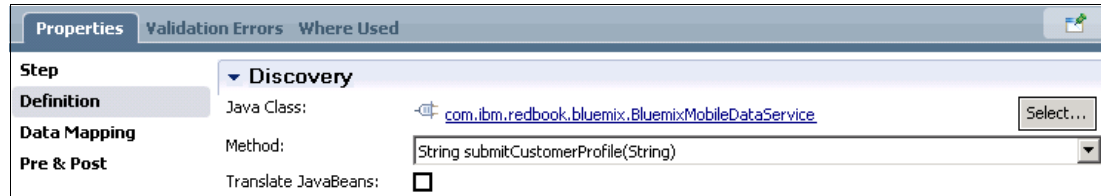


Figure 6-39 Set the Java Adapter method

14. Select the **Data Mapping** property.
15. Set the Input Mapping for Parameter 1 to **tw.local.customerDetailsJSON** and the Output Mapping for Return Value to **tw.local.response** as shown in Figure 6-40.

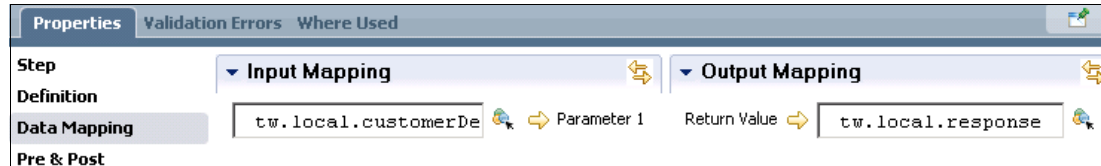


Figure 6-40 Set the Java Adapter parameter mapping for submitCustomerProfile

16. Select the Parse Response task and under the Properties section, select **Implementation**.
17. Enter the JavaScript code as shown in Example 6-47. The json2.js library is used again, this time to parse the returned string and create an object.

Example 6-47 Parse Response task javascript

```
var responseParsed = JSON.parse(tw.local.response);

if (responseParsed["status"] == "success") {
    // set the customerId as the ObjectId
    tw.local.customerId = responseParsed["object"][0]["objectId"];
    tw.local.customerOrder.CustomerDetails.CustomerID = tw.local.customerId;
}
```

The customerId is set so that it may be passed on to other systems as required. In this scenario, it will be passed to the Fulfillment system.

Configure the Create Customer Profile activity

In the New Order Installation BPD perform the following steps:

1. Right-click the **Create Customer Profile** activity and select **Activity Wizard** from the context menu.
2. Under **Activity Type Selection** select **System Task**.
3. Under **Library Element Selection**, select the **Select an existing service** option, and select **Bluemix Mobile Data Service**.

4. Click **Finish**.
5. Select the **Create Customer Profile** activity again and under the Properties section, select **Data Mapping**.
6. Set the Input Mapping for customerOrder to **tw.local.customerOrder** and the Output Mapping for customerOrder to **tw.local.customerOrder**.

The only change to the customerOrder is setting the `customerId`, which will be returned by the customer MDM system. When the customer order was taken previously, the `customerId` was set to an empty value.

6.6.6 Implementing the Schedule Work Order system activity

As with the Create Customer Profile activity, the Schedule Work Order activity is implemented to use Bluemix as a representation of a Work Order Scheduling system. However, for this activity, the Bluemix service is implemented using Node.js in addition to the Mobile Data service. The Create Customer Profile activity submits the customer order to Bluemix and in return it receives the associated work order.

Create the Bluemix Node.js service

The Bluemix application created in “Create the Bluemix Mobile Data service” on page 175 includes a Node.js runtime (see Figure 6-41). Node.js is an application server platform that runs Javascript-based applications. Refer to the node.js website at <http://nodejs.org>. This Bluemix application will be extended to support the Schedule Work Order System activity.

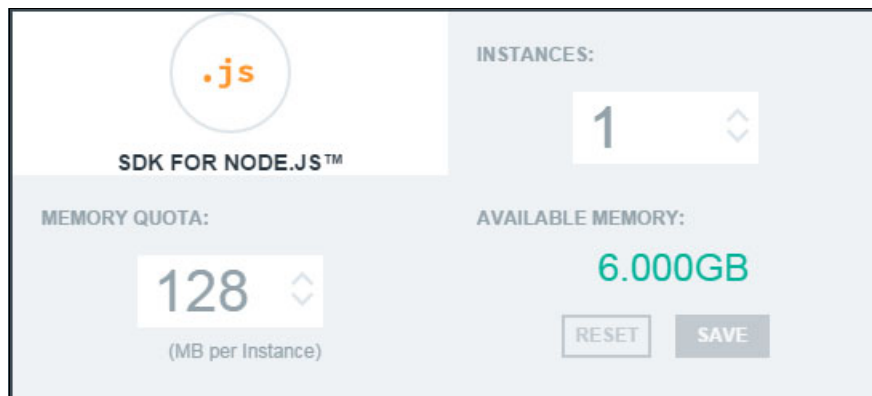


Figure 6-41 Bluemix Node.js service instance details

In Bluemix, perform the following tasks to implement the Node.js service:

1. Open the Bluemix application, for example, **redbookapp8240**.
2. Double-click the **Mobile Data Service** tile to open it.
3. Click the **Mobile Data** tab.
4. In a text editor, create the JSON file `CustomerOrder.json` as listed in Example 6-48 on page 185. See “Samples compressed file content” on page 308 for information about how to locate this file in the samples.

Notice that the JSON file contains an array with one element. This is required by the Mobile Data service when importing a JSON file.


```
[
  {
    "contractTerm": "12",
    "dateOfInstall": "17/10/2014",
    "timeOfInstall": "1",
    "CustomerDetails": {
      "customerId": "324275987234598345",
      "Firstname": "John",
      "Lastname": "Smith",
      "DateOfBirth": "10/10/1974",
      "StreetAddress": "1 Smith St",
      "Suburb": "Smithville",
      "PostCode": "30000",
      "MobileNumber": "555-555-5555",
      "EmailAddress": "john@smith.org"
    },
    "ProductDetails": {
      "productCode": "CBL",
      "cableModem": "0",
      "cableSpeed": "0"
    }
  }
]
```

5. Save and drag the file into the import dial as shown in Figure 6-42.

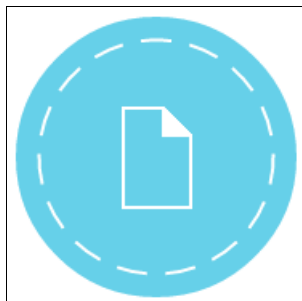


Figure 6-42 Drag file into the import dial

6. Dragging the file into the dial triggers an import process that prompts for the creation of the class for the data being imported (see Figure 6-43 on page 186).
7. Enter the Class Name **CustomerOrder** and click **Import**.

Import Data

You can import a wide variety of files, such as videos or JSON database files and then use the files with the services in your application.

Data File

CustomerOrder.json **Browse...**

Import to Class

Create a new Class

Class Name

CustomerOrder

Import **Cancel**

Current Storage **5.35KB**

Figure 6-43 Bluemix Mobile Data data import class creation form

8. Navigate back to the Overview of the Bluemix application and click the **ADD GIT** link on the upper right to create a GIT repository. There is already a Node.js runtime within this application.
9. Bluemix prompts to confirm the creation of the GIT Repository. Ensure that **Populate the repository with the starter application package...** is selected. Bluemix creates a repository in GIT using a Mobile Data Bluemix Node.js template.
10. Click **EDIT CODE** on the top of the Overview page to view the new repository.

Bluemix creates several files based on the Mobile Data Node.js template (see Figure 6-44 on page 187).

folder .git	6/10/2014 16:02:32	
folder lib	6/10/2014 16:02:32	
folder public	6/10/2014 16:02:32	
file .cfignore	6/10/2014 16:02:32	1 KB
file .gitignore	6/10/2014 16:02:32	1 KB
file app.js	6/10/2014 16:02:32	2 KB
file instructions.md	6/10/2014 16:02:32	1 KB
file License.txt	6/10/2014 16:02:32	1 KB
file manifest.yml	6/10/2014 16:02:32	1 KB
file package.json	6/10/2014 16:02:32	1 KB
file project.json	6/10/2014 16:02:32	1 KB
file README.md	6/10/2014 16:02:32	1 KB

Figure 6-44 Bluemix Mobile Data files included in template

To make it easier to edit the application, clone the repository to your local hard disk.

11. If you do not have GIT installed, download it from <http://git-scm.com/downloads> and install it.

12. Open a terminal window and create a new folder to put the repository into.

13. Enter the command: **git clone https://hub.jazz.net/git/<username>/<appname>**.

For example: **git clone https://hub.jazz.net/git/johnsmith/redbookapp8240**.

Running this command makes a copy of all the files from the repository into your local hard disk. Enter your Bluemix user ID and password when prompted.

14. Install Node.js on your local machine; download the installer from <http://nodejs.org>.

15. From a terminal window, enter the command: **npm install node-uuid --save**.

A UUID generator library is downloaded and added to the local repository. The UUID will be used to generate identifiers for the Customer Order and the Work Order.

16. Enter the following command to add the new modules to git:

git add node_modules

17. Using a text editor, change the app.js to match the code shown in Example 6-49.

Example 6-49 Schedule Work Order app.js source code

```

var express = require('express'),
    uuid = require('node-uuid'),
    app = express(),
    ibmbuemix = require('ibmbuemix'),
    ibmdata = require('ibmdata'),
    config = {
      // change to real application route assigned for your application
      applicationRoute : "{appname}.mybluemix.net",
      // change to real application ID generated by Bluemix for your application

```

```

        applicationId : "{enter your app id}"
    };

    // init core sdk
    ibmbluemix.initialize(config);
    var logger = ibmbluemix.getLogger();

    //redirect to cloudcode doc page when accessing the root context
    app.get('/', function(req, res){
        res.sendFile('public/index.html');
    });

    // init service sdks
    app.use(function(req, res, next) {
        req.data = ibmdata.initializeService(req);
        // req.ibmpush = ibmpush.initializeService(req);
        req.logger = logger;
        next();
    });

    // init basics for an express app
    app.use(require('./lib/setup'));

    //uncomment below code to protect endpoints created afterwards by MAS
    //var mas = require('ibmsecurity')();
    //app.use(mas);

    var ibmconfig = ibmbluemix.getConfig();

    logger.info('mbaas context root: '+ibmconfig.getContextRoot());
    // "Require" modules and files containing endpoints and apply the routes to our
    application
    app.use(ibmconfig.getContextRoot(), require('./lib/accounts'));
    app.use(ibmconfig.getContextRoot(), require('./lib/staticfile'));

    // Want to see how you can easily extend this template to work with third party
    node modules?
    // If so, add the Twilio service to your Mobile Cloud application and uncomment
    this next line.
    // app.use(ibmconfig.getContextRoot(), require('./lib/mytwilio')(ibmbluemix));

    var contextRoot = ibmconfig.getContextRoot();
    var appContext = express.Router();
    app.use(contextRoot, appContext);

    console.log("console.log contextRoot: "+contextRoot);
    logger.info("logger.info contextRoot: "+contextRoot);

    appContext.get('/customers',function(req,res) {
        console.log("console.log /customer block entered");
        logger.info("logger.info /customer block entered");
        var query = req.data.Query.ofType("CustomerProfile");

        query.find().done(function(items) {

```

```

        res.send(items);
    }, function(err) {
        res.status(500);
        res.send(err);
    });
});

appContext.get('/workorders',function(req,res) {
    console.log("console.log /customer block entered");
    logger.info("logger.info /customer block entered");
    var query = req.data.Query.ofType("WorkOrder");

    query.find().done(function(items) {
        res.send(items);
    }, function(err) {
        res.status(500);
        res.send(err);
    });
});

appContext.post('/location',function(req,res) {

    console.log("console.log /location block entered");
    logger.info("logger.info /location block entered");

    var item = req.data.Object.ofType("Location",req.body);

    item.save().then(function(saved) {
        res.send({"satus":"ok"})
    },function(err) {
        res.status(500);
        res.send(err);
    });
});

appContext.post('/customerorders',function(req,res) {

    console.log("console.log /workorders block entered");
    logger.info("logger.info /workorders block entered");

    var item = req.data.Object.ofType("CustomerOrder",req.body);

    // create a UUID for the customer order

    var custmrOrderId = uuid.v4();
    var customerDetails = item.get("CustomerDetails");
    var customerLocation = {
        "streetAddress":customerDetails["StreetAddress"],
        "suburb":customerDetails["Suburb"],
        "postCode":customerDetails["PostCode"]
    }
}

```

```

// var productDetails = item.get("ProductDetails")

// set the customer order id in the customer order document
item.set("customerOrderId",custmrOrderId);

item.save().then(function(saved) {

    console.log("console.log /workorders customer order saved");

    // create a work order id

    var workOrderId = uuid.v4();

    var workOrder = {
        "customerOrderId":custmrOrderId,
        "orderId":workOrderId,
        "groupId":"FSTeamTag1",
        "title":"Repair cable internet",
        "description":"Cable internet not functioning, must be resolved",
        "tasks":"1. Check cable modem, 2. Check connections, 3. Rebook",
        "dateRequired":"17/10/2014",
        "timeRequired":"1",
        "location":{"streetAddress":customerLocation["streetAddress"],
                    "suburb":customerLocation["suburb"],
                    "postCode":customerLocation["postCode"]}
    };

    res.send(workOrder);

},function(err) {
    console.log("console.log /workorders there was an error");
    res.status(500);
    res.send(err);
});
});

app.listen(ibmconfig.getPort());
logger.info('Server started at port: '+ibmconfig.getPort());

```

18. Save the file.

19. Enter the following GIT command to confirm that GIT has detected the changes:

```
git status
```

20. Enter the following GIT command to commit the changes to the local repository:

```
git commit -a -m "added node applicaton code"
```

21. Running **git status** again shows that the changes have been committed locally but now need to be pushed up to the remote repository so they can be deployed to the Node.js runtime on Bluemix.

22. Enter the following GIT command to push the changes to the remote Bluemix repository:

git push

The Bluemix DevOps service is configured by default to auto-redeploy when a push is made.

23. You can observe the console output by connecting to Bluemix using the Cloud Foundry command-line interface (CLI) tool for Bluemix and entering the command **cf logs <appname>**. For example, **cf logs redbookapp8240**. Download the Cloud Foundry CLI tool from <https://github.com/cloudfoundry/cli#downloads>.

The Node.js code has hard-coded details for the work order. In a real implementation, this detail comes from a production work order system and provides different responses. A Location resource is also included in the Node.js code to support Worklight submitting location data. The location data is added as a Location JSON object in the Bluemix Mobile Data service.

You can try the new Bluemix service using the **CURL** command shown in Example 6-50.

Example 6-50 CURL command to test Bluemix service

```
curl -X POST
"http://redbookapp8240.mybluemix.net/redbookapp8240/v1/apps/5fb5d138-7af0-4a2a-8c4
a-c3517f8be528/customerorders" \
-H "Content-Type: application/json" \
-d $' {\n  "contractTerm": "12",\n  "dateOfInstall": "17/10/2014",\n
"timeOfInstall": "1",\n  "CustomerDetails": {\n    "customerId":
"324275987234598345",\n    "Firstname": "John08",\n    "Lastname":
"Smith08",\n    "DateOfBirth": "10/10/1974",\n    "StreetAddress": "1 Smith
St",\n    "Suburb": "Smithville",\n    "PostCode": "30000",\n
"MobileNumber": "555-555-5555",\n    "EmailAddress": "john@smith.org"\n  },\n
"ProductDetails": {\n    "productCode": "CBL",\n    "cableModem": "0",\n
"cableSpeed": "0"\n  }\n }' \
-m 30 \
-v \
```

Create the Bluemix integration service

In this section, the Java integration adapter implemented for the Crate Customer Profile activity is extended to include a call to the Bluemix Node.js resources created for the Schedule Work Order activity.

The Java source is modified using the Eclipse IDE to include a new method for the BluemixMobileDataService class. A JAR file will be regenerated and imported into IBM BPM, replacing the JAR file previously imported.

1. Open the Eclipse IDE and add the methods to the BluemixMobileDataService class as shown in Example 6-51.

Example 6-51 submitCustomerOrder method

```
public String submitCustomerOrder(String jsonString) throws Exception {

    String stringResponse;

    CloseableHttpClient httpClient = HttpClients.createDefault();

    try {
```

```

        HttpPost httpPost = new
HttpPost("http://{appname}.mybluemix.net/{appname}/v1/apps/"
        + "{appId}/customerorders");

        httpPost.setHeader("Content-type","application/json");

        System.out.println("inputParameter = "+jsonString);
        httpPost.setEntity(new StringEntity(jsonString));

        CloseableHttpResponse response = httpClient.execute(httpPost);
        try {
            System.out.println(response.getStatusLine());
            HttpEntity entity = response.getEntity();

            stringResponse = EntityUtils.toString(entity);
            System.out.println("Content = "+ stringResponse);
            System.out.println("Parameter = " + jsonString);

            EntityUtils.consume(entity);
        } finally {
            response.close();
        }

        } finally {
            System.out.println("Closing");
            httpClient.close();
        }

        System.out.println("stringResponse = " + stringResponse);
        return stringResponse;
    }

```

2. Export the JAR file.
3. In IBM Process Designer, open the **New Order Installation BPD**.
4. Select **Files** from the library section and double-click the **BluemixMobileDataService.jar** file.
5. Under the File section, select **Browse** to locate the newly created JAR file. This file will replace the one already loaded in IBM BPM.
6. Click **Save** to commit the change.
7. Open the Bluemix Mobile Data service integration under the Library section.
8. Select **Create Customer Profile** and reselect the submitCustomerProfile method.
9. Create a new Integration Service and label it **Bluemix Schedule Work Order**.
10. Drag from the palette two Server Script items and a Java Integration item as shown in Figure 6-45, wiring them up accordingly.

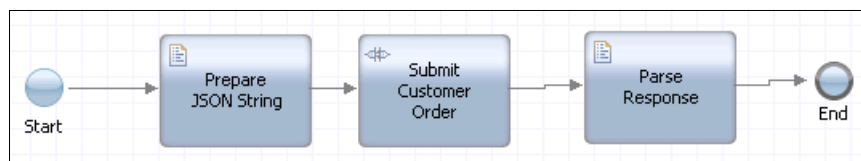


Figure 6-45 IBM BPM Integration Service used to interface with Bluemix Mobile Data service

11. Select the **Variables** tab and define the variables as shown in Figure 6-46. The customerOrder and workOrder variables use the existing CustomerOrderBO and WorkOrderBO types.

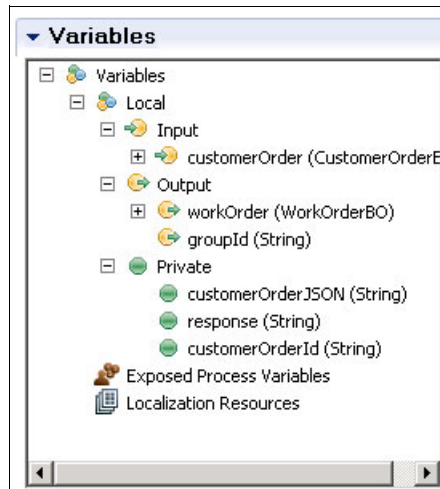


Figure 6-46 Integration Service variables

12. Select the **Prepare JSON String** step and under the Properties section, select **Implementation** and enter the script as shown in Example 6-52.

Example 6-52 Bluemix Schedule Work Order: Prepare JSON String script

```
var customerDetails = tw.local.customerOrder.CustomerDetails;

var customerOrder = {"contractTerm":tw.local.customerOrder.contractTerm,
    "dateOfInstall":tw.local.customerOrder.dateOfInstall,
    "timeOfInstall":tw.local.customerOrder.timeOfInstall,
    "CustomerDetails":{
        "customerId":customerDetails.CustomerID,
        "Firstname":customerDetails.Firstname,
        "Lastname":customerDetails.Lastname,
        "DateOfBirth": customerDetails.DateOfBirth,
        "StreetAddress": customerDetails.StreetAddress,
        "Suburb": customerDetails.Suburb,
        "PostCode": customerDetails.PostCode,
        "MobileNumber": customerDetails.MobileNumber,
        "EmailAddress": customerDetails.EmailAddress},
    "ProductDetails": {
        "productCode":
tw.local.customerOrder.ProductDetails.productCode,
        "cableModem":
tw.local.customerOrder.ProductDetails.cableModem,
        "cableSpeed":
tw.local.customerOrder.ProductDetails.cableSpeed}
    };

var orderString = JSON.stringify(customerOrder);

tw.local.customerOrderJSON = orderString;
```

13. Select the **Submit Customer Order** step and under the Properties section, select **Definition** and set the Java Class to **BluemixMobileDataService** and the Method to **submitCustomerOrder**, as shown in Figure 6-47.

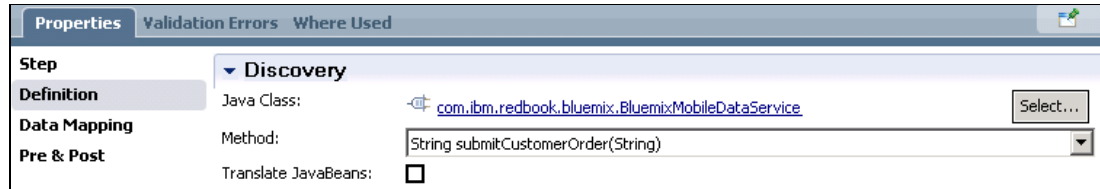


Figure 6-47 Set the Java Adapter method to submitCustomerOrder

14. Also within the Properties section, select **Data Mapping** and set the Input Mapping to **tw.local.customerOrderJSON-> Parameter 1** and the Output Mapping to **Return Value->tw.local.response**.
15. Select the **Parse Response** step and under the Properties section, select **Implementation** and enter the script as shown in Example 6-53.

Example 6-53 Bluemix Schedule Work Order: Parse Response script

```
var responseParsed = JSON.parse(tw.local.response);

tw.local.customerOrderId = responseParsed["customerOrderId"];
tw.local.groupId = responseParsed["groupId"];

var location = responseParsed["location"];

tw.local.workOrder = new tw.object.WorkOrderB0();

tw.local.workOrder.orderId = responseParsed["orderId"];
tw.local.workOrder.title = responseParsed["title"];
tw.local.workOrder.description = responseParsed["description"];
tw.local.workOrder.tasks = responseParsed["tasks"];
tw.local.workOrder.dateRequired = responseParsed["dateRequired"];
tw.local.workOrder.timeRequired = responseParsed["timeRequired"];

tw.local.workOrder.location = new tw.object.LocationB0();

tw.local.workOrder.location.streetAddress = location["streetAddress"];
tw.local.workOrder.location.suburb = location["suburb"];
tw.local.workOrder.location.postCode = location["postCode"];
```

Configure the Schedule Work Order system activity

Perform the following steps:

1. Switch back to the New Order Installation BPD, select the **Schedule Work Order** and under the Properties section, select **Implementation** and set the service to **Bluemix Schedule Work Order**.
2. Within the Properties section, select **Data Mapping** and set the Input Mapping to **tw.local.customerOrder->customerOrder** and Output Mapping to **workOrder->tw.local.workOrder** and **groupId->tw.local.groupId**.

6.6.7 Implementing the Send Notification to Field Staff activity

When a work order is created, the Work Order Scheduler assigns the work order to a field service team. The team is notified via a push notification to alert them to a new order.

IBM Worklight exposes a service that IBM BPM can call to send a push notification. The Send Notification to Field Staff activity is implemented as an automated system activity that calls the IBM Worklight service and passes it the groupId received from the scheduling system to send the notification and a notification message.

An integration service is created for interfacing with Worklight.

Create the IBM BPM Worklight Notification Integration Service

Perform the following steps:

1. Open Eclipse and create a new Java project.
2. Import the same Apache HTTP Client library as for the Schedule Work Order and Create Customer Profile Java projects.
3. Create a Class titled WorklightService and enter the Java code as shown in Example 6-54.

Example 6-54 WorklightService Java Source Code

```
package com.ibm.redbook.worklight;

import java.net.URLEncoder;

import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;

public class WorklightService {

    public String sendPushNotification(String groupId,String notificationMsg)
    throws Exception {

        String stringResponse;

        CloseableHttpClient httpClient = HttpClients.createDefault();

        try {

            String urlString =
                "[\"FieldService\",[\""+groupId+"\"],\""+notificationMsg+"\"]";
            System.out.println("Unencoded urlString = " + urlString);

            String encodedUrlString = URLEncoder.encode(urlString, "UTF-8");

            System.out.println("Encoded urlString = " + encodedUrlString);

            HttpGet httpget = new HttpGet("http://10.12.6.27:9084/WL_BPM_Project/");
```

```

        + "invoke?adapter=PushAdapter&procedure=submitNotification"
        + "&parameters="+encodeURIComponent);

    System.out.println("inputParameter = "+groupId);

    CloseableHttpResponse response = httpClient.execute(httpget);
    try {
        System.out.println(response.getStatusLine());
        HttpEntity entity = response.getEntity();

        stringResponse = EntityUtils.toString(entity);
        System.out.println("Content = "+ stringResponse);
        System.out.println("Parameter = " + groupId);

        EntityUtils.consume(entity);
    } finally {
        response.close();
    }

    } finally {
        System.out.println("Closing");
        httpClient.close();
    }

    System.out.println("stringResponse = " + stringResponse);
    return stringResponse;
}
}

```

4. Export the Java project as a JAR file and import it to IBM Process Designer by selecting **Files** under the library section and selecting the exported JAR file.
5. Select **Implementation** in the library section, add a new integration service, and give it the label **Worklight Service**.

This integration service is simple because only two strings need to be passed as parameters into the Worklight Java service.

6. Drag a single Java Integration Service from the Palette to create the flow as shown in Figure 6-48.

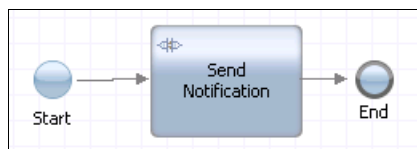


Figure 6-48 Java Integration Service for sending push notification

7. Select the **Variables** tab of the Worklight Service Integration Service and define two Input variables; **groupIdentifier** [String] and **notificationMessage** [String] and one Private variable **notificationResult** [String].
8. Select the **Send Notification** step and under Properties select **Definition** and set the Java Class to **WorklightService** and the Method to **sendPushNotification**.

9. Still under Properties, select **Data Mapping** and set the Input Mapping as **tw.local.groupIdentifier->Parameter 1**, **tw.local.notificationMessage->Parameter 2** and the Output Mapping as Return **Value->tw.local.notificationResult**.
10. Open the **New Order Installation** BPD and select the **Send Notification to Field Staff** activity again.
11. Under the Properties section, select **Implementation** and then select the newly created Worklight Service integration service.
12. Select the **Data Mapping** section and set the Input Mapping as **tw.local.groupId->groupIdentifier** and **tw.local.workOrder.title->notificationMessage**.

Implementing the Update Fulfillment System and Notify Customer via SMS activities

These activities are not implemented in this IBM Redbooks publication scenario. They follow the same pattern as the Customer Profile and Work Order System interaction with Bluemix. It is left to the reader to implement if wanted.

6.7 IBM BPM REST API

IBM BPM includes rich REST APIs that support interacting with business processes, cases, and tasks. The subset of the BPM REST APIs used to interact with the IBM BPM tasks is listed below.

Retrieve the list of work orders

The list of work orders are retrieved by requesting a list of tasks that have been allocated to a user group. In this scenario, the group is the Field Service Technicians. Once a work order is accepted, it must no longer appear in the work order list for other technicians.

This feature can be implemented by inspecting the `taskAssignedTo>type` element as follows:

- If the `taskAssignedTo:type` element is set to User, the work order has been accepted:

```
"taskAssignedTo": {
  "type": "User",
  "who": "bdaniel"
},
```

- If the `taskAssignedTo:type` element is set to Group, the work order has not been accepted:

```
"taskAssignedTo": {
  "type": "Group",
  "who": "Field Service
Team_T_2ef09645-2261-4b63-b76b-0103188f975b.dc725aa9-78cb-4a64-a85b-5e18f61320d2"
},
```

It is expected that the Worklight Server will filter the work orders using the content of the `type` data element before sending the list of work orders to the Field Service app.

In the case that a field technician accepts an order that has already been accepted, an appropriate error message is displayed. This situation may occur if work orders are being accepted while a field technician is looking at a list.

Note: A silent push notification could be sent to the mobile app to automatically remove accepted work orders while a technician is viewing the work order list. This feature is not covered in this IBM Redbooks publication but may be a useful exercise for the reader.

HTTP Method: PUT

URL:

`https://<bpm_host>:9443/rest/bpm/wle/v1/search/query?organization=byInstance&run=true&shared=false&filterByCurrentUser=true&columns=orderId%2Ctitle%2CtimeRequired%2CstreetAddress%2Csuburb`

Parameters:

The retrieve work order list parameters are listed in Table 6-3.

Table 6-3 Retrieve work order list parameters

Name	Type	Description	Value
organization	String	A string indicating how the results should be organized. Valid values are: byTask, byInstance.	byInstance
run	String	A flag which indicates whether or not this search query should be performed.	true
shared	Boolean	If this query is a saved search, this flag indicates whether it should be shared with other users.	false
filterByCurrentUser	String	A flag which indicates whether the search results are restricted to entities associated with the current user	true
columns	String	A comma-separated list of column names that should be returned by the search.	orderId,title,timeRequired,streetAddress,suburb

Response:

The sample response is shown in Example 6-55.

Example 6-55 Retrieve List of Work Orders Sample Response

```
{
  "status": "200",
  "data": {
    "data": [
      {
        "assignedToRole": null,
        "taskAssignedTo": {
          "type": "User",
          "who": "bdaniel"
        },
        "orderId": "W01001",
```

```

        "streetAddress": "24, Blue Street",
        "suburb": "Texas",
        "timeRequired": 1,
        "title": "Install cable Internet",
        "instanceId": 510,
        "instanceStatus": "Completed",
        "taskAttachedExtActivityRef": null,
        "taskAttachedInfoPathFormRef": null,
        "taskId": 15317,
        "taskStatus": "Closed"
    },
    {
        "taskAssignedTo": {
            "type": "Group",
            "who": "Field Service
Team_T_2ef09645-2261-4b63-b76b-0103188f975b.c9d1fb54-0506-4309-97e4-e3e8b2eedae2"
        },
        "assignedToUser": null,
        "orderId": "W01001",
        "streetAddress": "12, Green Road",
        "suburb": "South Bank",
        "timeRequired": 1,
        "title": "Install cable Internet",
        "instanceId": 512,
        "instanceStatus": "Active",
        "taskAttachedExtActivityRef": null,
        "taskAttachedInfoPathFormRef": null,
        "taskId": 15321,
        "taskStatus": "Received"
    },
    {
        "assignedToRole": null,
        "taskAssignedTo": {
            "type": "User",
            "who": "bdaniel"
        },
        "orderId": "W01001",
        "streetAddress": "11, Sth Street",
        "suburb": "Blue",
        "timeRequired": 3,
        "title": "Install cable Internet",
        "instanceId": 513,
        "instanceStatus": "Active",
        "taskAttachedExtActivityRef": null,
        "taskAttachedInfoPathFormRef": null,
        "taskId": 15323,
        "taskStatus": "Received"
    },
    {
        "taskAssignedTo": {
            "type": "Group",
            "who": "Field Service
Team_T_2ef09645-2261-4b63-b76b-0103188f975b.dc725aa9-78cb-4a64-a85b-5e18f61320d2"
        },
        "assignedToUser": null,

```

```

        "orderId": "W01001",
        "streetAddress": "12, Red Street",
        "suburb": "Valley",
        "timeRequired": 0,
        "title": "Install cable Internet",
        "instanceId": 514,
        "instanceStatus": "Active",
        "taskAttachedExtActivityRef": null,
        "taskAttachedInfoPathFormRef": null,
        "taskId": 15325,
        "taskStatus": "Received"
    }
],
"organization": "byInstance",
"autoColumns": [
    "assignedToRole",
    "assignedToUser",
    "instanceId",
    "instanceStatus",
    "taskAssignedTo",
    "taskAttachedExtActivityRef",
    "taskAttachedInfoPathFormRef",
    "taskId",
    "taskStatus"
],
"offset": 0,
"size": 5,
"requestedSize": 10000,
"totalCount": 5
}
}

```

Accept the work order

A field technician can accept a work order and start updating the work order status.

HTTP Method: PUT

URL:

https://<bpm_host>:9443/rest/bpm/wle/v1/task/{taskId}?action=assign&toMe=true&parts=none

Parameters:

The work order parameters are listed in Table 6-4.

Table 6-4 Accept the work order parameters

Name	Type	Description	Value
taskId	String	The task identifier.	Example: 15373
action	String	Action to be performed for the specified task instance.	assign

Name	Type	Description	Value
toMe	Boolean	If specified as true then the task is assigned to the current user.	true
parts	String	A string indicating which parts of the response data should be returned (data, all, none)	none

Response:

The sample response is shown in Example 6-56.

Example 6-56 Accept the work order sample response

```

{
  "status": "200",
  "data": {
    "activationTime": "2014-09-28T07:40:06Z",
    "atRiskTime": "2014-09-27T19:01:57Z",
    "clientTypes": [
      "IBM_WLE_Coach"
    ],
    "completionTime": null,
    "containmentContextID": "539",
    "description": "",
    "displayName": "Step: Perform Installation",
    "dueTime": "2014-09-28T08:40:06Z",
    "isAtRisk": true,
    "kind": "KIND_PARTICIPATING",
    "lastModificationTime": "2014-09-28T07:40:06Z",
    "name": "Perform Installation",
    "originator": "admind",
    "owner": "bdaniel",
    "priority": 30,
    "startTime": "2014-09-28T07:40:06Z",
    "state": "STATE_CLAIMED",
    "tkiid": "15373",
    "piid": "539",
    "processInstanceName": "Installation Work Order:539",
    "status": "Received",
    "priorityName": "Normal",
    "assignedTo": "bdaniel",
    "assignedToDisplayName": "Bonnie",
    "assignedToType": "user",
    "serviceID": "1.eb0fb5a8-cb5f-4e9c-98e4-99d955ae6efb",
    "flowObjectID": "bpdid:d7864cc5ffa79ff9:-543ca621:148068f7725:-7fcb",
    "nextTaskId": null,
    "collaboration": {
      "status": false,
      "currentUsers": []
    }
  }
}

```

Get Task Data

A field technician can retrieve the business data from a task. Get Task Data is used to retrieve the Work Order Report business data from within the task.

HTTP Method: GET

URL:

`https://<bpm_host>:9443/rest/bpm/wle/v1/task/15373?action=getData&fields=workOrderReport`

The Get Task Data parameters are listed in Table 6-5.

Table 6-5 Get Task Data parameters

Name	Type	Description	Value
taskId	String	The task identifier	Example: 15373
action	String	Action to be performed for the specified task instance	getData
fields	String	A comma-separated list of variables to be retrieved	workOrderReport

Response:

The sample response is shown in Example 6-57.

Example 6-57 Get Task Data sample response

```
{
  "status": "200",
  "data": {
    "activationTime": "2014-09-28T07:40:06Z",
    "atRiskTime": "2014-09-27T19:01:57Z",
    "clientTypes": [
      "IBM_WLE_Coach"
    ],
    "completionTime": null,
    "containmentContextID": "539",
    "description": "",
    "displayName": "Step: Perform Installation",
    "dueTime": "2014-09-28T08:40:06Z",
    "isAtRisk": true,
    "kind": "KIND_PARTICIPATING",
    "lastModificationTime": "2014-09-28T07:40:06Z",
    "name": "Perform Installation",
    "originator": "admind",
    "owner": "bdaniel",
    "priority": 30,
    "startTime": "2014-09-28T07:40:06Z",
    "state": "STATE_CLAIMED",
    "tkiid": "15373",
    "piid": "539",
    "processInstanceName": "Installation Work Order:539",
    "status": "Received",
    "priorityName": "Normal",
    "assignedTo": "bdaniel",
```

```

    "assignedToDisplayName": "Bonnie",
    "assignedToType": "user",
    "serviceID": "1.eb0fb5a8-cb5f-4e9c-98e4-99d955ae6efb",
    "flowObjectId": "bpdid:d7864cc5ffa79ff9:-543ca621:148068f7725:-7fcb",
    "nextTaskId": null,
    "collaboration": {
      "status": false,
      "currentUsers": []
    }
  }
}

```

Set Task Data

A field technician can set the business data for a task. Set Task is used to update the Work Order Report business data from within the Task.

HTTP Method: PUT

URL:

`https://10.12.6.27:9443/rest/bpm/wle/v1/task/15373?action=setData¶ms=%7B%22workOrderReport%22%3A%7B%22orderId%22%3A%2222332%22%2C%22status%22%3A%22On%20Site%22%2C%22summary%22%3A%22%22%7D%7D&parts=none`

The Set Task Data parameters are listed in Table 6-6.

Table 6-6 Set Task Data parameters

Name	Type	Description	Value
taskId	String	The task identifier	Example: 15373
action	String	Action to be performed for the specified task instance	setData
params	String	A string containing a JSON expression that contains one or more variable settings.	Example: {"workOrderReport":{"orderId":"22332","status":"On Site","summary":""}}

Response:

The sample response is shown in Example 6-58.

Example 6-58 Set Task Data sample response

```

{
  "status": "200",
  "data": {
    "result": "{\\\"workOrderReport\\\":{\\\"orderId\\\":\\\"22332\\\",\\\"status\\\":\\\"On Site\\\",\\\"summary\\\":\\\"\\\",\\\"@metadata\\\":{\\\"objectId\\\":\\\"fb82f165-ce79-4a38-a855-c6a5692d26fe\\\",\\\"dirty\\\":true,\\\"shared\\\":false,\\\"rootVersionContextID\\\":\\\"2064.9fe6c2eb-9757-4fd1-bca4-e9e06144e110T\\\",\\\"className\\\":\\\"WorkOrderReportB0\\\"}}}}",
    "resultMap": {
      "workOrderReport": {
        "orderId": "22332",
        "status": "On Site",

```

```

    "summary": "",
    "@metadata": {
      "objectID": "fb82f165-ce79-4a38-a855-c6a5692d26fe",
      "dirty": true,
      "shared": false,
      "rootVersionContextID": "2064.9fe6c2eb-9757-4fd1-bca4-e9e06144e110T",
      "className": "WorkOrderReportB0"
    }
  }
}
}
}
}
}

```

Finish Task

A field technician can complete a task. Finish Task is used to close the Work Order and set the final Work Order Report business data from within the task.

HTTP Method: PUT

URL:

`https://<bpm_host>:9443/rest/bpm/wle/v1/task/15317?action=finish&parts=none¶ms=%7B%22workOrderReport%22%3A%7B%22orderId%22%3A%2222332%22%2C%22status%22%3A%22Complete%22%2C%22summary%22%3A%22Cable%20modem%20installed.%20Internet%20active.%22%7D%7D`

Parameters:

The Finish Task parameters are listed in Table 6-7.

Table 6-7 Finish Task parameters

Name	Type	Description	Value
taskId	String	The task identifier.	Example: 15373
action	String	Action to be performed for the specified task instance.	finish
parts	String	A string indicating which parts of the response data should be returned (data, all, none).	none
params	String	A string containing a JSON expression that contains one or more variable settings.	Example: {"workOrderReport":{"orderId":"22332","status":"Complete","summary":"Cable modem installed. Internet active."}}

Response:

The sample response is shown in Example 6-59 on page 205.

```
{
  "status": "200",
  "data": {
    "activationTime": "2014-10-08T01:45:08Z",
    "atRiskTime": "2014-10-07T05:42:26Z",
    "clientTypes": [
      "IBM_WLE_Coach"
    ],
    "completionTime": "2014-10-08T06:52:22Z",
    "containmentContextID": "554",
    "description": "",
    "displayName": "Step: Perform Installation",
    "dueTime": "2014-10-08T02:45:08Z",
    "isAtRisk": true,
    "kind": "KIND_PARTICIPATING",
    "lastModificationTime": "2014-10-08T06:52:22Z",
    "name": "Perform Installation",
    "originator": "admind",
    "owner": "bdaniel",
    "priority": 30,
    "startTime": "2014-10-08T01:45:08Z",
    "state": "STATE_FINISHED",
    "tkiid": "15411",
    "piid": "554",
    "processInstanceName": "Installation Work Order:554",
    "status": "Closed",
    "priorityName": "Normal",
    "assignedTo": "bdaniel",
    "assignedToDisplayName": "Bonnie",
    "assignedToType": "user",
    "serviceID": "1.eb0fb5a8-cb5f-4e9c-98e4-99d955ae6efb",
    "flowObjectID": "bpdid:d7864cc5ffa79ff9:-543ca621:148068f7725:-7fcb",
    "nextTaskId": [],
    "collaboration": {
      "status": false,
      "currentUsers": []
    }
  }
}
```



Scenario 2: Advanced features

This chapter builds on the implementation of scenario 1 discussed in Chapter 6, “Scenario 1: Getting started” on page 111 by adding advanced features from IBM Worklight and IBM Business Process Manager.

In scenario 1, the field technicians use a mobile application to manage the work orders. In scenario 2, the use case is extended by adding a service parts order mobile application called *Parts app*. Furthermore, the customer in scenario 1 can order cable installation service over the phone. In scenario 2, the customer can complete and sign a new order agreement to request new TV services.

The IBM Worklight features added in this scenario are:

- ▶ Lightweight Third Party Authentication (LTPA) authentication and token propagation.
- ▶ Device single sign-on.
- ▶ Simple Data Sharing between two mobile applications.
- ▶ Integration with IBM Bluemix Mobile Data database.

The IBM BPM features added in this scenario are:

- ▶ Responsive coaches
- ▶ Case management

7.1 Scenario 2 overview

This section describes the scenario 2 use cases.

7.1.1 Scenario 2 requirements

Scenario 2 extends scenario 1 by adding additional requirements, which are implemented in the use cases described in this section. The additional requirements are:

- ▶ Customers must be able to submit a new order agreement form to request new TV services.
- ▶ Field technicians must be able to order parts needed to complete a work order from their mobile devices. To enhance the user experience, the field technician should be able to log in only once across mobile applications (single sign-on feature). For accountability, the new part order must reference the corresponding work order number in which the part will be used.

7.1.2 Customer use case

The following steps describe the customer interaction flow:

1. The customer completes a new order agreement form and signs it. The customer then either mails or e-mails the document to the cable TV company. This is a new step for scenario 2.
2. The Customer Accounts Team is notified of a new customer request and uploads a digital copy of the new order agreement form into the embedded Enterprise Content Management. When the document is uploaded, a new customer order case is automatically started.
3. A Customer Accounts Team member claims the first activity of the new customer order case and decides whether to start the New Order Installation business process (NOI-BP) within the new customer order case.
4. The NOI-BP creates a full customer profile in the customer master data management (this step is simulated by calling a Bluemix service).
5. The NOI-BP checks the scheduler for available field technicians (this step is simulated by calling a Bluemix service).
6. The NOI-BP sends a notification message to the field service team using Worklight push notification and then starts an installation work order business process (WO-BP) and assigns the WO-BP to a field technician team.
7. The WO-BP sends an SMS notification to the customer when the work order is complete. This step is not implemented in this scenario; it is added to the flow for completeness.
8. To finish the work order, the field technician, using the Field Service app, updates the status variable `tw.local.workOrderReport.status` in a user task of the WO-BP.
9. The WO-BP completes and returns back to the NOI-BP that called it. If the status is set to `Complete`, the NOI-BP updates the fulfillment system and notifies the customer via an SMS message (both simulated, not implemented in this example) and then the business process ends.
10. If the status is set to anything but `Complete`, a user task called `Follow-up with customer` starts. When this user task is complete, the business process ends.

7.1.3 Field technician use case

The following steps describe the field technician interaction flow:

1. The members of the field technician team receive a push notification regarding new work orders available.
2. The members of the field technician team retrieve the list of available work orders using the Field Service app.
3. A field technician selects and accepts the work order using the Field Service app.
4. The field technician updates the status of the work order while the installation is performed using the Field Service app.
5. *New in scenario 2:* The field technician opens the Parts app (the technician is automatically authenticated) and orders a new service part. The Parts app attaches the work order number and submits the part order. A new part request business process is started (simulated).
6. The field technician finishes the work order and updates the work order status as complete using the Field Service app.

7.1.4 Overall architecture

Scenario 2 extends scenario 1 by enabling the customer to fill and sign a new order agreement form to request a new cable service. In addition, a new Parts app is introduced. This mobile application leverages a new service parts database in Bluemix.

Figure 7-1 on page 210 shows a high-level architecture overview diagram for scenario 2. The diagram shows on the left the field technician using both the Field Service app and the Parts app. The top of the diagram shows the customer ordering a new cable service by sending an email with the new order agreement.

From an IBM BPM perspective, this scenario introduces:

- ▶ A use case where customers can submit new order forms to request new service. The submission of the files acts as a trigger to start a case type, leveraging the case management capabilities of IBM BPM.
- ▶ Responsive coach design techniques so that a coach form can be written once and used on multiple device form factors, for example, tablets and smartphones.
- ▶ A technique to expose an IBM BPM service that is called from a Worklight application.

This scenario is designed to use the architectural pattern described in 4.4.4, “Pattern 4: IBM BPM REST API” on page 77 also known as the *Headless BPM* pattern. Besides pattern 4, in this scenario, the call center team members use coach forms via a browser as in pattern 1 described in section 4.4.1, “Pattern 1: IBM BPM Process Portal” on page 76.

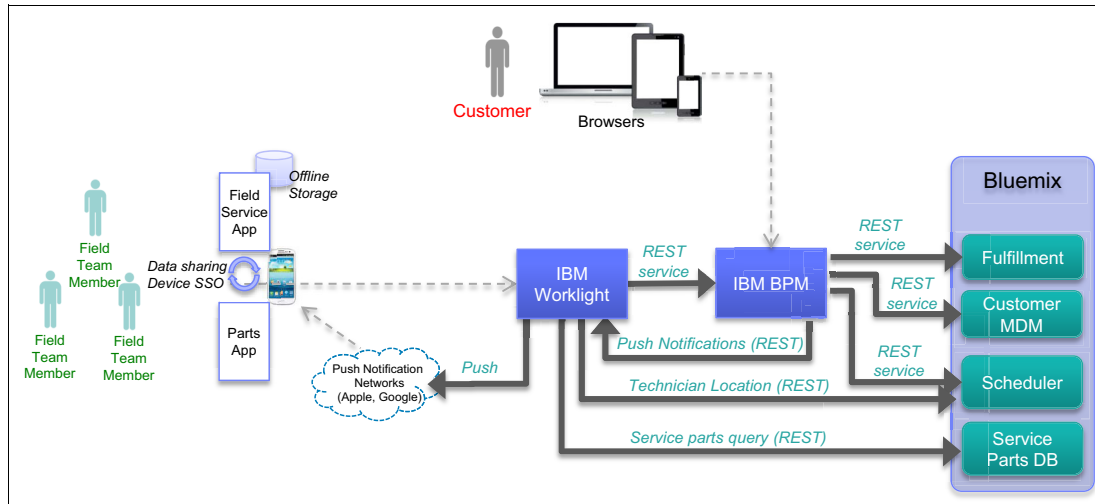


Figure 7-1 Scenario 2 architecture overview diagram

7.2 Parts app

In this scenario, the field technician uses a new Parts app to order service parts while the cable installation work is done. The application user interface leverages the jQuery Mobile framework. The application logic is written in JavaScript and uses the Worklight JavaScript APIs. From an application architecture point of view, the Parts app in scenario 2 is similar to the Field Service app from scenario 1.

The new Parts app works with the existing Field Service app that was introduced in scenario 1. The two mobile applications support single sign-on for a better user experience and also both applications share work order data.

The Parts app consists of the following pages:

- Login page: the field technician enters the login credentials in the UI shown in Figure 7-2 on page 211. According to the single sign-on feature, this step is not required if the field technician is already logged in to the Field Service app.

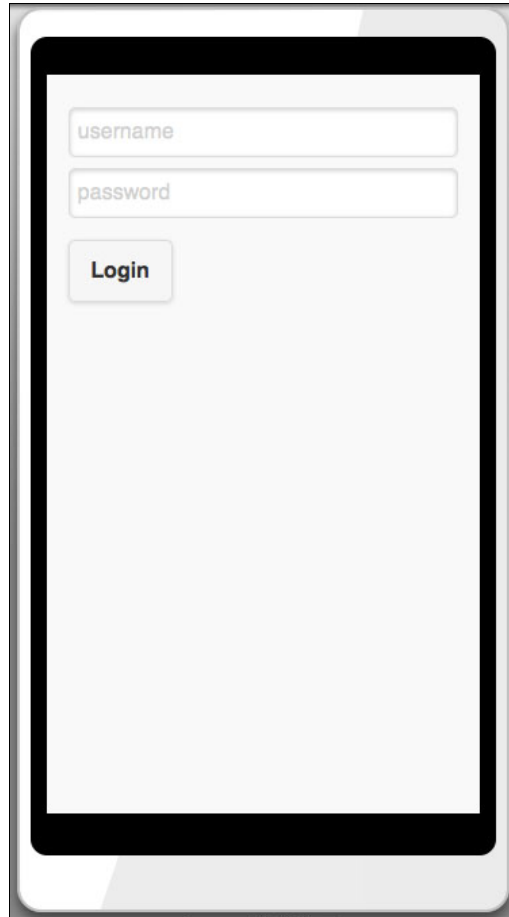


Figure 7-2 Parts app: Login page

- Search Input page: the field technician types in the part name that is required to finish the service work shown in Figure 7-3 on page 212.

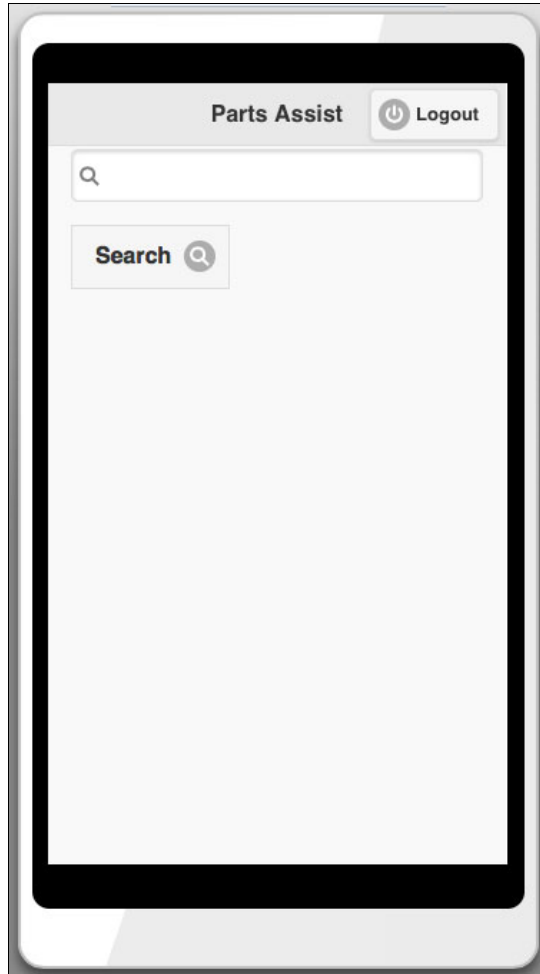


Figure 7-3 Parts app: Search Input page

- Search Results page: the field technician finds the service part in the Search Results page shown in Figure 7-4 on page 213.

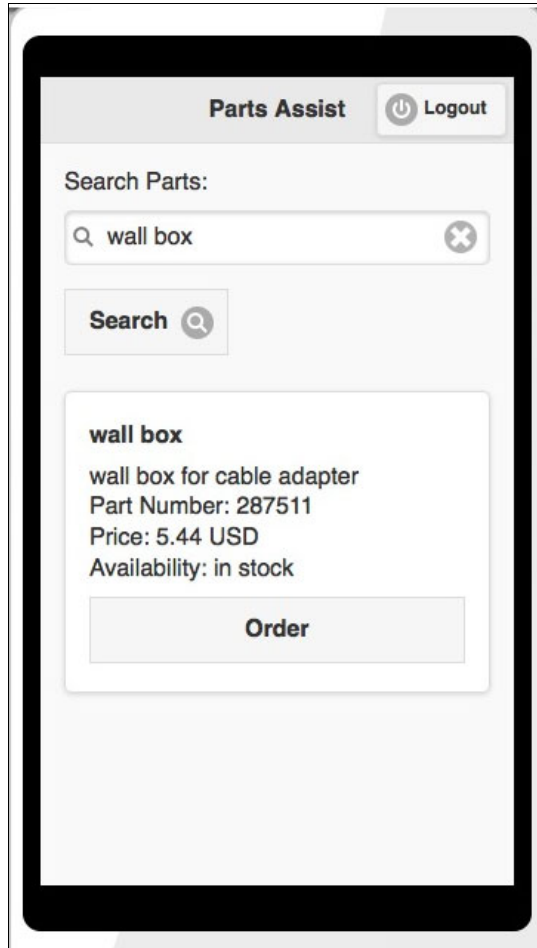


Figure 7-4 Parts app: Search Results page

- Part Order Confirmation page: the field technician clicks Order and receives a confirmation message that shows the part number and the related work order number. This work order number references the last work order accepted in the Field Service app. In other words, the work order number is the shared data across the two mobile applications. Figure 7-5 on page 214 shows the Part Order Confirmation page.

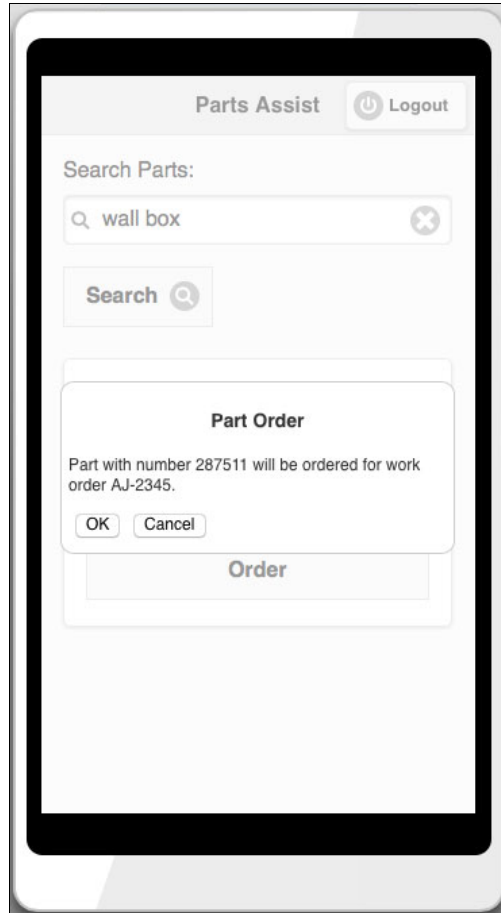


Figure 7-5 Parts app: Part Order Confirmation page

7.3 IBM Worklight features in this scenario

This section describes the following IBM Worklight features:

- ▶ LTPA authentication and token propagation.
- ▶ Device single sign-on.
- ▶ Simple Data Sharing between the two mobile applications.
- ▶ Integration with IBM Mobile Data for Bluemix database.

7.3.1 LTPA authentication and token propagation

Lightweight Third Party Authentication (LTPA) is a security token type that is used by IBM WebSphere Application Server and other IBM products. LTPA can be used to send the credentials of an authenticated user to back-end services. In this scenario, the back-end system is the IBM BPM server.

Use case security requirements

In scenario 1, Worklight delegated the validation of the user credentials to the IBM BPM server. In scenario 2, Worklight itself is responsible for validating the user credentials, and also sending the authentication LTPA token to the IBM BPM server.

Implementation of LTPA authentication and token propagation

After the user logs in by providing the user ID and password, the Worklight Server authenticates the credentials and generates an LTPA token, which is an encrypted hash that contains authenticated user information. The token is signed by a private key that is shared among all the servers that need to decode it. The token is in cookie form for HTTP services. LTPA tokens have a configurable expiration time to reduce the possibility for session hijacking.

Figure 7-6 shows the LTPA authentication sequence diagram:

1. The mobile application requests access to a secured resource.
2. Worklight Server responds requesting credentials. This request is called a *challenge*.
3. The challenge handler in the mobile application detects the challenge, collects the user credentials, and sends them to the Worklight Server.
4. The Worklight Server successfully validates the user credentials against the users defined locally in WebSphere Application Server and returns the LTPA token along with the resource initially requested.
5. Now the mobile application calls a secured adapter and sends the LTPA token in the request.
6. The adapter procedure extracts the LTPA token from the user session and sends it in the HTTP header to the IBM BPM REST API service on the IBM BPM server.
7. The IBM BPM server validates the LTPA token, runs the service, and sends the response back to the Worklight Server. Finally, the Worklight Server sends the response to the mobile application.

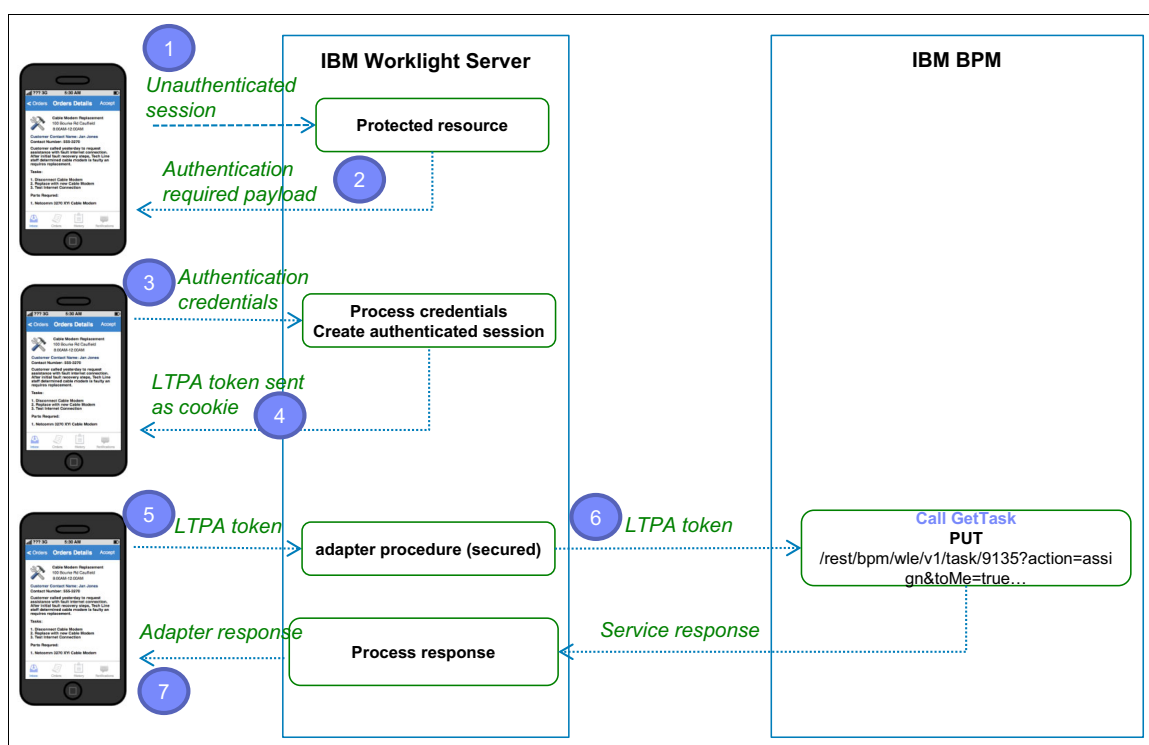


Figure 7-6 LTPA authentication sequence diagram

The following list provides a high-level implementation description. Refer to section 7.4, “SSO configuration with LTPA implementation” on page 217 for detailed implementation steps.

- ▶ The file `authenticationConfig.xml` must be updated with an appropriate security test, realm, and login module to enable the LTPA authentication in the Worklight Server.
- ▶ If Worklight and IBM BPM are running in different WebSphere Application Server domains, the LTPA private key must match between the two domains. Sharing the LTPA private key is accomplished by exporting it from one WebSphere Application Server instance and importing it into the other instance.
- ▶ To enable the LTPA token propagation to the IBM BPM server, every Worklight adapter procedure must include the LTPA token in the HTTP header of the IBM BPM application programming interface (API) call.

Note: In a real production implementation, both the Worklight Server and the IBM BPM server are connected to a single user directory or LDAP for authentication. In this scenario, the user directory is simulated by creating the same users in both the Worklight Server and the IBM BPM server.

7.3.2 Device single sign-on

When a user successfully logs in through a single sign-on (SSO)-enabled login module, the user gains access to all resources that are using the same login module, without having to authenticate again to each one.

Use case device single sign-on requirements

The field technician must be able to authenticate only once between the Field Service app and the Parts app.

Implementation of device single sign-on

Single sign-on is a property of a login module. You can enable single sign-on from a `mobileSecurityTest` element or from a `loginModule` element of `authenticationConfig.xml`. Enabling single sign-on for a custom security test is done on the `loginModule` element. Enabling single sign-on for a mobile security test is done on the `testUser` realm of the `mobileSecurityTest` element. The latter approach is used in the scenario 2 implementation, where the property `sso="true"` is added to the `testUser` realm.

Additional considerations

In a production environment if you enabled device single sign-on, it is recommended also to enable the Worklight application authenticity feature. Application authenticity makes sure that any application that tries to connect to a Worklight Server is authentic and was not tampered with or modified by some third-party attacker.

7.3.3 Data sharing across mobile applications

Starting with IBM Worklight v6.2.0, Worklight applications on a single device can securely share lightweight information. The Simple Data Sharing feature uses native APIs that are already present in the different mobile SDKs to provide one unified developer API. This Worklight API abstracts the different platform complexities, making it easier for developers to quickly implement code that allows for inter-application communication. This feature is supported on iOS and Android for both hybrid and native applications. After you enable the Simple Data Sharing feature, you can use the provided hybrid and native APIs to exchange simple key-value tokens among applications on the same device.

Use case data sharing requirements

The field technician updates the status of the work order while the installation is performed using the field service mobile application. When the field technician orders replacement parts using the Parts app, the part request submitted to IBM BPM must contain the original work order for which the replacement part is required to complete the job.

Data sharing implementation

The Simple Data Sharing feature must be enabled in both the Field Service app and the Parts app by defining the same application family domain in the Worklight application descriptor file. Using the Simple Data Sharing APIs, the field service mobile application sets the token for the current work order. Later, the Parts app receives the current work order number from the same key-value token.

7.3.4 Integration with IBM Mobile Data for Bluemix database

Bluemix is the IBM open cloud platform that provides mobile and web developers access to IBM software for integration, security, transaction, and other key functions, as well as software from IBM Business Partners. Built on the Cloud Foundry open source technology, Bluemix offers more control to application developers by using its Platform-as-a-Service (PaaS) offering, and also provides pre-built Mobile Backend-as-a-Service (MBaaS) capabilities. The goal is to simplify the delivery of an application by providing services that are ready for immediate use and hosting capabilities to enable internal scale development.

Bluemix Mobile Data is a no-sql cloud storage service that provides a native feel for storing mobile data, while the management and implementation of the data store is hidden. There are JavaScript, native, and REST APIs available that can be used to query the data.

Use case parts database requirements

The Parts app that the field technician uses to order new parts must be able to connect and query a remote parts database that provides information, such as part price and availability.

Parts database integration

When the field technician uses the Parts app to search for parts, the mobile application calls an HTTP Worklight adapter. This adapter, in turn, calls a REST web service exposed by IBM Mobile Data for Bluemix. The creation of the IBM Mobile Data for Bluemix parts database is described in the implementation details in section 7.6.1, “Creating the Parts database in Bluemix” on page 230.

7.4 SSO configuration with LTPA implementation

This section describes the implementation of SSO using Lightweight Third Party Authentication (LTPA). LTPA is used for authentication and also for sharing the authentication token to invoke the IBM BPM REST APIs.

Note: In this scenario, the WebSphere Application Server *Full* Profile is used for the Worklight Server.

To implement WebSphere LTPA-based authentication, perform the following steps:

1. Configure LTPA on WebSphere Application Server.

2. Configure the Worklight Server to authenticate using LTPA.
3. Configure the Worklight mobile app to authenticate using LTPA.

Note: See Appendix A, “Samples included with this book” on page 307 for information about the samples provided with this IBM Redbooks publication.

7.4.1 Configuring LTPA on WebSphere Application Server

To configure LTPA on WebSphere Application Server for this scenario, perform the following steps on the Worklight WebSphere Application Server and the IBM BPM WebSphere Application Server:

1. Enable administrative security on WebSphere Application Server:
 - a. In the WebSphere Application Server administrative console, go to **Security** → **Global Security**.
 - b. Select **Enable administrative security** as shown in Figure 7-7.

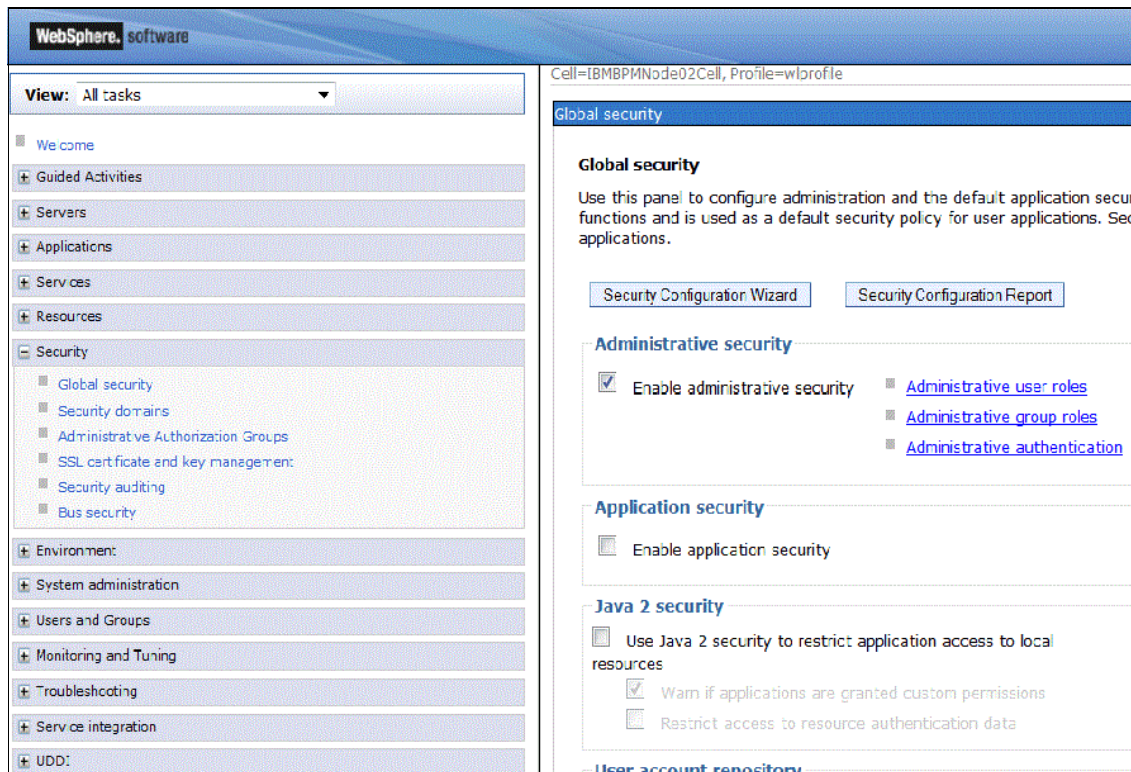


Figure 7-7 Enabling administrative security in WebSphere Application Server

2. Add users to WebSphere Application Server.

To share the authentication token between the Worklight and IBM BPM, both servers should be configured to use the same user registry or LDAP. In this scenario, the Worklight Server and IBM BPM server are *not* configured to use the same LDAP server. In this scenario, the environment is simulated by creating the same users on both servers.

To create new users in WebSphere Application Server, perform the following steps:

- a. Open the WebSphere Application Server administrative console where the Worklight Server or the IBM BPM server runs.

- b. Navigate to **Users and groups** → **Manage Users**.
- c. **Users and Groups** → **Manage Users**.
- d. Click **Create** as shown in Figure 7-8.

Manage Users

Search for Users

Search by: User ID * Search for: * * Maximum results: 100

Search

2 users matched the search criteria.

Create... **Delete** Select: Select an action...

Select	User ID	First name	Last name	E-mail	Unique Name
<input type="checkbox"/>	appcenteradmin	Administrator	of IBM Application Center		uid=appcenteradmin,o=defaultWIMFileBasedRealm
<input type="checkbox"/>	wasadmin	wasadmin	wasadmin		uid=wasadmin,o=defaultWIMFileBasedRealm

Page 1 of 1 Total: 2

Figure 7-8 Creating users on WebSphere Application Server

- e. Enter the User ID, First name, Last name.
 - f. Enter the password and confirm password
 - g. Click **Create**.
 - h. Click **OK**.
3. Configure SSO support across multiple WebSphere Application Server domains or cells.
- To support SSO in WebSphere Application Server across multiple WebSphere Application Server domains or cells, you must share the LTPA keys and the password among the domains. Perform the following steps to configure LTPA and generate the keys:

- a. Configure LTPA and generate the first LTPA keys. Follow the steps described in the WebSphere Application Server IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/tsec_ltpa_and_keys_step1.html?lang=en

- b. Generate keys manually or automatically, and control the number of active keys. Follow the steps described in the WebSphere Application Server IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/tsec_ltpa_and_keys_step2.html?lang=en

Note: In this scenario, there are two separate WebSphere Application Server domains: the IBM BPM server and Worklight Server. They do not share LTPA keys; therefore, the IBM BPM server is unable to authenticate the LTPA token sent by the Worklight Server unless the same keys are exported from one server and imported in the other server as described in step c on page 220 and step d on page 220.

- c. Export the LTPA keys from the WebSphere Application Server in either the Worklight Server or IBM BPM server domain or cell.

Follow the steps documented in *Exporting Lightweight Third Party Authentication keys* at the following link:

http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/tsec_altpaexp.html?cp=SSEQTP_8.5.5%2F1-3-0-19-2-1&lang=en

When you export the LTPA keys you must provide a password that is required later to import the keys. The LTPA keys are exported to a file that must be readable in an ASCII editor like Notepad.

- d. Import the LTPA keys on the other cell or domain.

If the other cell is on a separate system, you must FTP the key file in binary format. To import keys, you must know the password that was used when the key file was exported to access the LTPA keys. After the keys are imported and saved, restart WebSphere Application Server.

For details about importing LTPA keys, follow the steps documented in *Importing Lightweight Third Party Authentication keys* in the IBM Knowledge Center:

http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.base.doc/ae/tsec_altpaimp.html?cp=SSEQTP_8.5.5%2F1-3-0-19-2-2&lang=en

7.4.2 Configuring the Worklight Server to authenticate using LTPA

This section describes how to modify the `authenticationConfig.xml` file under the `WL_BPM_Project` project that was created in scenario 1 in the `server/conf` folder to authenticate using the LTPA.

Perform the following steps:

1. Add the Worklight LTPA realm definition to the `<realms>` element.

In `authenticationConfig.xml`, uncomment the realm under the `<!-- For websphere -->` comment as shown in Example 7-1. In this example, the default realm name `WASLTPARealm` is used. The realm name must match through the configuration as shown in Example 7-3 on page 221, Example 7-11 on page 224, and Example 7-13 on page 227.

Example 7-1 Configuring LTPA authentication realm in authenticationConfig.xml

```
<!-- For websphere -->
    <realm name="WASLTPARealm" loginModule="WASLTPAModule">

<className>com.worklight.core.auth.ext.WebSphereFormBasedAuthenticator</className>
    <parameter name="login-page" value="/login.html"/>
    <parameter name="error-page" value="/loginError.html"/>
</realm>
```

2. Add the login module definition to the `<loginModules>` element.

Uncomment `loginModule` under the `<!-- For websphere -->` comment to obtain the text shown in Example 7-2.

Example 7-2 Configuring LTPA authentication loginModule in authenticationConfig.xml

```
<!-- For websphere -->
    <loginModule name="WASLTPAModule">
        <className>com.worklight.core.auth.ext.WebSphereLoginModule</className>
```

```
</loginModule>
```

3. Add security tests to authenticationConfig.xml as appropriate:

- Add webSecurityTest if you plan to develop for web environments.
- Add mobileSecurityTest if you plan to develop for mobile environments.

The mobileSecurityTest is used in this scenario to protect the mobile apps.

You can enable SSO between mobile applications from the mobileSecurityTest element by setting the value of the sso attribute to true in the testUser realm element, as shown Example 7-3.

Example 7-3 Configuring LTPA authentication mobileSecurityTest in authenticationConfig.xml

```
<mobileSecurityTest name="WAS-securityTest">  
    <testDeviceId provisioningType="none"/>  
    <testUser realm="WASLTPARealm" sso="true"/>  
</mobileSecurityTest>
```

4. Create and add the login.html and loginError.html files to the root directory.

Add the login.html and loginError.html files to the root directory in the Worklight Server WAR file to provide a way for the user to log in:

- Create the login.html and loginError.html files.
- Save the files to the root directory in the Worklight Server WAR file under the WebSphere Application Server installation, in file: {WAS_HOME}/profiles/{your profile}/installedApps/{your node}/{worklight EAR}/{worklight WAR}.

For Example:

C:\IBM\WebSphere\AppServer\profiles\wlprofile\installedApps\IBMBPMNode02Cell\IBM_Worklight_project_runtime_WL_BPM_Sc1config101414.ear\WL_BPM_Project.war

Example 7-4 shows an example of a login.html file.

Example 7-4 login.html example

```
<html>  
<head></head>  
<body>  
<form action="j_security_check" method="post">  
Username: <input type="text" name="j_username" size="20"><br>  
Password: <input type="password" name="j_password" size="20"><br>  
<input type="submit" value="Login">  
</form>  
</body>  
</html>
```

Example 7-5 shows an example of a loginError.html file.

Example 7-5 loginError.html example

```
<html>  
<head></head>  
<body>  
    LOGIN ERROR  
</body>  
</html>
```

5. Add the LTPA security test to the adapters procedures that invoke the IBM BPM REST APIs.

The security test used on the adapter procedures in scenario 1 is `FieldServiceSecurityTest`. This security test will be replaced with the LTPA security test, `WAS-securityTest` created in step 3 on page 221.

The adapters are:

- `OrderAdapter`
- `PartAdapter`
- `PartOrderAdapter`

Example 7-6, Example 7-7, and Example 7-8, show the three adapters with `WAS-securityTest` security test applied to their procedures.

Example 7-6 OrderAdapter.xml with “WAS-securityTest” on procedures

```
<procedure name="acceptOrder" securityTest="WAS-securityTest"></procedure>
<procedure name="getAssignedOrders" securityTest="WAS-securityTest">
</procedure>
<procedure name="getOrder" securityTest="WAS-securityTest"> </procedure>
<procedure name="getAssignedOrderDetails"
securityTest="WAS-securityTest"></procedure>
<procedure name="updateOrder" securityTest="WAS-securityTest"> </procedure>
```

Example 7-7 PartAdapter.xml with “WAS-securityTest” on procedure

```
<procedure name="searchParts" securityTest="WAS-securityTest"/>
```

Example 7-8 PartOrderAdapter.xml with “WAS-securityTest” on procedure

```
<procedure name="orderPart" securityTest="WAS-securityTest"/>
```

6. Add an LTPA cookie to adapters procedures that invoke the IBM BPM REST APIs.

To call the IBM BPM REST APIs using the LTPA Token, a cookie must be created with that token and passed to the IBM BPM server to invoke the REST APIs. The name of the cookie must be `LtpaToken2` and it is the shared WebSphere token between the Worklight Server and the IBM BPM server.

The `OrderAdapter` was created in scenario 1 and now needs to be modified to support LTPA.

The `PartOrderAdapter` is created in 7.6.5, “Creating the Parts Order adapter” on page 233 and LTPA must be enabled as shown in Example 7-10 on page 223.

In `OrderAdapter`, there are four procedures that need to be changed from the ones used in scenario 1. The Basic Authorization header will be removed and replaced with LTPA Token as shown in Example 7-9. The example shows the `acceptOrder` procedure implementation that passes the LTPA Token in the header, while invoking the `AssignToMe` IBM BPM REST API.

The other procedures of `OrderAdapter`, `getAssignedOrders`, `getClaimedOrders`, and `updateOrder` must be implemented as shown for `acceptOrder`.

Example 7-9 Adding header cookie with the LTPA Token to acceptOrder procedure of OrderAdapter

```
function getToken(){
    var token = WL.Server.getActiveUser().attributes.LtpaToken;
    var fulltoken = "LtpaToken2=" + token;
```

```

        WL.Logger.warn("**** This is the LTPA token: " + fulltoken);

        return fulltoken;
    }

    function acceptOrder(taskId) {
        WL.Logger.warn("acceptOrder: enter " + taskId);

        var input = {
            method : 'put',
            returnedContentType : 'json',
            path :
'/rest/bpm/wle/v1/task/'+taskId+'?action=assign&toMe=true&parts=none',
            headers: {"Cookie": getToken()}
        };

        WL.Logger.warn("acceptOrder: exit");

        return WL.Server.invokeHttp(input);
    }

```

In PartOrderAdapter, the orderPart procedure must pass a header cookie with the LTPA Token while invoking the createPart IBM BPM REST API, as shown in Example 7-10.

Example 7-10 Adding Header Cookie with the LTPA Token in orderPart procedure of PartOrderAdapter

```

function getToken(){
    var token = WL.Server.getActiveUser().attributes.LtpaToken;
    var fulltoken = "LtpaToken2=" + token;

    WL.Logger.warn("**** This is the LTPA token: " + fulltoken);

    return fulltoken;
}

function orderPart(partNumber, orderId, quantity) {
    WL.Logger.warn('orderPart ' + partNumber + ' ' + orderId + ' ' + quantity);

    var path = '/rest/bpm/wle/v1/service/FFA@orderParts?action=start&params=%7B' +
'%22partNumber%22:%22' + partNumber + '%22,' +
'%22quantity%22:%22' + quantity + '%22,' +
'%22workOrderNumber%22:%22' + orderId + '%22'+

'%7D&snapshotId=2064.85c6fa34-eee3-4280-a75b-9fed3bc231aa&createTask=false&parts=a
ll';

    WL.Logger.warn('path ' + path);

    WL.Logger.warn('Credentials: ' + WL.Server.getActiveUser().credentials);

    var input = {
        method : 'put',
        returnedContentType : 'json',
        path : path,
        headers: {"Cookie": getToken()}
    }

```

```
};
```

7.4.3 Configuring the Worklight mobile app to authenticate using LTPA

The following components of the mobile app must be configured to support LTPA authentication:

- Authentication challenge handler
- application-descriptor.xml

In this scenario, there are two client-side applications with separate authentication challenge handlers and application-descriptor.xml files.

Authentication challenge handler configuration

In scenario 1, the authentication challenge handler in the Field Service app is configured for adapter-based authentication, which invokes an adapter procedure to authenticate the user.

In this scenario, the authentication challenge handler in the Field Service app is configured for LTPA-based authentication, which uses a form-based authentication as shown in Example 7-11 where the WASLTPARealm is used in the WL.Client.createChallengeHandler method.

Since SSO is enabled between the two mobile applications, Field Service app and Parts app, as described in step 3 on page 221, you must check if the user is already authenticated in the challengeHandler.isCustomResponse method, using the WL.Client.isUserAuthenticated("WASLTPARealm") check.

If the user is already authenticated with the WASLTPARealm from the Field Service app or from the Parts app, the user is not challenged again to enter the credentials.

Example 7-11 Field Service app authenticationRealmChallengeProcessor.js

```
var loginIndicator = 'j_security_check';
var errorIndicator = 'LOGIN ERROR';
var challengeHandler = WL.Client.createChallengeHandler("WASLTPARealm");

challengeHandler.isCustomResponse = function(response) {
    console.log('===== isCustomResponse');
    var isAuth = WL.Client.isUserAuthenticated("WASLTPARealm");
    console.log('===== isAuth is ' + isAuth + ' current page is ' +
$.mobile.activePage[0].id);
    if (isAuth == true) {
        console.log('===== the user is authenticated in WASLTPARealm');
        $.mobile.changePage('#page-assigned-order', {transition : 'slide'});
    }
    if (!response || response.responseText === null) {
        return false;
    }
    var indicatorIdx = response.responseText.search(loginIndicator);
    var errorIdx = response.responseText.search(errorIndicator);

    if (indicatorIdx >= 0 || errorIdx >= 0){
        return true;
    }
    return false;
}
```



```

};

challengeHandler.handleFailure = function(response) {
    $('#message-invalid-login').show();
};

challengeHandler.handleChallenge = function(response){
    console.log('===== handleChallenge');
    $('#input-password').val('');
    setTimeout(function() {
        $('#message-invalid-login').hide();
        $.mobile.navigate('#page-login');
    }, 2000);
};

$("#link-login").bind('click', function () {
    console.log('===== doChallenge');
    var reqURL = 'j_security_check';
    var options = {};
    options.parameters = {
        j_username : $('#input-username').val(),
        j_password : $('#input-password').val()
    };
    options.headers = {};
    challengeHandler.submitLoginForm(reqURL, options,
challengeHandler.submitLoginFormCallback);
    $('#message-invalid-login').hide();
});

challengeHandler.submitLoginFormCallback = function(response) {
    console.log('===== submitLoginFormCallback');
    var isLoginFormResponse = challengeHandler.isCustomResponse(response);
    if (isLoginFormResponse){
        var errorIdx = response.responseText.search(errorIndicator);
        if(errorIdx >= 0) {
            challengeHandler.handleFailure(response);
        }
        else {
            challengeHandler.handleChallenge(response);
        }
    }
    else {
        challengeHandler.submitSuccess();
        $.mobile.navigate("#page-assigned-orders");
    }
};

```

In the same manner, the Parts app is configured for LTPA-based authentication, which uses a form-based authentication as shown in Example 7-12 on page 226 where the `WASLTPARealm` is used in the `WL.Client.createChallengeHandler` method.

Since SSO is enabled between the two mobile applications, Field Service app and Parts app, as described in step 3 on page 221, you must check if the user is already authenticated in the `challengeHandler.isCustomResponse` method, using the `WL.Client.isUserAuthenticated("WASLTPARealm")` check.

If the user is already authenticated with the WASLTPARealm from the Field Service app or from the Parts app, the user is not challenged again to enter the credentials.

Example 7-12 Parts app authenticationRealmChallengeProcessor.js

```
var loginIndicator = 'j_security_check';
var errorIndicator = 'LOGIN ERROR';
var challengeHandler = WL.Client.createChallengeHandler("WASLTPARealm");

challengeHandler.isCustomResponse = function(response) {
    console.log('===== isCustomResponse');
    var isAuthenticated = WL.Client.isUserAuthenticated("WASLTPARealm");
    console.log('===== isAuthenticated is ' + isAuthenticated + ' current page is ' +
$.mobile.activePage[0].id);
    if (isAuthenticated == true) {
        console.log('===== the user is authenticated in WASLTPARealm');
        $.mobile.changePage('#page-search-parts', {transition : 'slide'});
    }
    if (!response || response.responseText === null) {
        return false;
    }
    var indicatorIdx = response.responseText.search(loginIndicator);
    var errorIdx = response.responseText.search(errorIndicator);

    if (indicatorIdx >= 0 || errorIdx >= 0){
        return true;
    }
    return false;
};

challengeHandler.handleFailure = function(response) {
    $('#message-invalid-login').show();
};

challengeHandler.handleChallenge = function(response){
    console.log('===== handleChallenge');
    $('#input-password').val('');
    setTimeout(function() {
        $('#message-invalid-login').hide();
        $.mobile.navigate('#page-login');
    }, 2000);
};

$("#link-login").bind('click', function () {
    console.log('===== doChallenge');
    var reqURL = 'j_security_check';
    var options = {};
    options.parameters = {
        j_username : $('#input-username').val(),
        j_password : $('#input-password').val()
    };
    options.headers = {};
    challengeHandler.submitLoginForm(reqURL, options,
challengeHandler.submitLoginFormCallback);
    $('#message-invalid-login').hide();
});
```

```

challengeHandler.submitLoginFormCallback = function(response) {
    console.log('===== submitLoginFormCallback');
    var isLoginFormResponse = challengeHandler.isCustomResponse(response);
    if (isLoginFormResponse){
        var errorIdx = response.responseText.search(errorIndicator);
        if(errorIdx >= 0) {
            challengeHandler.handleFailure(response);
        }
        else {
            challengeHandler.handleChallenge(response);
        }
    }
    else {
        challengeHandler.submitSuccess();
        $.mobile.navigate("#page-search-parts");
    }
};

```

Logout

In the Field Service app, change the realm used in scenario 1 (realm="FieldServiceAuthRealm", see "Authentication configuration" on page 141) to the new realm WASLTPARealm for LTPA used in this scenario for LTPA authentication (see Example 7-13).

Example 7-13 Logout from WASLTPARealm

```

function doLogout(){
    WL.Client.logout('WASLTPARealm', {onSuccess: WL.Client.reloadApp});
}

```

application-descriptor.xml configuration

In the application-descriptor.xml file, specify the security test that your application must use for the appropriate environments.

In the Field Service app, the common and Android environments are used and protected with the WAS_securityTest. Example 7-14 shows the function that handles the event when the user clicks the log out button.

Example 7-14 Field Service app — descriptor.xml configuration

```

<common securityTest="WAS-securityTest"/>
<android version="1.0" securityTest="WAS-securityTest">
...
...
</android>

```

In the Parts app, the same configuration is applied as in the Field Service app as shown in Example 7-15.

Example 7-15 Parts app — descriptor.xml configuration

```

<common securityTest='WAS-securityTest' />
<android version="1.0" securityTest='WAS-securityTest'>
...
...

```

7.4.4 Testing the SSO configuration

After completing the configuration of the servers and the mobile applications, deploy the Worklight project and the applications on the WebSphere Application Server. Test the authentication and invocation of the adapters procedures with LTPA using a user created on WebSphere Application Server in step 2 on page 218.

7.5 Data sharing across apps implementation

In this section, the Field Service app is modified to store the shared data (order ID). The new Parts app is created to read the shared order ID and request for new parts based on this order ID.

7.5.1 Enabling Simple Data Sharing in the Field Service app

To share the order ID between the Field Service app and the Parts app, the Worklight Simple Data Sharing feature must be enabled in the `application-descriptor.xml` file for both apps.

Simple Data Sharing configuration is done for each environment separately. Android is the environment that is used in this example.

Perform the following steps to add Android phone and tablets to the Field Service app:

1. Open the `application-descriptor.xml` file for the Field Service app. Under the Android environment (Android phone and tablets), add the **Simple Data Sharing** feature.
2. Select **Enable Simple Data Sharing among an application family**.
3. Enter the application family in the Application family field. The application family must be the same for all the apps for which you want to enable data sharing. In this example, the application family is `com.ibm.itso.wlbp`. See Figure 7-9 on page 229.

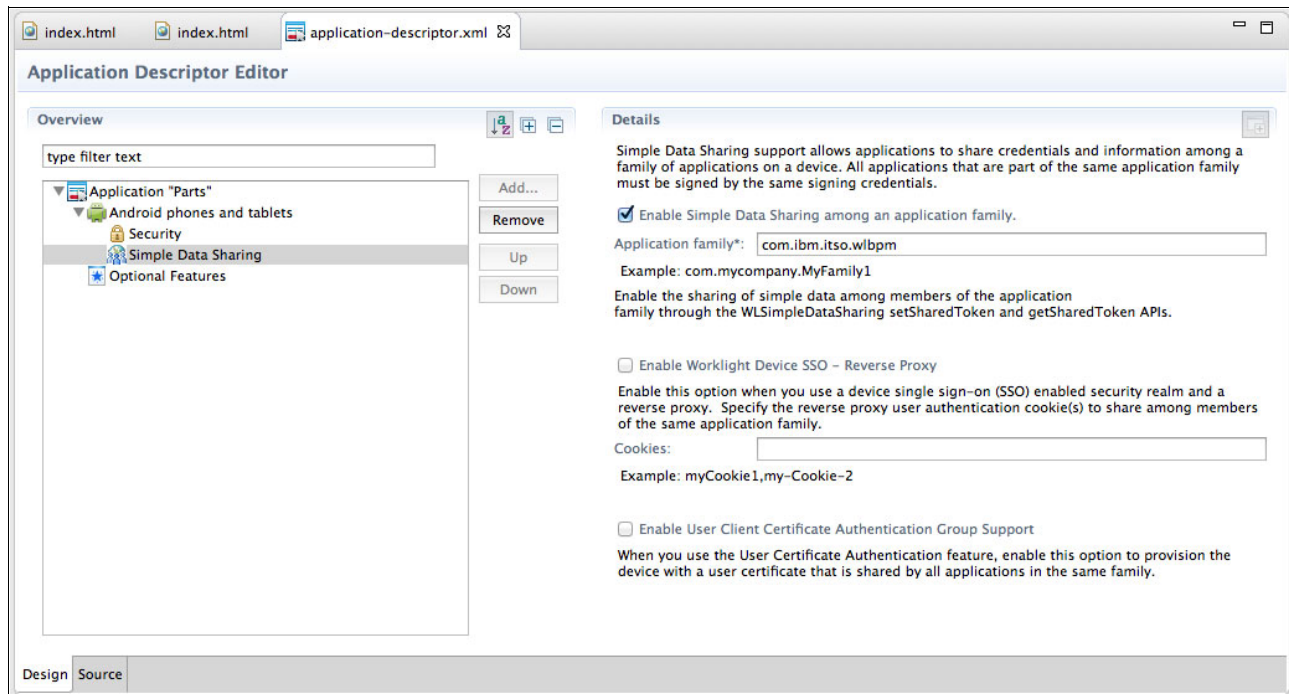


Figure 7-9 Enabling Simple Data Sharing in an Android environment

4. Save your changes.
5. Make sure to regenerate the Android environment by selecting the Field Service app and clicking **Run As** → **Build All Environments**.

7.5.2 Saving Order ID in the Field Service app

The Field Service app must be modified to capture the Order ID that the field technician starts working on. The order ID is captured when the field technician accepts the assigned order and the call to the procedure `acceptOrder()` in adapter `OrderAdapter` is successful.

Add a function that calls the API `WL.Client.setSharedToken` in the `main.js` file of the Field Service app. This function takes a key value pair to store the value as shared data. See Example 7-16.

Example 7-16 Saving simple shared data

```
function saveOrderInSharedData(orderId){
    WL.Logger.warn('Shared order Id: ' + orderId);

    WL.Client.setSharedToken({key: 'sharedOrderId', value:
orderId}).then(function(result){
        WL.Logger.warn("Value set");
    }).fail(function(errorObject){
        WL.Logger.warn(JSON.stringify(errorObject));
    });
}
```

This function must be called when accepting an order. Example 7-17 on page 230 shows the function `saveOrderInSharedData()` that is called when the call to the adapter to accept an order is successful.

Example 7-17 calling saveOrderInSharedData

```
//accept an assigned order
function acceptAssignedOrder(orderId, taskId){
    var invocationData = {
        adapter : "OrderAdapter",
        procedure: "acceptOrder",
        parameters: [taskId]
    };

    WL.Client.invokeProcedure(invocationData).then( function(response){
        //show confirmation dialog
        if(response.status == 200){
            WL.SimpleDialog.show("Alert", "Order accepted successfully!", [{text:
"OK", handler: null}], null);

            //save order Id in shared data
            saveOrderInSharedData(orderId);
        }

    }).then( function(response){

        return initJSONStore();

    }).then( function(response){

        return loadClaimedOrders();

    }).fail( function(errorObject){

        WL.Logger.warn(JSON.stringify(errorObject));

    });
}
```

7.6 Parts app implementation

This section describes the implementation of the second mobile application, the Parts app. Before creating the Parts app, the parts database in Bluemix must be created.

7.6.1 Creating the Parts database in Bluemix

IBM Bluemix is an open-standards, cloud-based platform for building, managing, and running apps of all types, such as web, mobile, big data, and smart devices. Capabilities include Java, mobile back-end development, and application monitoring, as well as features from ecosystem partners and open source, all provided as-a-service in the cloud.

Bluemix delivers a set of pre-built services and hosting infrastructure to host application and business logic for mobile and web developers. You must be registered in Bluemix in order to log in and create applications.

Follow these steps to create the Parts database in Bluemix:

1. Log in to Bluemix. Your dashboard is displayed. It includes the apps and services that you have created and their state.
2. In the Applications area of the dashboard, click **+** (plus sign) to create an application.
3. Click **Mobile Cloud** within the Boilerplates section. An application is created. The application is composed of:
 - Node.js for server-side JavaScript
 - Mobile Push for notifications
 - Mobile Application Security
 - Mobile Data

In this example, only the IBM Mobile Data for Bluemix service is used.

4. In the Catalog window that opens, enter a name for your Bluemix application. Call it **wl-mobiledata-yourname** and let the host name default to the same value. Then, click **Create**.
5. You will see the screen change as Bluemix creates, configures, and provisions your application. In about 10-15 seconds, you will see a notification in the web page indicating that your application and a set of services have been provisioned and a green check mark for App Health.
6. Upload to Bluemix the parts data for the Parts app. The parts data is retrieved by a Worklight adapter. A simple way to upload small amounts of data is in the form of a text file containing the data in JavaScript Object Notation (JSON) format.
7. Click the **Mobile Data Service**, and then select the **Manage Data** tab. Then, click the blue circle on the right.
8. In the Import Data dialog, browse to the location of the `parts-bluemix.json` file.
9. Enter the class name **Parts** and click **Import**.
10. Expand **Data Classes** (if not already expanded) and then expand **Parts** to see the parts data that has been imported.
11. After the data is loaded, Bluemix Mobile Data automatically exposes REST services that can be used to query this data.
12. As a final step, go back to the application Overview page and take note of these three values that are required to build the Worklight adapter that will query the parts data:
 - Routes: The host name to use in the adapter.
 - App ID: The application unique identifier.
 - App Secret: The secret token required to call the REST services.

7.6.2 Creating the Parts app

Add a Worklight hybrid application to the project WL_BPM:

1. In Worklight Studio, WL_BPM project, right-click the **apps** folder.
2. Click **New** → **Worklight Hybrid Application**.
3. Enter **Parts** as the application name.
4. Configure JQuery Mobile as described in section 6.4.1, “Worklight project” on page 128.

7.6.3 Enabling Simple Data Sharing in the Parts app

Follows the same steps described in 7.5.1, “Enabling Simple Data Sharing in the Field Service app” on page 228, but apply them to the Parts app.

Make sure that you enter exactly the same application family name `com.ibm.itso.wlbpm`.

7.6.4 Creating the Parts DB Bluemix adapter

Example 7-18 shows the PartAdapter created with Bluemix connection parameters, using the domain name of the application you created in Bluemix as described in 7.6.1, “Creating the Parts database in Bluemix” on page 230. Note that the port 443 is used since the call is authenticated as shown in the adapter code listed in Example 7-19.

Example 7-18 Adapter configuration to connect to Parts DB on Bluemix

```
<wl:adapter name="PartAdapter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wl="http://www.worklight.com/integration"
  xmlns:http="http://www.worklight.com/integration/http">

  <displayName>PartAdapter</displayName>
  <description>PartAdapter</description>
  <connectivity>
    <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
      <protocol>https</protocol>
      <domain>mobile.ng.bluemix.net</domain>
      <port>443</port>
    </connectionPolicy>
    <loadConstraints maxConcurrentConnectionsPerNode="2" />
  </connectivity>

  <procedure name="searchParts" securityTest="WAS-securityTest"/>

</wl:adapter>
```

Example 7-19 shows the code for connecting to the Bluemix Parts database. Note the usage of the application ID and the secret in the connection parameters. These are parameters provided in Bluemix when you create a new application to make a secure call.

Example 7-19 Adapter code to connect to Parts DB on Bluemix

```
var appID = 'APP_ID_XXXXX';
var secret = 'SECRET_XXXXX';

//call BlueMix to search the provided part name
function searchParts(partName) {

  predicate = '{ "predicate":{ "attributes":{"partName":"' + partName + '" } } }';

  WL.Logger.warn("predicate: " + predicate);

  var input = {
    method : 'post',
    returnedContentType : 'json',
    headers: {'IBM-Application-Secret':secret},
```



```

        body:{
            contentType:'application/json; charset=UTF-8',
            content: predicate,
        },
        path : 'data/rest/v1/apps/'+appID+'/query'
    };

    return WL.Server.invokeHttp(input);
}

```

7.6.5 Creating the Parts Order adapter

Create a PartOrder adapter to connect to IBM BPM server and create the order for the required parts. Connection parameters are the same as in scenario 1, described in 6.4.2, “Creating the adapters” on page 130. See Example 7-20.

Example 7-20 Adapter configuration to create a Part Order adapter

```

<wl:adapter name="PartOrderAdapter"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:wl="http://www.worklight.com/integration"
  xmlns:http="http://www.worklight.com/integration/http">

  <displayName>PartOrderAdapter</displayName>
  <description>PartOrderAdapter</description>
  <connectivity>
    <connectionPolicy xsi:type="http:HTTPConnectionPolicyType">
      <protocol>http</protocol>
      <domain>10.X.X.X</domain>
      <port>9080</port>
    </connectionPolicy>
    <loadConstraints maxConcurrentConnectionsPerNode="2" />
  </connectivity>

  <procedure name="orderPart" securityTest="WAS-securityTest"/>

</wl:adapter>

```

For creating an order, substitute the partNumber, orderId, and quantity in the URL shown in Example 7-21. Note that the LTPA token must be set as a cookie in the header.

Example 7-21 Calling the IBM BPM process to create a part order

```

//call to BPM to create an order for a part number
function orderPart(partNumber, orderId, quantity) {
    WL.Logger.warn('orderPart ' + partNumber + ' ' + orderId + ' ' + quantity);

    //substitute parameters in the path
    var path = '/rest/bpm/wle/v1/service/FFA@orderParts?action=start&params=%7B' +
        '%22partNumber%22:%22' + partNumber + '%22,' +
        '%22quantity%22:%22' + quantity + '%22,' +
        '%22workOrderNumber%22:%22' + orderId + '%22'+
        '%7D&createTask=false&parts=all';

    WL.Logger.warn('path ' + path);
}

```

```

WL.Logger.warn('Credentials: ' + WL.Server.getActiveUser().credentials);

var input = {
    method : 'put',
    returnedContentType : 'json',
    path : path,

    headers: {"Cookie": getToken()}
};

return WL.Server.invokeHttp(input);
}

//get LTPA token for the currently authenticated user
function getToken(){

    var token = WL.Server.getActiveUser().attributes.LtpaToken;
    var fulltoken = "LtpaToken2=" + token;

    WL.Logger.warn("This is the LTPA token: " + fulltoken);

    return fulltoken;
}

```

Parts Assist page

Besides the login page as in Field Service app, one page is created to display a search box and display the results. The field technician can order each result item by clicking the order button displayed below it.

Example 7-22 shows the HTML code for the Parts Assist page. The page ID is page-search-parts. It is a simple input and search button, and the unordered list tag (ul) place holder for the search results. The list has the ID search-results.

Example 7-22 index.html in Parts application

```

<div data-role="page" id="page-search-parts" data-theme="a">
    <div data-role="content">
        <div id="orders-header" data-role="header" data-position="fixed">
            <h1>Parts Assist</h1>
            <a id="button-logout" href="#" data-icon="power"
                class="ui-btn-right">Logout</a>
        </div>
        <label for="input-search">Search Parts:</label>
        <input type="search" name="input-search" id="input-search" value="">
        <a id="button-search" href="javascript:searchParts();" class="ui-btn
ui-btn-inline ui-icon-search ui-btn-icon-right">Search</a>
        <ul data-role="listview" id="search-results" data-inset="true">
            </ul>
        </div>
    </div>

```

In main.js, add the function searchParts(). The function reads the keyword that the user enters and submits it in the adapter call. See Example 7-23.

Example 7-23 Calling adapter to search for parts

```
//function to invoke adapter to search for a part number
function searchParts(partName){
    var partName = $('#input-search').val();
    WL.Logger.warn("Search part: " + partName);

    var invocationData = {
        adapter : "PartAdapter",
        procedure: "searchParts",
        parameters: [partName]
    };

    WL.Client.invokeProcedure(invocationData, {
        onSuccess: function(response){
            if(response.status == 200){
                updatePartsUI(response.invocationResult.object);
            }
        },
        onFailure: function(response){
            WL.Logger.warn(JSON.stringify(response.invocationResult));
        }
    });
}
```

Example 7-24 shows updating the UI with the list of parts returned from the search results. Note that there is an Order button created for each result item. The button invokes the function verifyPartOrderInfo passing the part number.

Example 7-24 Updating the UI with parts search results

```
//update search results
function updatePartsUI(parts){

    $("#search-results").empty();

    for ( var i = 0; i < parts.length; i++) {
        var li = document.createElement("li");
        li.innerHTML =
            "<h2>" + parts[i].attributes.partName + "</h2>" +
            "<div>" + parts[i].attributes.partDesc + "</div>" +
            "<div>" + 'Part Number: ' + parts[i].attributes.partNumber +
"</div>" +
            "<div>" + 'Price: ' + parts[i].attributes.partPrice + " USD
"</div>" +
            "<div>" + 'Availability: ' + parts[i].attributes.partAvail +
"</div>" +
            "<div><a href='javascript:verifyPartOrderInfo(" +
parts[i].attributes.partNumber + ");' class='ui-btn'>Order</a></div>";

        $("#search-results").append(li);
    }
}
```

```

    };

    $("#search-results").listview("refresh");
}

```

The function `verifyPartOrderInfo()` first retrieves the shared order ID using the `WL.Client.getSharedToken()` API.

Then, it displays a dialog box prompting the user for confirmation to create a part order with the retrieved order ID and the submitted part number.

Note the `globalWorkOrderId` and `globalPartNumber` that are declared as global variables to pass the values between the successive callbacks. See Example 7-25.

Example 7-25 Reading the shared order ID

```

//global variables to hold returned values between callbacks
var globalWorkOrderId;
var globalPartNumber;

//first ask for user confirmation on the order and part number,
//then call the backend to create an order
function verifyPartOrderInfo(partNumber){

    globalPartNumber = partNumber;

    WL.Client.getSharedToken({key:
    'sharedOrderId'}).then(function(sharedWorkOrderId){

        globalWorkOrderId = sharedWorkOrderId;

        WL.SimpleDialog.show(
            "Part Order",
            "Part with number " + globalPartNumber + " will be ordered for work
order " + globalWorkOrderId + ".",
            [
                {text: "OK", handler: function(){//callback when OK is clicked
                    //create an order
                    orderPart(globalPartNumber, globalWorkOrderId);
                }},
                {text: "Cancel", handler: null}//callback when Cancel is clicked,
            ],
            null);

    }).fail(function(errorObject){
        WL.Logger.warn(JSON.stringify(errorObject));
        alert(JSON.stringify(errorObject));
    });
}

```

When the field technician confirms the workOrder ID and part number, the adapter for Part Order is invoked with default quantity one item. See Example 7-26 on page 237.

Example 7-26 Calling adapter to create a Part Order

```
//function to invoke the adapters to create a part order
function orderPart(partNumber, workOrderId){
    var invocationData = {
        adapter : "PartOrderAdapter",
        procedure: "orderPart",
        parameters: [partNumber, workOrderId, '1'/*quantity*/]
    };

    WL.Client.invokeProcedure(invocationData, {
        onSuccess: function(response){
            if(response.status == 200){
                alert('Part ordered successfully!');
            }
        },
        onFailure: function(errorObject){
            alert(JSON.stringify(errorObject));
        }
    });
}
```

7.7 Extending scenario 1 with IBM BPM case management

Note: See Appendix A, “Samples included with this book” on page 307 for information about the samples provided with this IBM Redbooks publication.

Business process management and case management are two approaches to build a process. The type of business situation you are addressing determines which approach you use. Case management functions are only available if you have IBM BPM Advanced with the basic case management feature installed. IBM Business Process Manager provides tools to define and then improve a process: business process management and case management. Figure 7-10 on page 238 shows an overview of case management.

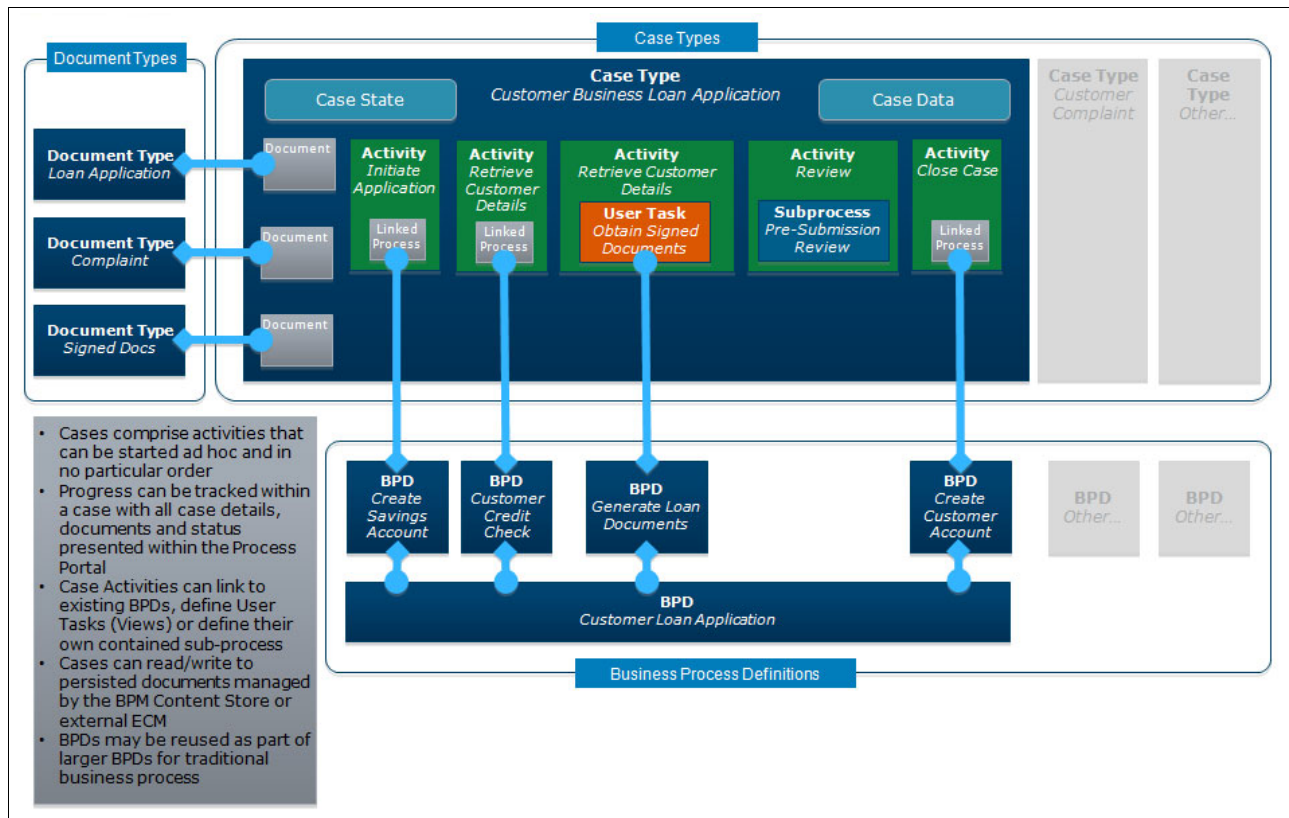


Figure 7-10 Overview of case management

Case management is applied to solutions with the following characteristics:

- ▶ Involve collaborative and ad hoc processes.
- ▶ Activities that are event-driven.
- ▶ Work is knowledge intensive.
- ▶ Content is essential for decision making.
- ▶ Outcomes are goal-oriented.
- ▶ The judgment of people impact how the goal is achieved.
- ▶ Process is often not predetermined.

Case management is most often used in the following situations:

- ▶ Addressing situations that cannot be easily handled by processes.
- ▶ It is very difficult or nearly impossible to design a process that addresses all possible situations.
- ▶ Optimal process steps cannot be determined due to too many dependencies on the process input.
- ▶ Knowledge workers require to examine each unique situation and use their judgment to determine what actions in the process to take next.
- ▶ Unstructured team work is required.
- ▶ Collaboration between experts is required to determine case outcome.
- ▶ Negotiations are required to determine process steps.

IBM BPM V8.5.5 introduces basic case management as a built-in feature of IBM BPM Advanced (*not* available in Standard). The features included are:

- ▶ Embedded, Enterprise Content Management repository to support basic case documents and folders.
- ▶ Can be extended with external IBM Enterprise Content Management to support unlimited content use cases.
- ▶ Web-based case designer tool.
- ▶ Designed for knowledge worker subject matter experts (SMEs).
- ▶ Integrated with IBM Process Designer and IBM Process Center.
- ▶ Case activities for ad hoc collaboration.
- ▶ Ad hoc activities can be implemented either as human tasks or processes.
- ▶ Configure ad hoc activity behavior: required, optional, pre-conditions.
- ▶ Case UI extensions to Process Portal.
- ▶ Case details instance viewer.
- ▶ Case folder and document viewer.
- ▶ Case work items viewers.
- ▶ Case search.
- ▶ Case task visibility via IBM BPM dashboards.

To start building cases, you first have to create a case type and one or more document types for the case.

A *document type* describes the types of documents that will be attached to a case. The reception (or upload) of a *document type* to the Enterprise Content Manager system is, most often, the trigger to start a new case type.

A *case type* is equivalent to an IBM BPM process definition (BPD) with the difference that it is not rigidly structured meaning that the activities in a case type are not wired together. Instead, they are run depending on actual conditions occurring in the case at run time.

Each case type can have any number of activities. Activities can be marked as required or optional. Required activities must be completed at some point before the case can be completed.

An activity can be implemented as a client-side human services, a linked process, or a subprocess:

- ▶ Client-side human services

This activity contains a flow with coach forms.

- ▶ Linked process

A case can call a business process through a linked process activity. When the linked process activity is triggered at run time, the linked process is run. After the linked process finishes, the calling activity is complete. Other cases can also reuse a linked process. For example, the steps for creating a customer account might be common to several different cases. If you group these steps together in a Create Customer Account process, you can use linked process activities. The linked process activities can call this process from all cases that require it.

► Subprocess

A subprocess is an option for encapsulating logically related steps within a case. Steps in a subprocess can directly access variables from the case. No data mapping is required. However, unlike a linked process, a subprocess can be accessed and instantiated only from the case that is using it. It is not reusable by any other case or process.

7.7.1 NewCustomerOrderCase case type overview

This section describes how to add some new features to the IBM BPM process defined in scenario 1 in section 6.6, “Building the business process” on page 167.

Before starting the use case implementation, you should become familiar with how to access the embedded Enterprise Content Management system because case management often depends on management of documents.

Note: You can access the embedded Enterprise Content Management system (IBM Content Foundation) with the URL:
`https://<bpm_server_hostname>:9443/acce/login.jsp`.

Figure 7-11 shows the login page for the embedded Enterprise Content Management system.

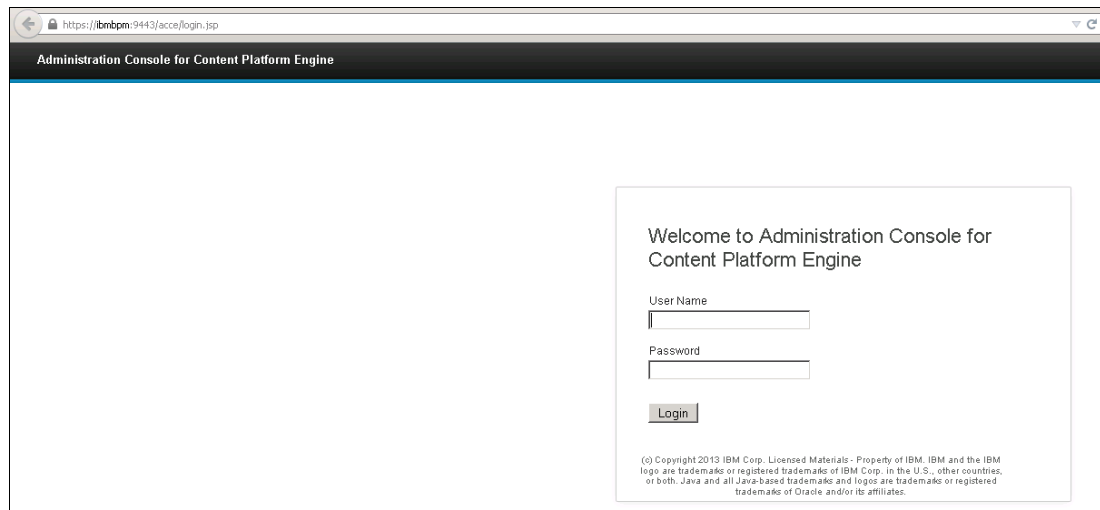


Figure 7-11 IBM Content Foundation login page

The following message is displayed when you log in:

This IBM Content Foundation instance is a limited use license in support of IBM BPM Advanced Edition. Consult the product license for limitations on what you are licensed to do before using this tool.

There are licensing restrictions to the embedded Enterprise Content Management system. Become familiar with the restrictions to make sure that you are compliant.

After successful login, the top-level page for the Administration Console for Content Platform Engine is displayed as shown in Figure 7-12 on page 241.

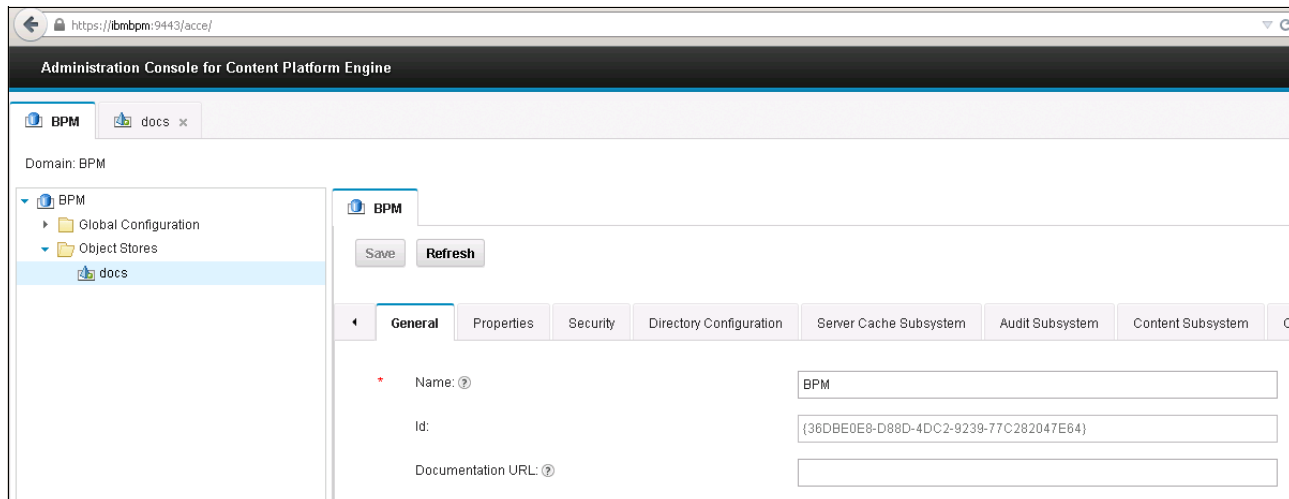


Figure 7-12 Top-level page in IBM Content Foundation

From here, you can navigate to the files that are contained in the embedded Enterprise Content Management system and you can manage those files, for example, you can delete and view the files.

Returning to the use case implementation for this scenario, the requirement for this use case is to support the start of a New Order Installation process triggered by uploading a document into the embedded IBM BPM Enterprise Content Management system instead of being created by the Call Center Staff manually using the Process Portal.

This scenario shows how to build a case in IBM BPM with the following sequence of activities:

1. The customer fills out an order form, signs it, and sends it to the Call Center. The Call Center scans the form into digital form and uploads it into the embedded Enterprise Content Management. Uploading a New Order Document document type starts the NewCustomerOrderCase (NCOC) case.
2. The first task of the NCOC case will be claimed by an employee in the CustomerAccounts team.
3. A member of the CustomerAccounts team processes the new order form received, for example, checks the customer's credit rating. The member of the CustomerAccounts team then decides whether a New Order Installation process described in 6.5, "IBM BPM processes supporting this scenario" on page 162 should be started. The CustomerAccounts team member indicates that the installation should proceed by checking a box on the coach form and completing it.
4. While the case is still active, the member of the CustomerAccounts team can upload and attach a CustomerSatisfactionSurvey document type that was received from the customer via mail. The CustomerAccounts team member scans the form and uploads it to the embedded Enterprise Content Management.
5. If a CustomerSatisfactionSurvey is uploaded, another activity (which in the sample NCOC case is a user task) is started automatically for an employee in the CustomerAccounts team to claim and complete. Uploading a CustomerSatisfactionSurvey document type outside of an active case does not trigger a new case or activity in this example.

To accomplish the goal of this scenario, a case called NCOC is created with two document types:

- ▶ **NewOrderDocument**: uploading this document type triggers a new case instance.
- ▶ **CustomerSatisfactionSurvey**: Uploading this document type triggers a new activity in an existing NCOC case instance.

Figure 7-13 shows the NCOC case and the document types in the case.

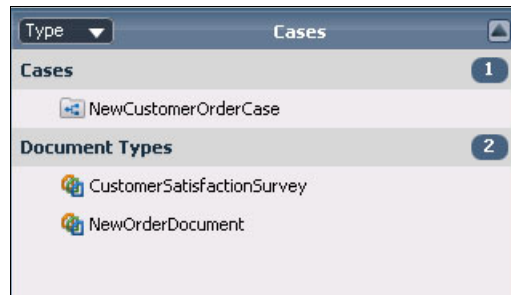


Figure 7-13 Top-level case artifacts

Perform the following steps using Process Designer:

1. Create the **NewOrderDocument** document type definition. Figure 7-14 shows the **NewOrderDocument** document type properties. These properties are metadata for each document stored in Enterprise Content Management. The data types are complex but they only contain a string attribute.

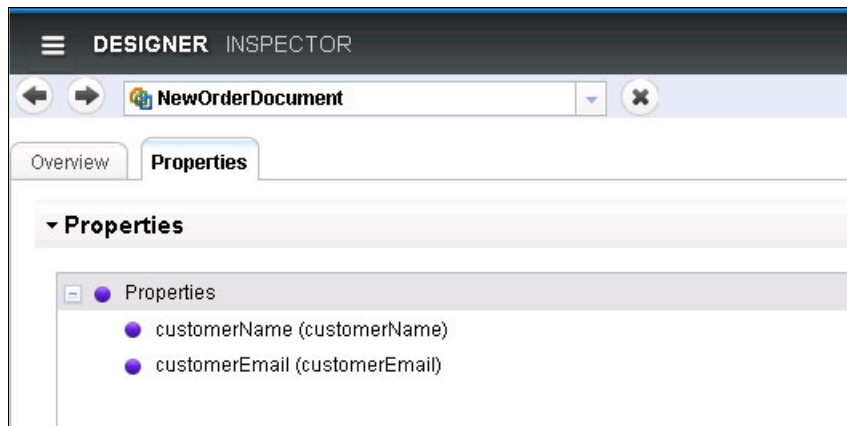


Figure 7-14 NewOrder Document document type

2. Create the **CustomerSatisfactionSurvey** document type definition. Figure 7-15 on page 243 shows the **CustomerSatisfactionSurvey** document type properties. The data types are complex but they only contain a string attribute.

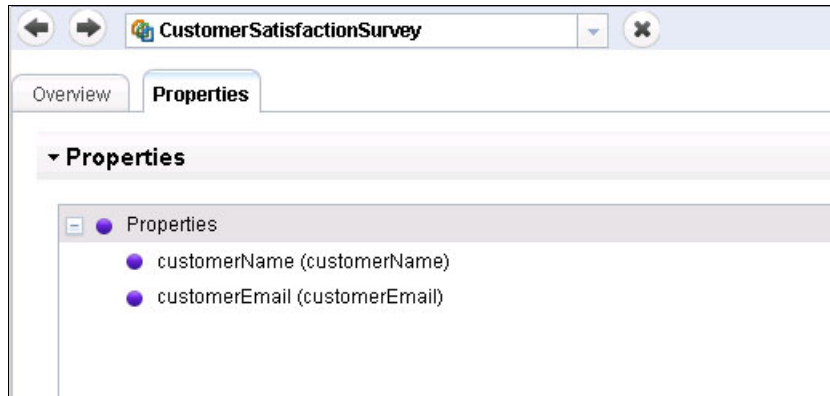


Figure 7-15 CustomerSatisfactionSurvey document type

3. Create the NewCustomerOrderCase case type as shown in Figure 7-16. The figure shows the Overview tab for the NewCustomerOrderCase case.

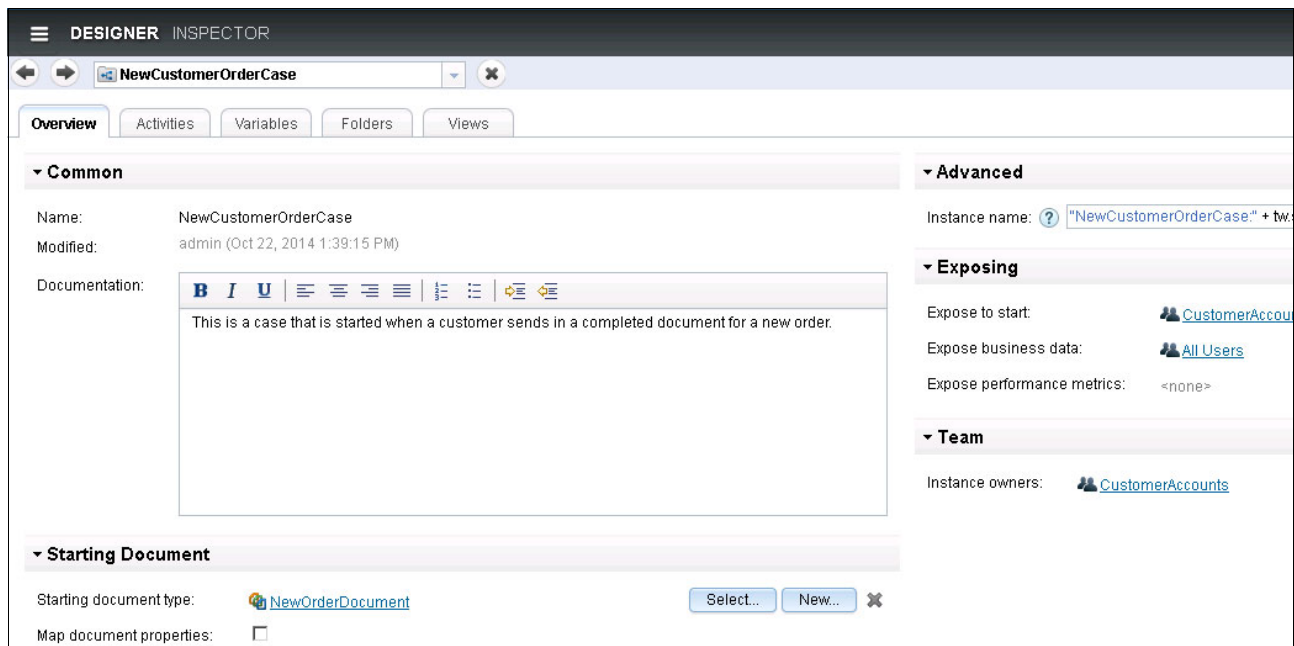


Figure 7-16 NewCustomerOrderCase Overview tab

There are some important parameters shown in Figure 7-16:

- ▶ Starting document type:

Specifies the type of document type that will start a case instance if that document type is uploaded (NewOrderDocument) into the embedded Enterprise Content Management. If you do not specify a document type, the case can still be started manually.

- ▶ Map document properties:

Select it to automatically initialize case folder property values with the values of matching document properties. If you select this check box, the properties of a starting document are matched to the properties of the case folder. Properties are matched based on the symbolic name, which is composed of the business object name, type, and cardinality. For example, assume that you have a business object that is called customerID of simple type string. Both the case type and the starting document type contain a property of type

customerID. When a starting document type is created, these properties are matched and the value of the customerID property in the case is updated with the value of the customerID property in the starting document type.

Map document properties are not selected in this example.

- ▶ **Expose to start:** Defines the team (CustomerAccounts in this example) whose members can start the case from the Process Portal or REST APIs.
- ▶ **Instance owners:** Defines the team (CustomerAccounts in this example) whose members can claim or work on the case instance.
- ▶ **Expose business data:** Defines which users (All Users in this example) can view business data displayed in Process Portal.
- ▶ **Expose performance metrics:** The team assigned in this parameter determines who can manage the process and its instances in the Process Performance dashboard. This parameter is left blank in this example.

Figure 7-17 shows all of the activities defined in the case. Notice that they are grouped into required activities and optional activities. And, they are not wired together. They can occur in an ad hoc manner depending on what dynamically occurs in the case instance at any time.

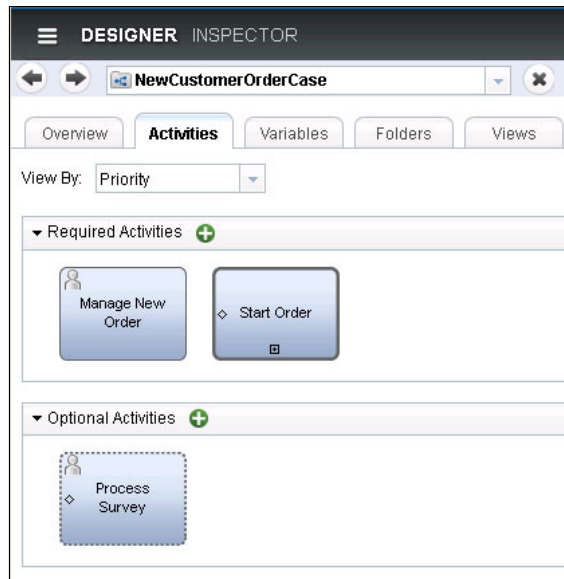


Figure 7-17 NewCustomerOrderCase activities

Figure 7-18 on page 245 shows the variables defined for the case in the Variables tab. In this example, only the Case Folder Properties variables are defined.

These variables are global to all activities in the case whereas the Input, Output, and Private variables are mapped to activity variables. The case folder properties are addressed in JavaScript with the prefix `tw.local.` just like input, output, and private variables.

A case folder property can reference only a custom simple type business object. It cannot reference a complex type business object or one of the simple types, such as a string, directly.

Also, case folder properties have display and symbolic names. The display name field shows a generated name that is based on the property type. This name appears on the case folder in the IBM BPM content store. If you create a case folder property that is called `customerCardNumber` and `customerCardNumber` is based on a simple type that is called `CustCardNumber`, then the display name field becomes `Cust Card Number`. In other words,

the simple type name becomes the display name with a slight change for readability. The Symbolic name field shows a generated name that is based on the property type. You can use this name for Enterprise Content Management operations.

The Visible Variables section shows that some variables can be made visible in Process Portal as business data. The Visible Variables section shown in Figure 7-18 is used to mark certain variables that will be visible to a user in the Process Portal.

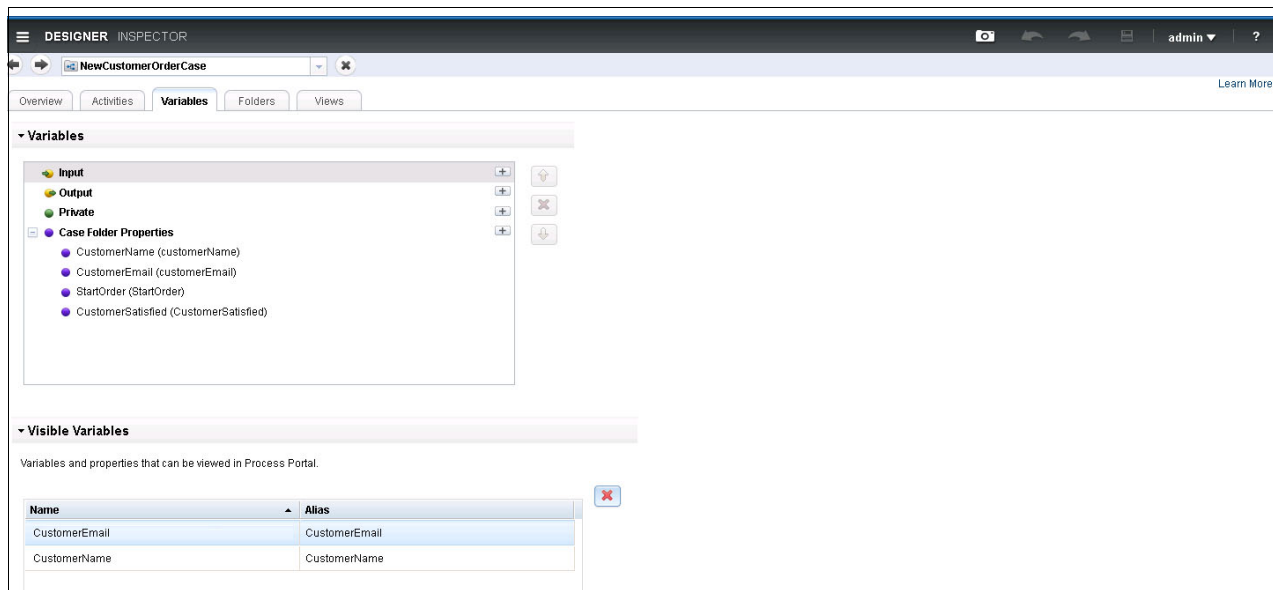


Figure 7-18 *NewCustomerOrderCase* variables

Figure 7-19 shows the document folders defined for this case.

Document folders are a convenience for the user completing the case because cases are normally document-centric. Since it is possible that many documents will be uploaded and attached to a case instance before it is completed (even document types that do not trigger an activity), document folders provide a way to organize these documents logically.

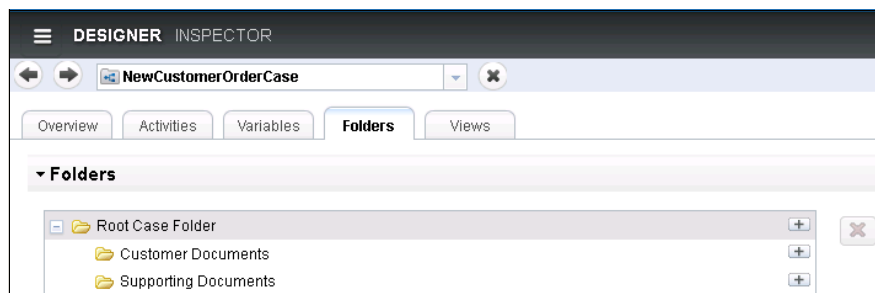


Figure 7-19 *NewCustomerOrderCase* document folders

You can see that the document folders defined in your case are duplicated in the embedded Enterprise Content Management. Notice that the case documents are stored automatically in subdirectories based on the date and time that the case instance was started. Figure 7-20 on page 246 shows the duplicated document folders in the embedded Enterprise Content Management.

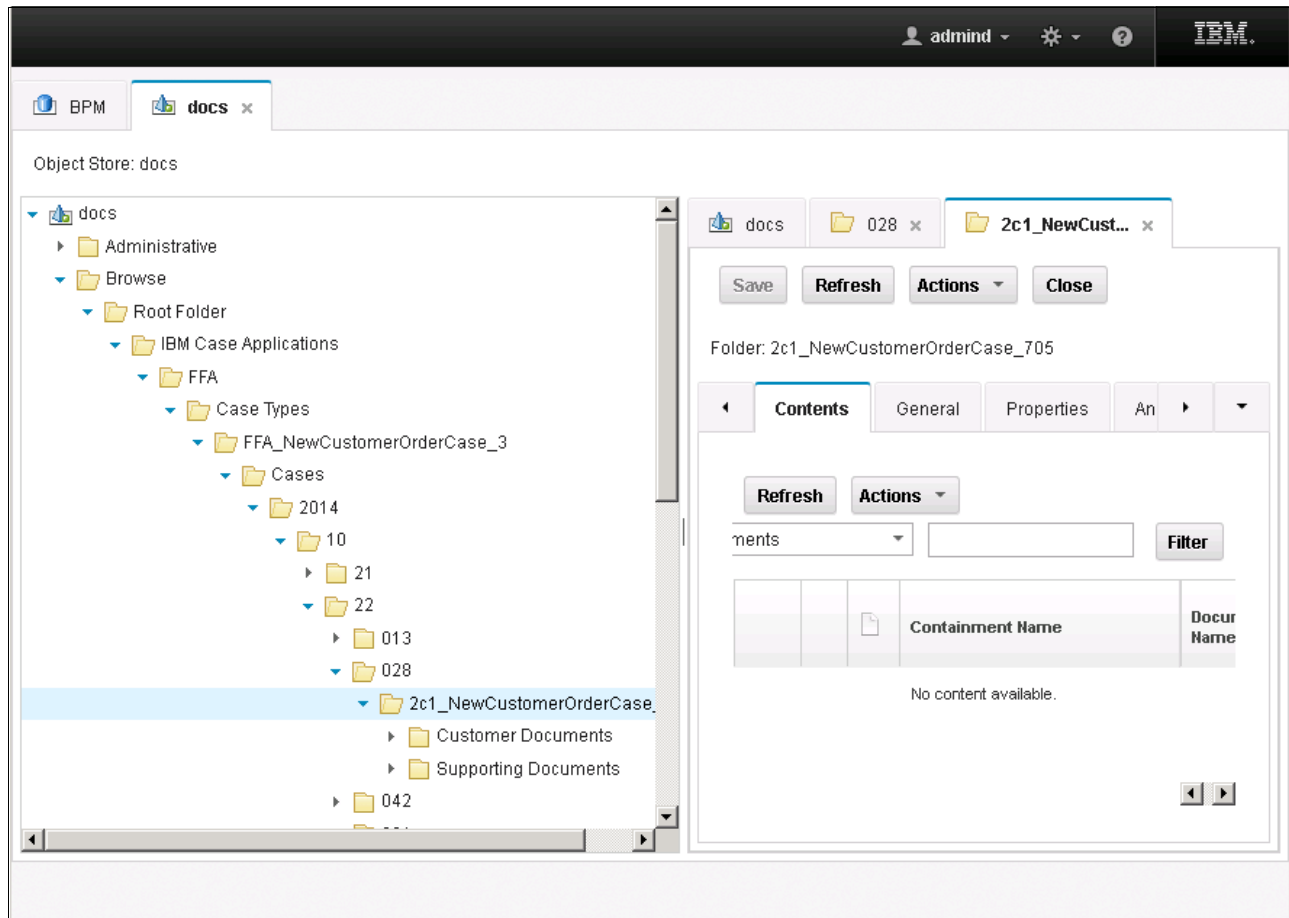


Figure 7-20 Document folders in Enterprise Content Management

7.7.2 Activities in NewCustomerOrderCase

This section discusses the set of parameters in the NCOC activities:

- ▶ Manage New Order
- ▶ Start Order
- ▶ Process Survey

Highlighting an activity displays four tabs at the bottom of the editor:

- ▶ General
- ▶ Implementation
- ▶ Data Mapping
- ▶ Preconditions

The following sections walk through each tab for the activities in this scenario.

Manage New Order activity

Figure 7-21 on page 247 shows the General tab for the Manage New Order activity.

General	
Implementation	
Data Mapping	
Preconditions	

Common

Name: Manage New Order
Documentation:

Behavior

How is the activity started?
☒ Automatically by the system
☐ Manually by the user
Does the activity have to be completed?
☒ Yes. The activity is **required**
☐ No. The activity is **optional**
☐ Repeatable. The activity can be invoked multiple times
☐ Hidden. This is a background activity that users will not see

Figure 7-21 Manage New Order HS General tab

The Behavior section in the General tab defines the behavior of the activity:

- ▶ Define how the activity is started:
 - Automatically: The activity starts automatically when a case is created. If a precondition is defined, the precondition must also be met in order for the case to start automatically.
 - Manually: The activity must be started by a user. You can define preconditions that must be met to put a manual activity into Ready state. However, the activity does not start until the caseworker decides to manually start the activity.

- ▶ Define whether the activity is required or optional.

If you change this setting, the activity moves into the appropriate section.

- ▶ Repeatable indicates that the activity can be invoked multiple times.

You can mark an activity to repeat when there is a property change or when a document-filing precondition is defined for the activity. An activity that is marked as repeatable can occur multiple times during the lifetime of the case it is part of and can cause new activities to be created and repeated. Activities can be repeated as needed, even if the activity has already completed.

- ▶ Hidden indicates that the activity is hidden from users in Process Portal.

If the activity is a background activity that users should not see, select Hidden. Hidden activities can be started automatically or manually as soon as the case is created or after the preconditions are met for the activity. The activity does not appear in the Activities section in the Process Portal, but assigned users can view and work with resulting tasks in the Process Portal task lists. Typically, hidden activities are automatic but they can also be manual. A manual hidden activity can be started only programmatically, by using the `ActivityInstance.start()` method. For more information, see *JavaScript API for IBM Process Designer* at the following link:

http://www-01.ibm.com/support/knowledgecenter/api/content/SSFPJS_8.5.5/com.ibm.wbpm.ref.doc/ae/doc/JSAPI.html

Also see *REST API for the Activity Instance (Ad Hoc) Resource* at the following link:

http://www-01.ibm.com/support/knowledgecenter/api/content/SSFPJS_8.5.5/com.ibm.wbpm.ref.doc/rest/bpmrest/rest_bpm_wle_v1_activity_instanceid.htm

Figure 7-22 on page 248 shows the Implementation tab for the Manage New Order activity in this scenario. This activity is implemented as a user task named Manage New Order HS.

Figure 7-22 Manage New Order HS Implementation tab

In the Implementation tab, you can choose from one of the following implementations:

- User task

A task that is performed by a user in Process Portal. The user task is implemented as a client-side human service, which creates the user interface.

Note: In a case type, all user task implementations are done with client-side human services. A client-side human service runs in the browser. You can select an existing client-side human service, or create a new one. When you create a new client-side human service, the input and output variables are automatically taken from the case type variables. All input and output variables are selected by default. You can clear the input and output variables that you do not want to use with your human service.

You can select an existing client-side human service, or create a new one. When you create a new client-side human service, the input and output variables are automatically taken from the case type variables. All input and output variables are selected by default. You can clear the input and output variables that you do not want to use with your human service.

- Linked process

Calls another process (BPD) within the process application.

Note: You cannot create a linked process from the case type editor. To create a linked process, start the Process Designer desktop editor

- Subprocess

This is a non-reusable process meaning other activities cannot reuse it.

Important: If you define a repeatable activity that is implemented as a subprocess, multiple instances of the subprocess might be active at the same time (for example, when multiple documents arrive within a short period). These instances concurrently access the same Case Folder Properties and can therefore interfere with each other. Therefore, when implementing repeatable activities and subprocesses, avoid modifying the Case Folder Properties (reading them is not an issue).

In the Implementation tab Priority Settings, specify your settings:

- Priority: Establishes the priority that you want for this activity in the Process Portal. It ranges from lowest to highest.

- **Due in:** Specifies the time when the activity must be completed from the time the activity begins, and the time zone that is used by the Due in field.
- **Task Header:** Specify a display name for the activity in the Subject field. If you do not specify a display name, the activity name is used. In the Narrative field, you can enter a description or instructions for this activity. You can use JavaScript to specify these fields.

Figure 7-23 shows the Data Mapping tab for the Manage New Order activity.

Figure 7-23 Manage New Order Data Mapping tab

For client-side human services and linked processes, you must map the case variables to the client-side human service or linked process variables. Subprocesses have direct access to the case variables and therefore do not need data mapping.

Data mapping maps the case type input and output variables to the input and output variables of the client-side human service or linked process. In some client-side human services, the mapping is complete since the wizard to create a human service creates the mapping as one of its steps.

Figure 7-24 shows the Preconditions tab for the Manage New Order activity.

Figure 7-24 Manage New Order HS Preconditions tab

You can specify preconditions that must be met before the activity is ready to start. Activities can start automatically after all the preconditions are met or manually by a user after all the preconditions are met. If you do not set any preconditions, automatic activities start as soon as the case is launched and manual activities must be started by a user.

A precondition consists of two parts:

- A precondition event.
- A precondition expression.

The two parts together determine when the activity starts:

- If both are defined, then when the precondition event occurs, the precondition expression is evaluated. If the expression evaluates to true, the activity starts.
- If there is a precondition event but no precondition expression, the activity starts when the precondition event is met.

Note: In addition to determining when the activity first starts, the precondition also determines when an activity that is already started changes state. For example, from a waiting state to a working state.

In the Precondition Event section, select the event that triggers the precondition expression to be evaluated as follows:

- ▶ No precondition event for this activity: Automatic activities start as soon as the case is launched. Manual activities must be started by a user.
- ▶ A document is filed in the case: The activity starts when a document is added to the case.
 - Any document type: The activity starts when a document is added to the case. This option applies to all documents, including documents that are not contained in this process application.
 - Choose one or more document types: The activity starts when a document of any of the specified document types is added to the case.

Inside the implementation of the activity, you can use JavaScript to access the identifier of the document that caused the activity to start:

```
tw.system.currentAdHocActivityInstance.enablingDocumentID
```

You can use this identifier to further process the document within the activity by using Enterprise Content Management operations.

- ▶ A case property or variable is updated: You can select multiple case properties or variables from the list provided. The activity starts when any of the specified properties or variables are updated.
- ▶ A precondition expression is met: There is no precondition event to be triggered. You must specify a precondition expression, and when the expression is met, the activity starts.

In the Precondition Expression section, create an expression if your precondition requires it. The expression must evaluate to true at the time the precondition event occurs, for the activity to start. To create an expression:

1. Click the + icon in the Precondition Expression section heading. This task cannot be performed if No precondition event for this activity is selected.
2. Specify the parameters of the expression. For example, you might specify `claimAmount is greater than 100`. You can specify multiple expressions for the precondition. For example, you might specify `creditCardNumber is not equal to 0` and `vendorName is not like Unknown`.
3. Select Match All if both expressions must evaluate to true for the activity to start. Select Match Any if the activity can start when any one expression evaluates to true.

Start Order activity

Figure 7-25 shows the General tab for the Start Order activity in this scenario.

The screenshot displays the 'General' tab for the 'Start Order' activity. On the left, a sidebar contains tabs for 'General', 'Implementation', 'Data Mapping', and 'Preconditions'. The 'General' tab is selected. The main area is divided into two sections: 'Common' and 'Behavior'. In the 'Common' section, the 'Name' field is populated with 'Start Order', and the 'Documentation' field is empty with a rich text editor toolbar. The 'Behavior' section on the right contains several configuration options: 'How is the activity started?' with 'Automatically by the system' selected; 'Does the activity have to be completed?' with 'Yes. The activity is required' selected; a 'Repeatable' checkbox which is checked; and a 'Hidden' checkbox which is unchecked. A help icon is visible next to the 'Repeatable' checkbox.

Figure 7-25 Start Order General tab

Figure 7-26 on page 251 shows the Implementation tab for Start Order activity.

Figure 7-26 Start Order Implementation tab

The Start Order Data Mapping tab is not included in this document because it is blank since no variables need to be mapped. The reason that no variables need to be mapped is because the New Order Installation BPD has no Input or Output variables defined.

Figure 7-27 shows the Preconditions tab for the Start Order activity.

Figure 7-27 Start Order Preconditions tab

The expression tests the case folder property StartOrder. When StartOrder is set to the string YES, the activity starts and at that point the New Order Installation BPD is started.

Process Survey activity

Figure 7-28 shows the General tab for the Process Survey activity.

Figure 7-28 Process Survey General tab

Figure 7-29 shows the Implementation tab for the Process Survey activity.

Figure 7-29 Process Survey Implementation tab

Figure 7-30 shows the Data Mapping tab for the Process Survey activity.

Figure 7-30 Process Survey Data Mapping tab

When this client-side human services finishes, the case folder property CustomerSatisfied, is set to either YES or NO.

Figure 7-31 shows the Preconditions tab for the Process Survey activity.

Figure 7-31 Process Survey Preconditions tab

This activity only starts if a CustomerSatisfactionSurvey document type is uploaded before the case instance completing. Because this activity is defined as non-repeating, only one survey can be uploaded for each case instance.

Since this activity is also defined as optional, a CustomerSatisfactionSurvey document type is not required to be uploaded.

A CustomerSatisfactionSurvey document type can be uploaded from the Manage New Order activity using the Document Explorer coach view included on the Manage New Order HS coach form.

7.7.3 User tasks (human services) implementation

This section describes how the user tasks (human services) are defined in this scenario. There are two user tasks:

- ▶ Manage New Order HS (which is the implementation for the Manage New Order activity)
- ▶ Process Sat Survey HS (which is the implementation for the Process Survey activity)

Manage New Order HS

Figure 7-32 on page 253 through Figure 7-37 on page 256 show the configuration of Manage New Order HS in this scenario. The Overview tab definition is not included because all defaults are selected in this example.

Figure 7-32 on page 253 shows the Manage New Order HS Diagram tab.

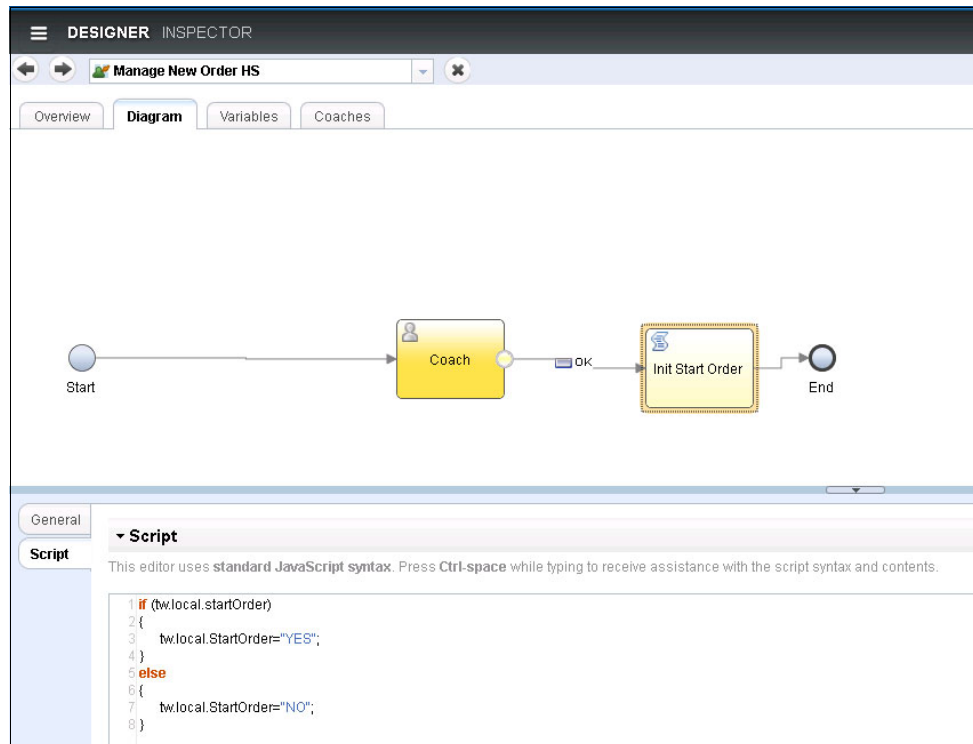


Figure 7-32 Manage New Order HS Diagram tab

This is a simple diagram. The Script step is included to make sure that the StartOrder case folder property is correctly set. As shown in Figure 7-27 on page 251, the precondition for the Start Order activity tests the StartOrder property.

Figure 7-33 shows the Manage New Order HS Variables tab.

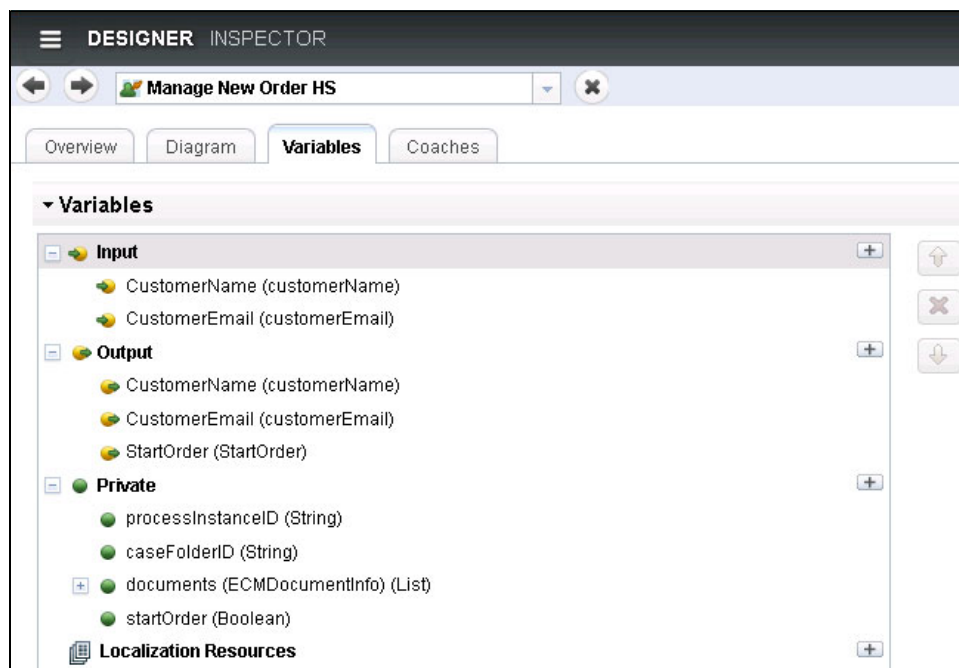


Figure 7-33 Manage New Order HS Variables tab

These variables are used as follows:

- ▶ `processInstanceId`: This variable is mapped to the Instance ID configuration parameter in the Document Explorer CV.
- ▶ `caseFolderID`: This variable is mapped to the Folder ID configuration parameter in the Document Explorer CV.
- ▶ `documents`: This variable is bound to the DocumentExplorer coach view used in the coach.
- ▶ `startOrder`: This variable is bound to the Checkbox coach view on the coach to determine whether the New Order Installation process should be started.

Figure 7-34 shows the Manage New Order HS Coaches tab.

Coaches

Customer Name

Customer Email

Documents

▼ Documents +

Documents > Correspondence > i

Correspondence
Merrisa Lee July 1, 2013

Supporting Documents
Sam Johnson June 25, 2013

Dispute Complaint
Sam Johnson July 2, 2013

Document Viewer

Document Viewer

Messageboard	Number
Selfboard	Selfboard Win
Shark Request	Self Dispute Book
Selfboard Win	Dispute Flip Flip Sample
Self Dispute Book	Shark Request
Dispute Flip Flip Sample	Selfboard Win

Open in new window

Start Order

☐ Yes ☒ No

OK

Figure 7-34 Manage New Order HS Coaches tab

Both the Document Explorer and the Document Viewer coach views are from the Content Management toolkit supplied with the IBM BPM product.

In the Documents section, you can upload as many documents for this case as needed. The objective is to provide a way for a user to upload a CustomerSatisfactionSurvey document type while the case is still open. In this example, this feature is implemented by providing a file upload feature in this coach form. If a CustomerSatisfactionSurvey document type is uploaded, it triggers the Process Sat Survey HS, which is the implementation of the Process Survey activity.

Figure 7-35 shows the Document Explorer coach view General tab.

Figure 7-35 Manage New Order HS Document Explorer coach view General tab

Figure 7-36 shows the Document Explorer coach view Configuration tab.

Figure 7-36 Document Explorer coach view Configuration tab

You can set the folder ID and the instance ID to your local private variables processInstanceID and caseFolderID respectively.

- Folder ID

Displays the list of documents and folders for the folder ID. If this property is configured, the instance ID is ignored. If the Folder ID property is not configured, the folder ID is derived from the associated Instance ID configuration.

If an instance ID is not specified, the folder ID is derived from the human service context. For client-side human services, the `tw.system.processInstance.id` and `tw.system.processInstance.caseFolderId` properties are used for the context. For heritage human services, `tw.system.currentProcessInstance.id` and `tw.system.currentProcessInstance.caseFolderId` properties are used for the context.

► Instance ID

Displays the list of documents and folders for the instance ID. If the Folder ID property is configured, the Instance ID configuration is ignored.

If neither the Folder ID or the Instance ID properties are configured, the folder ID is derived from the human service context. For client-side human services, the `tw.system.processInstance.id` and `tw.system.processInstance.caseFolderId` properties are used for the context. For heritage human services, `tw.system.currentProcessInstance.id` and `tw.system.currentProcessInstance.caseFolderId` properties are used for the context.

Figure 7-37 shows the Document Viewer coach view General tab.

The screenshot shows the configuration for the Document Viewer coach view. It is divided into two main sections: 'Common' and 'Behavior'.
In the 'Common' section, the 'Label' is set to 'DocumentViewer', the 'Help' field is empty, and the 'Control ID' is 'Document_Viewer1'.
In the 'Behavior' section, the 'Binding' is 'documents.listSelected' with a link to '(ECMDocumentInfo)'. There are 'Select...' and 'Clear' buttons next to it. The 'View' is 'Document Viewer (ECMDocumentInfo) Content Management' with 'Select...' and 'New...' buttons. The 'Label visibility' is set to 'Hide' with a dropdown arrow.

Figure 7-37 Document Viewer coach view General tab

Make sure that you bind your `documents.listSelected` variable to the coach view.

If you select a document from the Document Explorer coach view, the contents of the document will be viewable in the Document Viewer coach view.

Process Sat Survey HS

Figure 7-38 on page 257 through Figure 7-43 on page 260 show the configuration of Process Sat Survey HS in this scenario.

Figure 7-38 on page 257 shows the Process Sat Survey HS Overview tab.

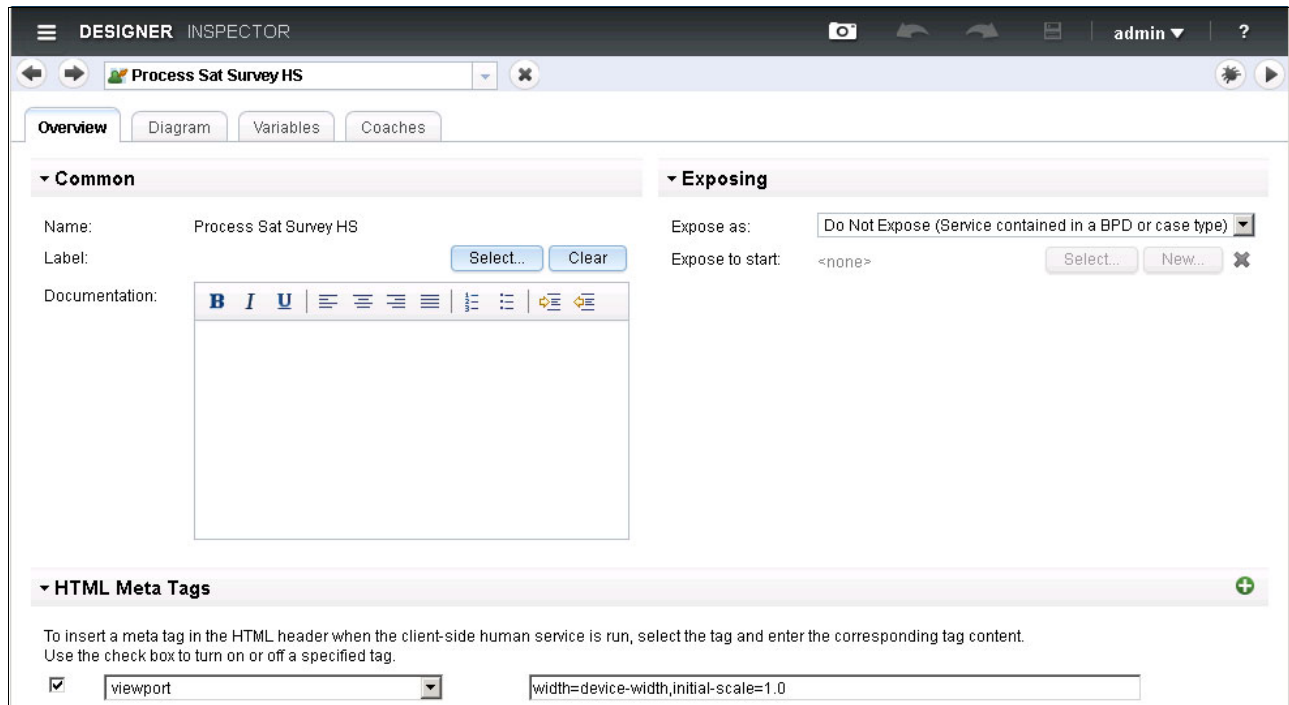


Figure 7-38 Process Sat Survey HS Overview tab

Figure 7-39 on page 258 shows the Process Sat Survey HS Diagram tab with the Is Cust Satisfied script highlighted.

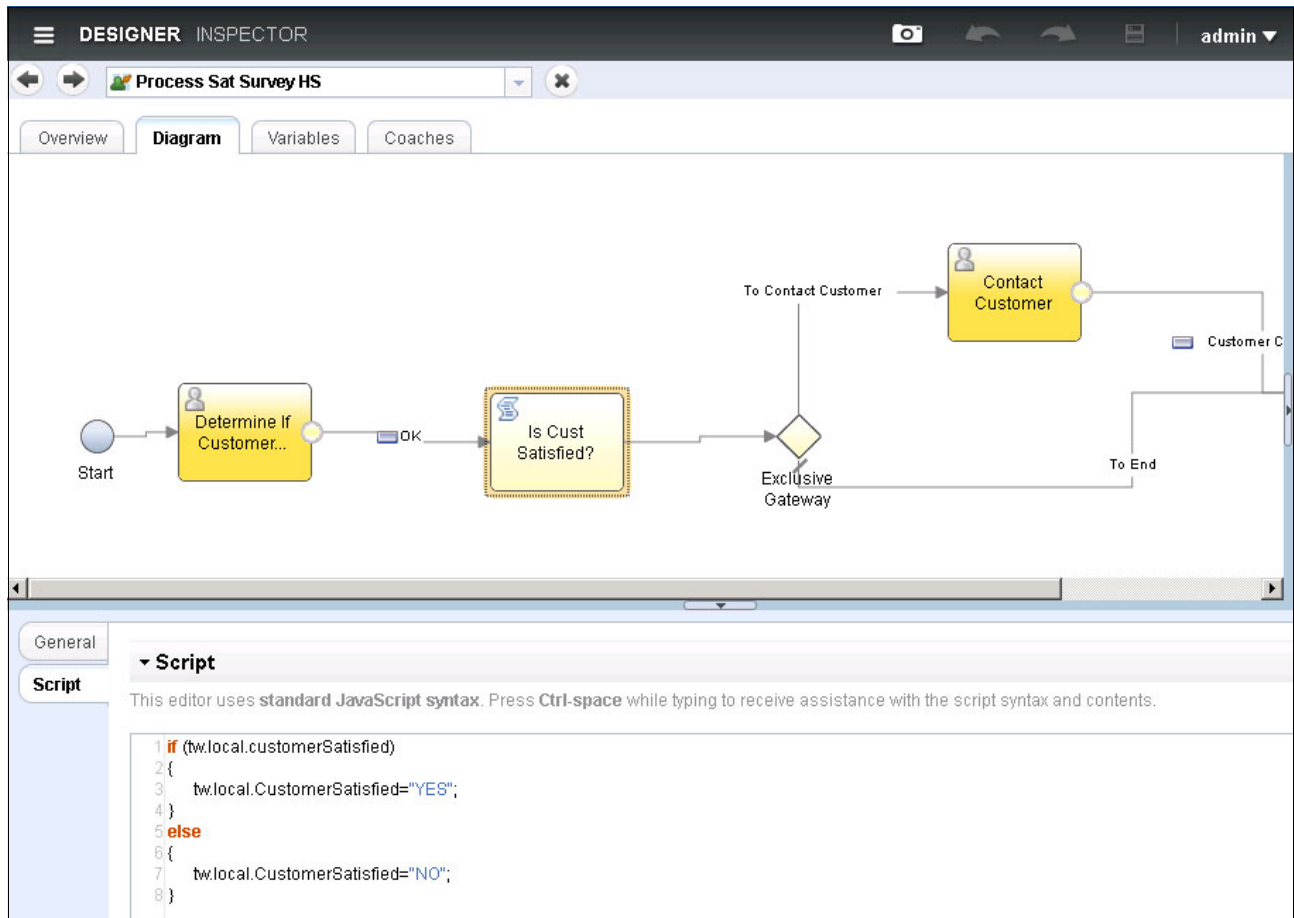


Figure 7-39 Process Sat Survey HS Diagram tab with the Is Cust Satisfied script implementation

The Is Cust Satisfied? script step sets the case folder property CustomerSatisfied.

Figure 7-40 on page 259 shows the Process Sat Survey HS Diagram tab with the Exclusive Gateway implementation highlighted.

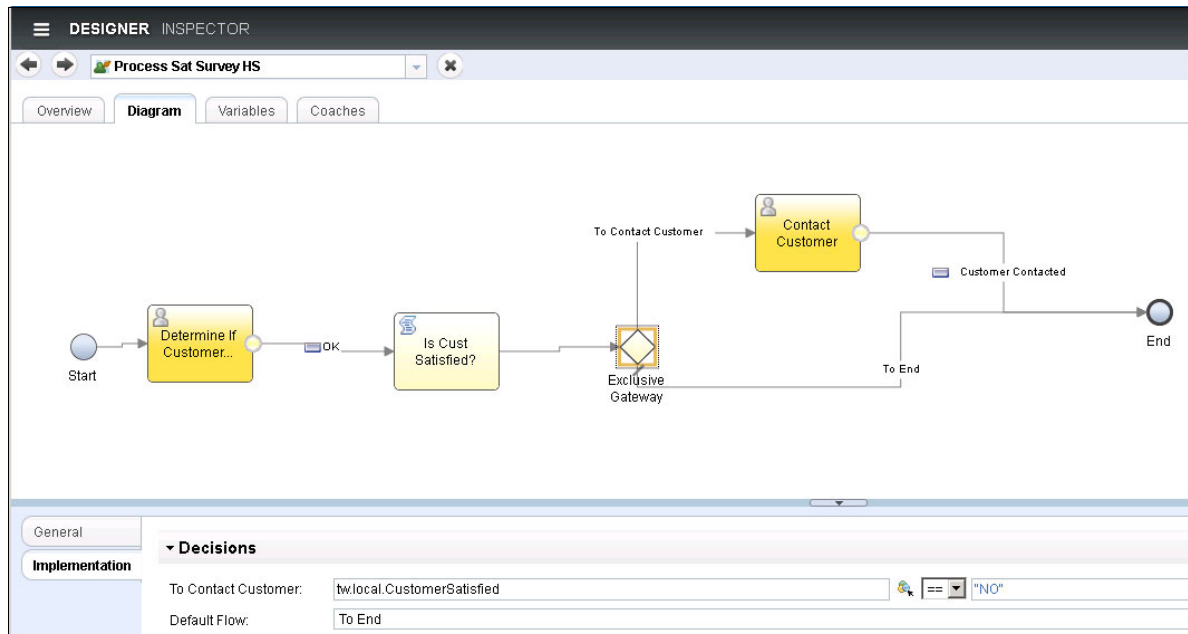


Figure 7-40 Process Sat Survey HS Diagram tab with the Exclusive Gateway implementation

The case folder property `CustomerSatisfied` is used in this example to branch appropriately in the flow.

Figure 7-41 shows the Process Sat Survey HS Variables tab.

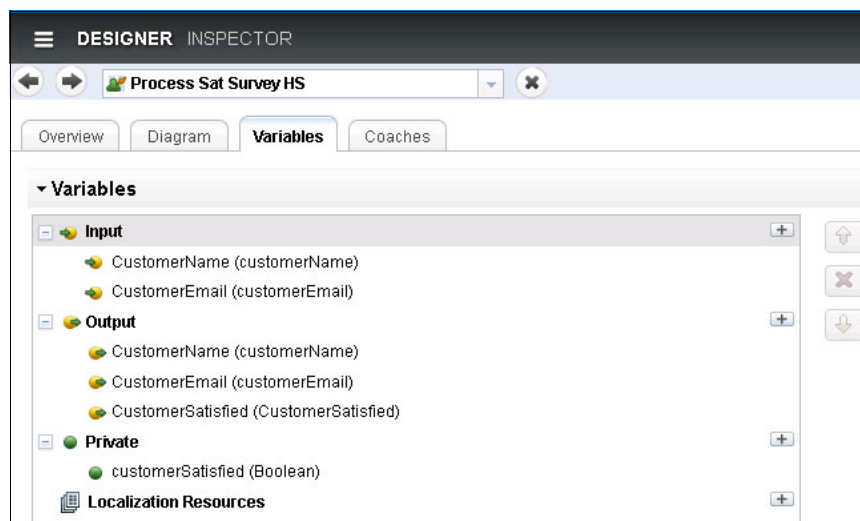


Figure 7-41 Process Sat Survey HS Variables tab

Figure 7-42 on page 260 shows the Process Sat Survey HS Coaches tab highlighting the Determine If Customer Satisfied coach form.

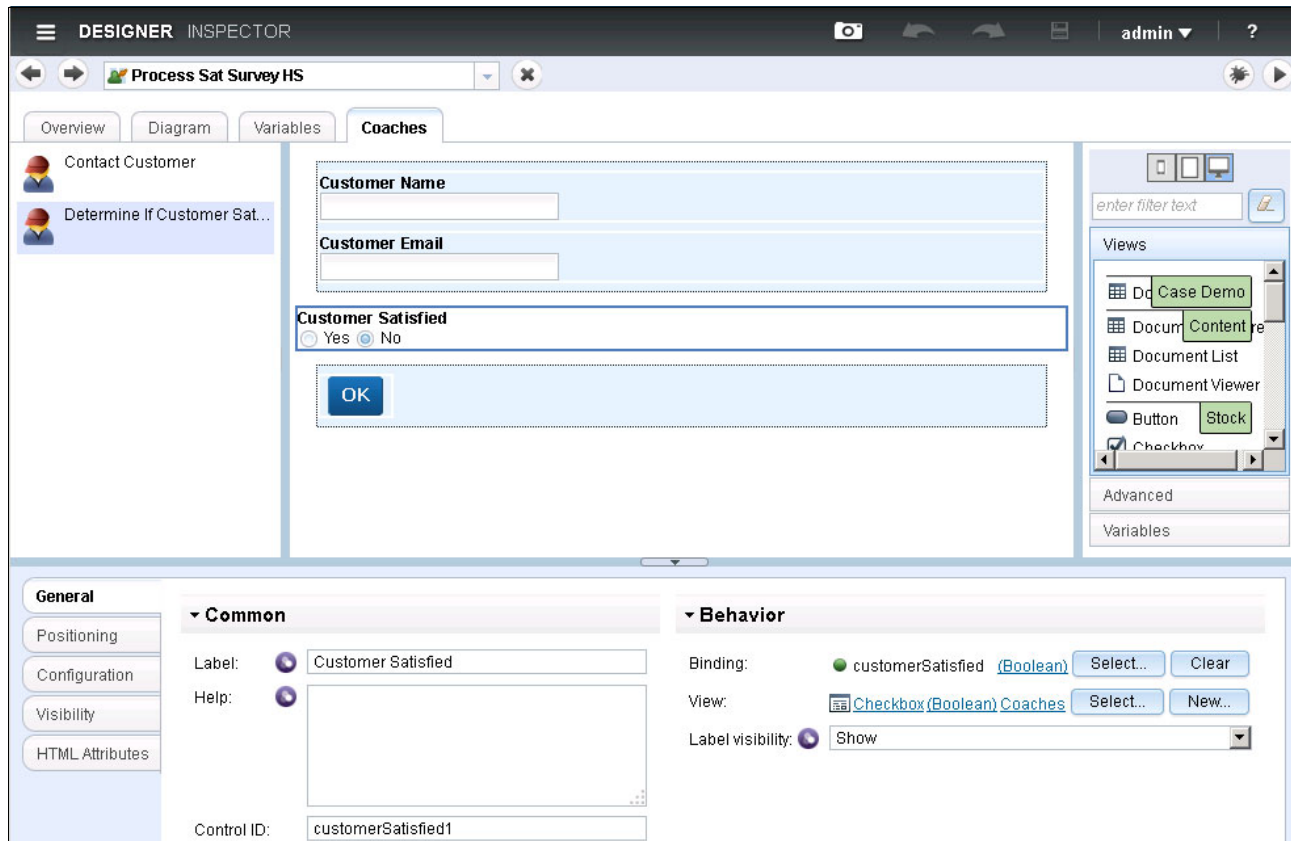


Figure 7-42 Process Sat Survey HS Coaches tab with Determine If Customer Satisfied coach form

Figure 7-43 shows the Process Sat Survey HS Coaches tab highlighting the Contact Customer coach form.

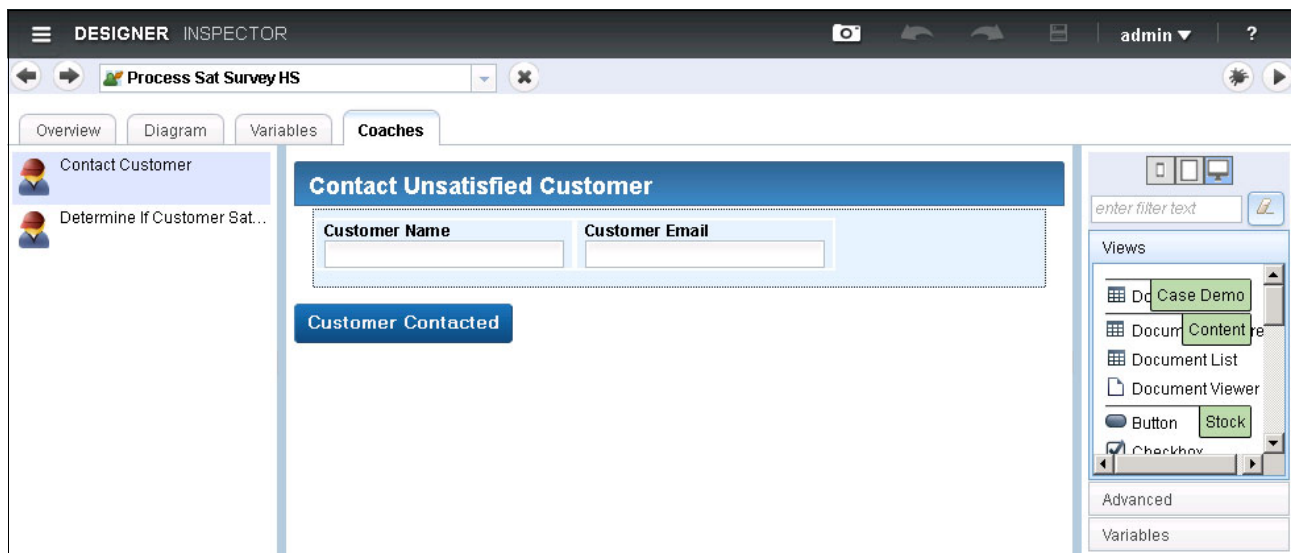


Figure 7-43 Process Sat Survey HS Coaches tab with Contact Customer coach form

7.8 Responsive coaches

Before IBM Business Process Manager V8.5.5, it was difficult to develop coaches for multiple form factors such as browsers, tablets, and smartphones. The stock coach views controls, such as text and date, were not developed to change size if the coach was rendered on different multiple-form-factor devices. Developers could develop around this limitation, but it was costly. Sometimes, it was easier to adopt the *headless* BPM pattern and create an external custom user interface to interact with the IBM BPM engine through the IBM BPM REST API.

Starting with IBM BPM V8.5.5, there are new features to help solve this problem by allowing you to design *responsive coaches* for use on multiple form factors. These new features allow you to:

- ▶ Play back coaches before deployment.
- ▶ Make Mobile part of coach design.
- ▶ Leverage responsive coaches in any external user interfaces including IBM Worklight and custom mobile applications.

To understand this new feature, it is important to know that in IBM BPM V8.5.5, a new type of human service was also introduced called a *client-side human service* that runs in the browser (the other type of human service is called *heritage human service* and runs on the IBM BPM server).

A client-side human service is the human service type that you need to use to develop responsive coaches. For more information, see 1.4.5, “Client-side human services” on page 22 and 1.4.6, “Responsive coach design” on page 23.

7.8.1 IBM Process Designer web editor

When you edit a coach form in a client-side human service, a browser-based editor opens. Figure 7-44 on page 262 shows the full screen view of the IBM Process Designer web editor.

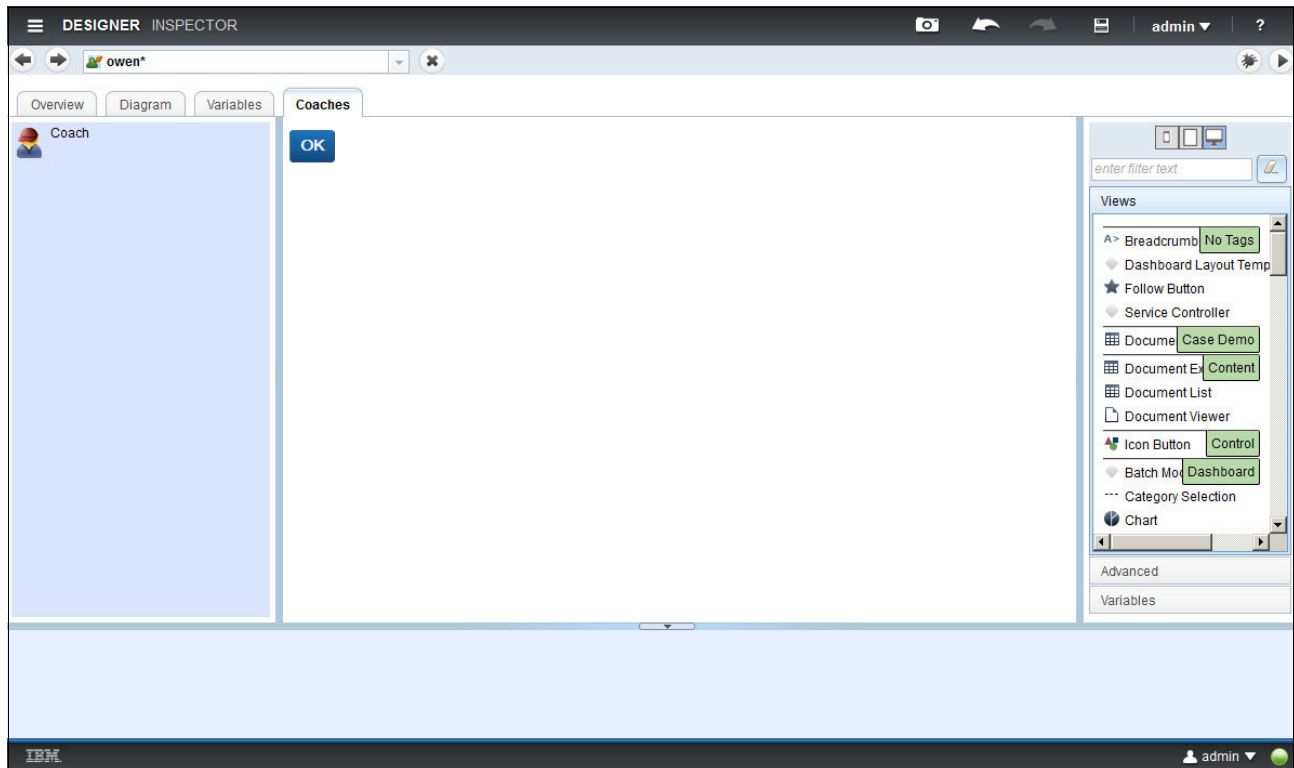


Figure 7-44 Full screen view of the IBM Process Designer web editor

It looks very similar to the IBM Process Designer coach editor (when editing coach forms in a heritage human service), but there are some key changes in this editor. The following sections described the main new capabilities in this editor to develop responsive coaches.

Preview and test coach forms in different form factors

In the upper right corner of the editor, there are three buttons to test the coach form in three different form factors. From left to right, there are icons that provide a selection of small (smartphone), medium (tablet), and large (desktop) form factors.

Figure 7-45 shows the form factor icons.



Figure 7-45 Form factor icons

Positioning

The key to enabling a coach form to adapt (be responsive) to different form factors is a new set of configuration parameters found in a new panel called *Positioning*. You can control the spacing of the coach views in your coaches by setting the positioning properties for each coach view instance.

Figure 7-46 on page 263 shows the IBM Process Designer web editor Positioning tab.

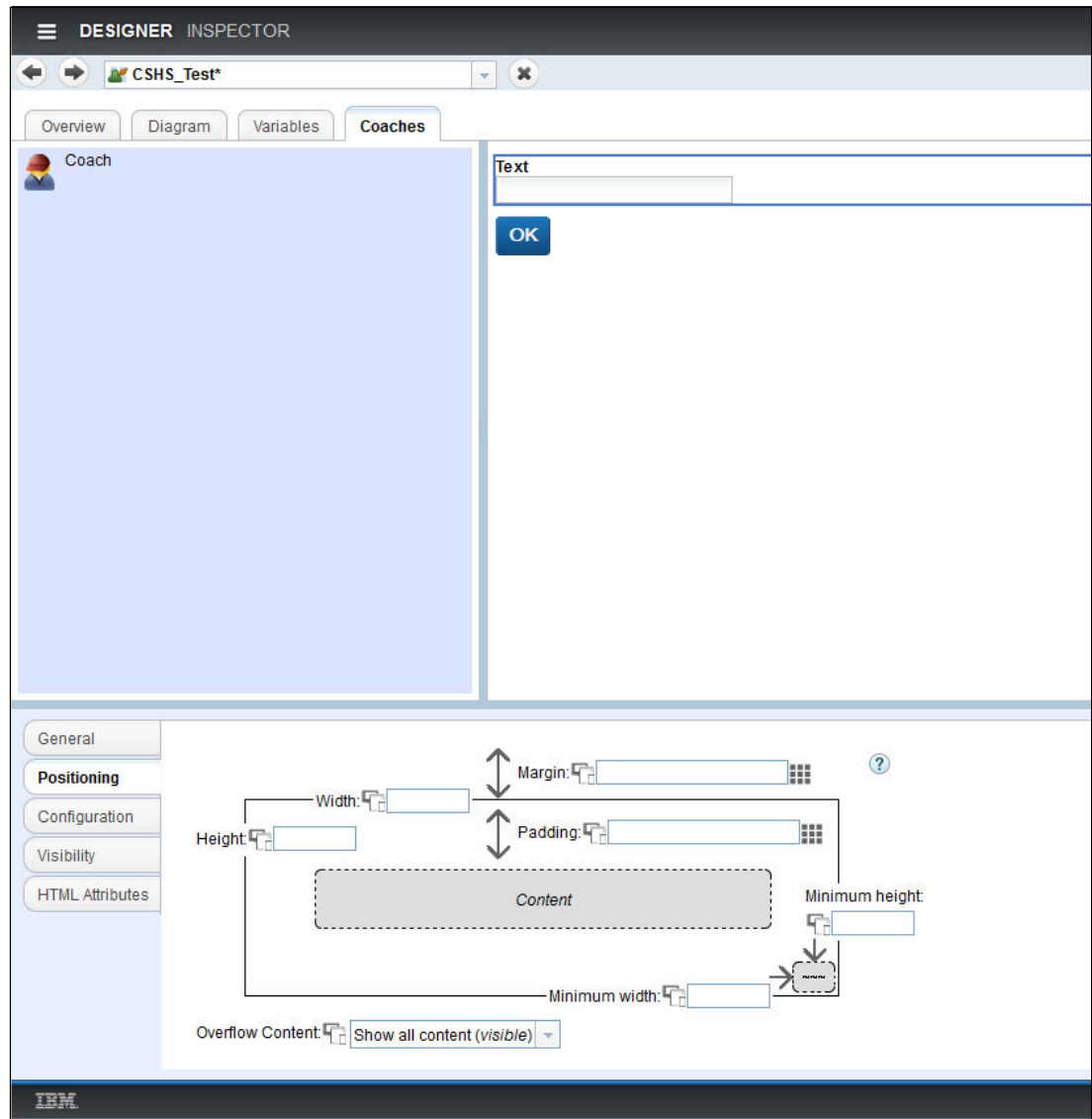


Figure 7-46 IBM Process Designer web editor Positioning tab

Each coach view now has positioning properties shown in Figure 7-46. You can specify in percentages how high or wide the coach view will be. These properties are enforced on all of the form factors. In addition, you can specify how overflow data will be handled in a coach view.

Figure 7-47 on page 264 shows a closer view of the positioning properties.

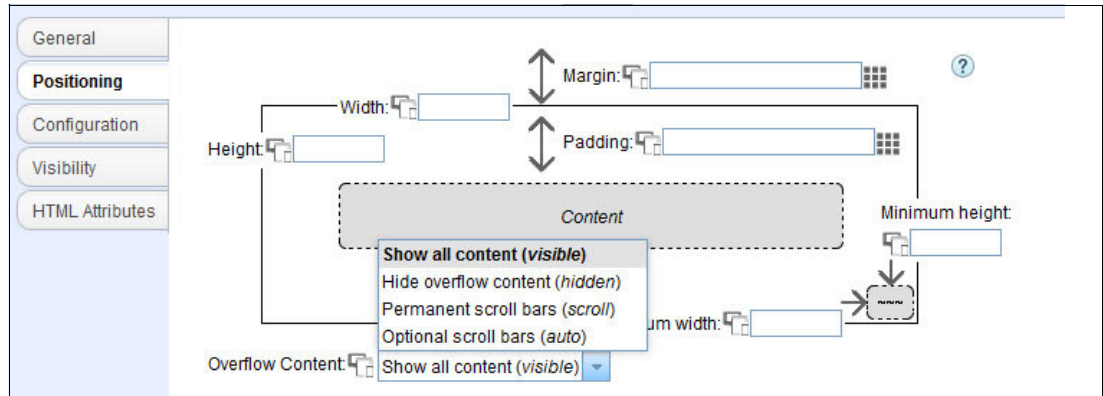


Figure 7-47 Closer look of the Positioning parameters

Figure 7-48 through Figure 7-50 on page 266 demonstrate how the positioning properties affect the rendering of the coach form.

Note: Because the coach has few controls, you will not see much difference in the various form factors in the figures but the text control shows as smaller in the medium and small form factors.

Figure 7-48 shows the coach rendered in the large form factor.

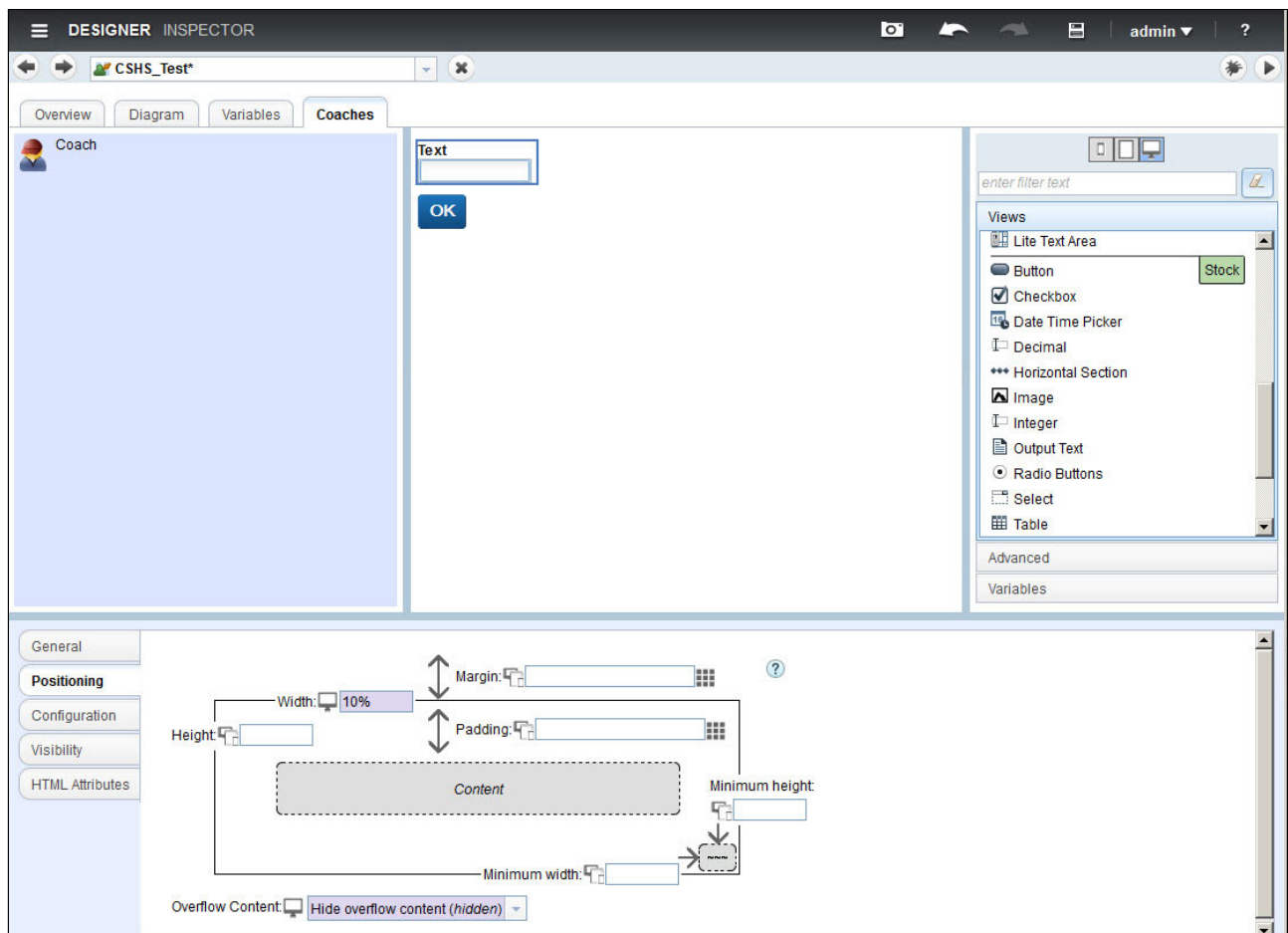


Figure 7-48 Large form factor

Figure 7-49 shows the coach rendered in the medium form factor.

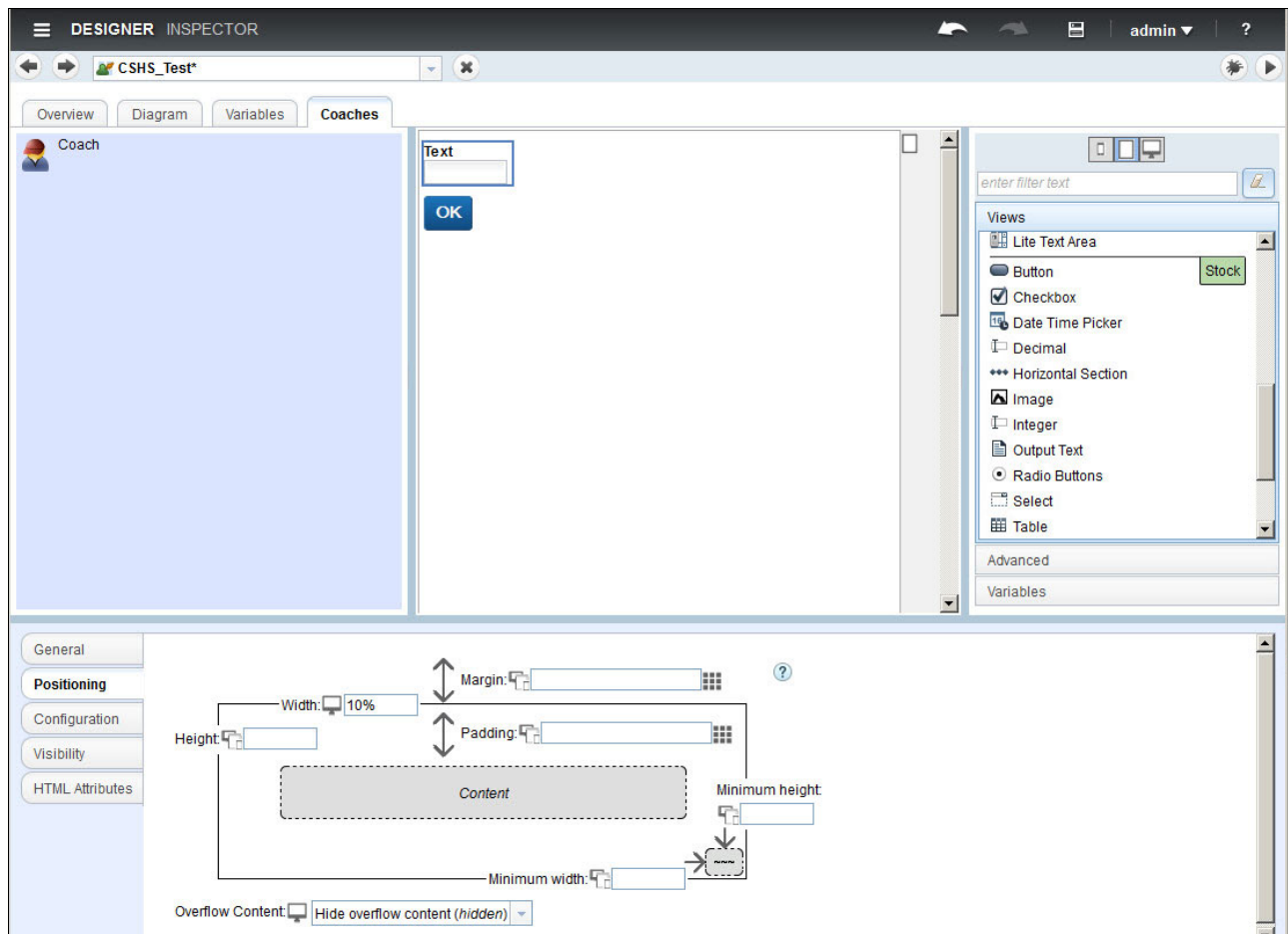


Figure 7-49 Medium form factor

Figure 7-50 on page 266 shows the coach rendered in the small form factor.

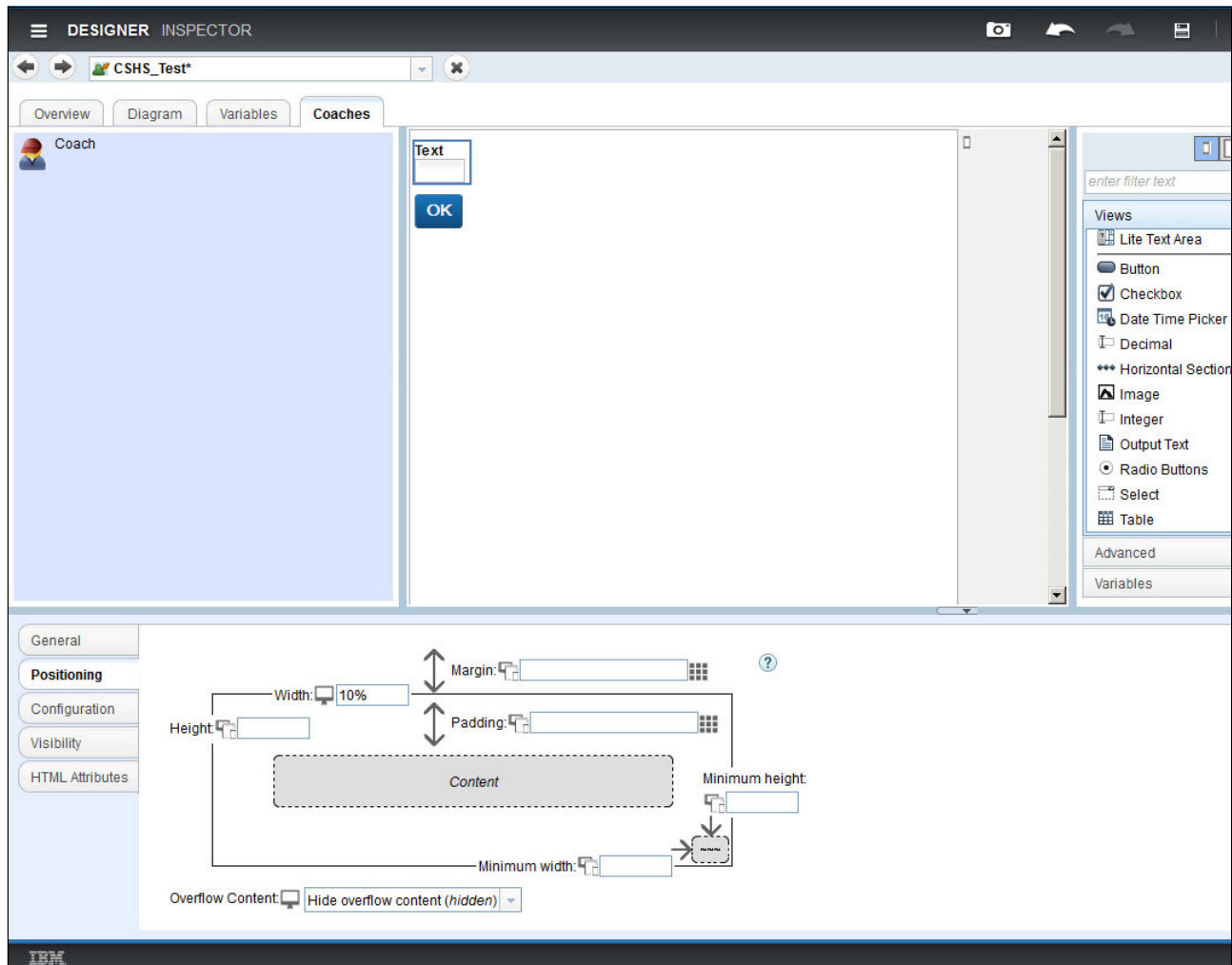


Figure 7-50 Small form factor

As stated earlier, all of the stock coach views (such as text and date) support positioning configuration settings.

Responsive coach configuration parameters

Certain controls allow you to choose different configuration parameters. The Configuration tab is located under the Positioning tab for each form factor.

For example, you can show the Date Time Picker control fully expanded (inline mode) for the large form factor.

Figure 7-51 on page 267 shows the Date Time Picker coach view inline.

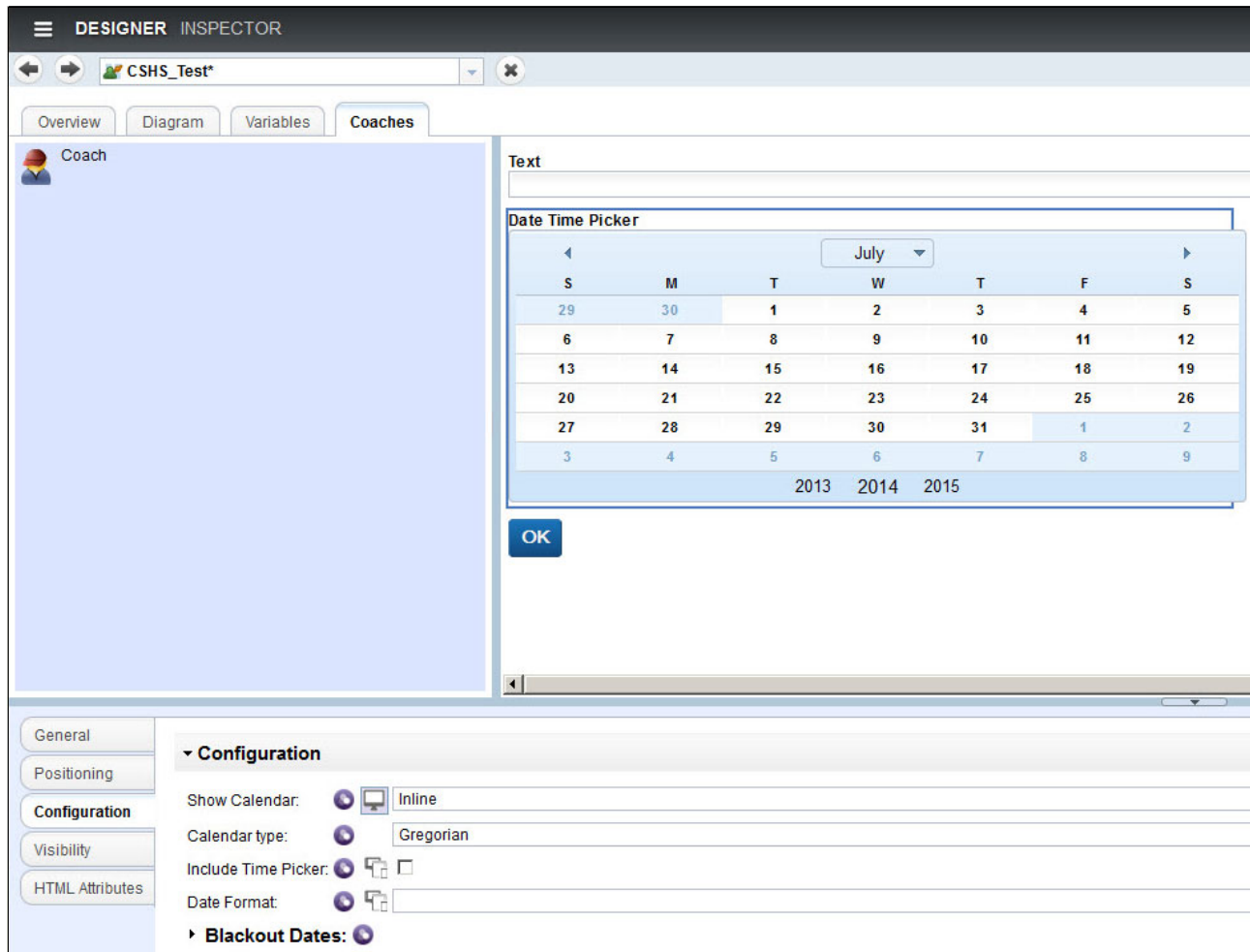


Figure 7-51 Date Time Picker coach view inline

And, you can show the Date Time Picker control fully collapsed (on click mode) for the medium form factor when the coach form renders. To show the Date Time Picker control fully collapsed, change the Show Calendar configuration parameter from `Inline` to `On click`.

Figure 7-52 on page 268 shows the Date Time Picker coach view collapsed.

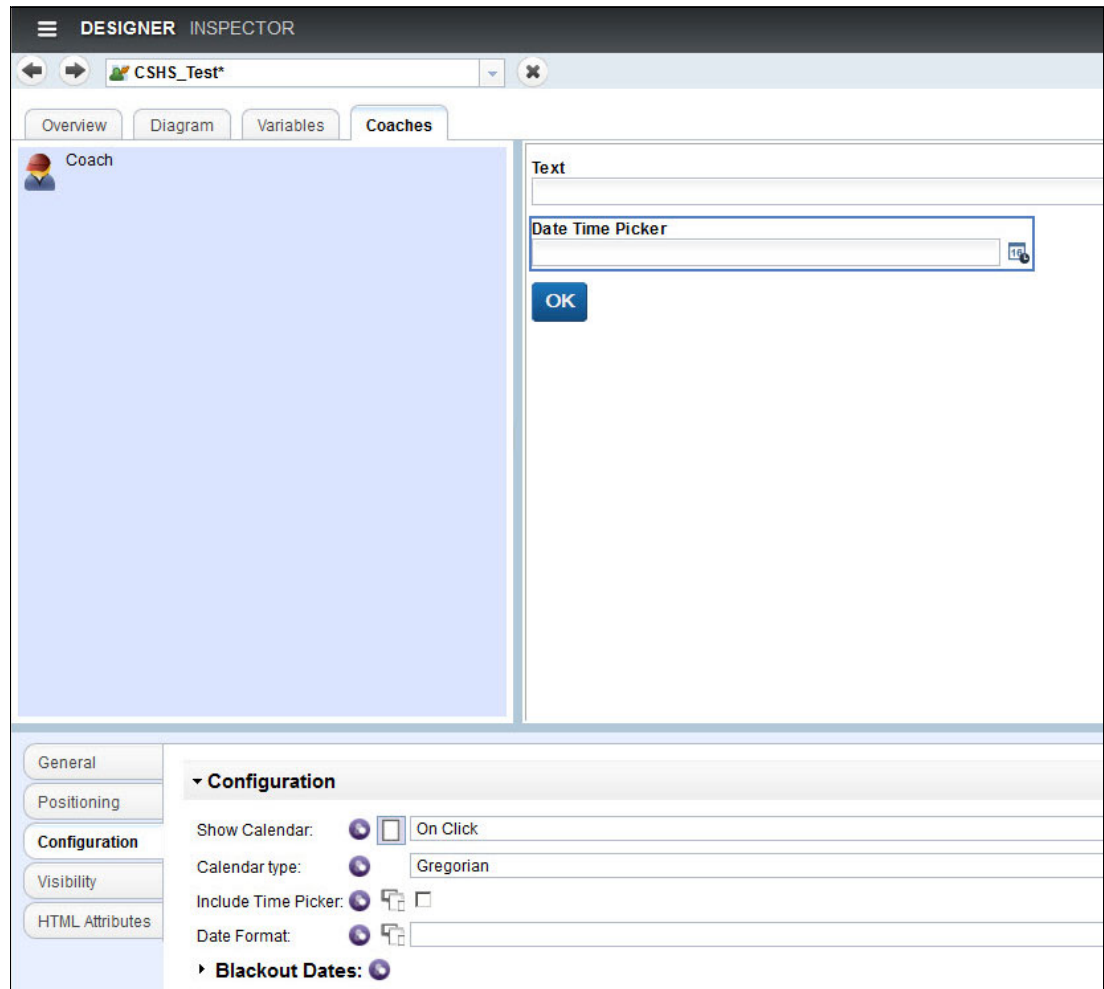


Figure 7-52 Date Time Picker coach view on click

7.8.2 Modifying a coach form to be responsive

This section describes how to reconfigure coach views to test how responsive coaches are rendered on various form factors. This example is based on the coach form in Manage New Order HS. Taking a closer look at the Horizontal Section coach view with the label of Documents, Figure 7-53 on page 269 shows the Documents Horizontal Section coach view selected. Notice that the Configuration tab is opened. The following properties are selected:

- ▶ Collapsible: The entire coach view can be collapsed.
- ▶ Automatic Wrap: Wraps all of the coach views contained in the Horizontal Section coach view.

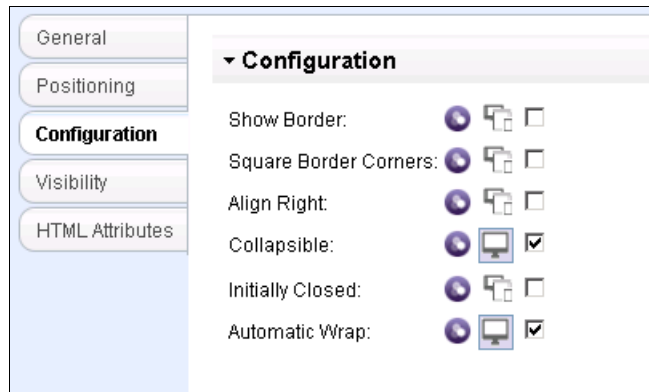


Figure 7-53 Documents horizontal section coach view

Figure 7-54 shows the Document Explorer coach view selected. Notice that the Configuration tab is opened. The Collapsible property is selected so that the entire coach view can be collapsed.

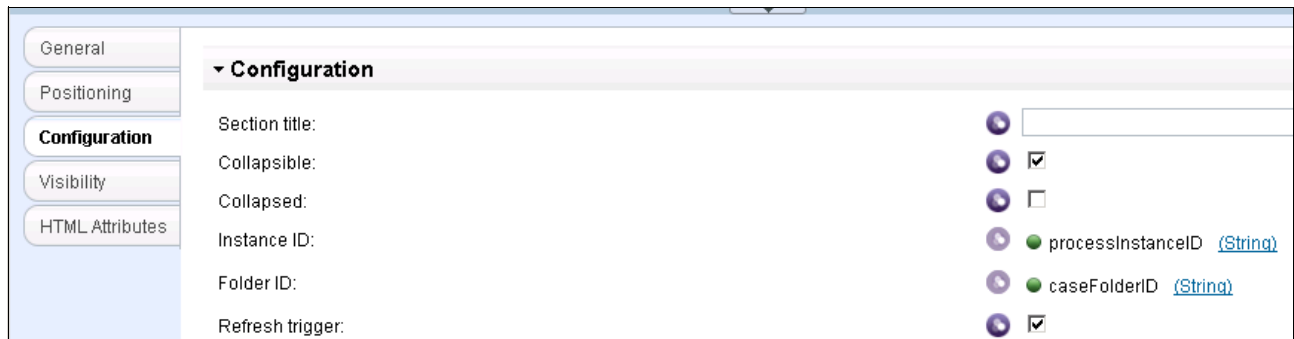


Figure 7-54 Document Explorer coach view

To see the responsive coach working, start a case instance. To start a case instance, upload a NewOrderDocument document type.

The samples provided with this IBM Redbooks publication include a dashboard coach called *Document Dashboard* (which is packaged in the toolkit DocumentUpload). It allows you to upload a document to the embedded Enterprise Content Management. For information about the samples provided with this book, see Appendix A, “Samples included with this book” on page 307.

Figure 7-55 on page 270 shows the document dashboard.

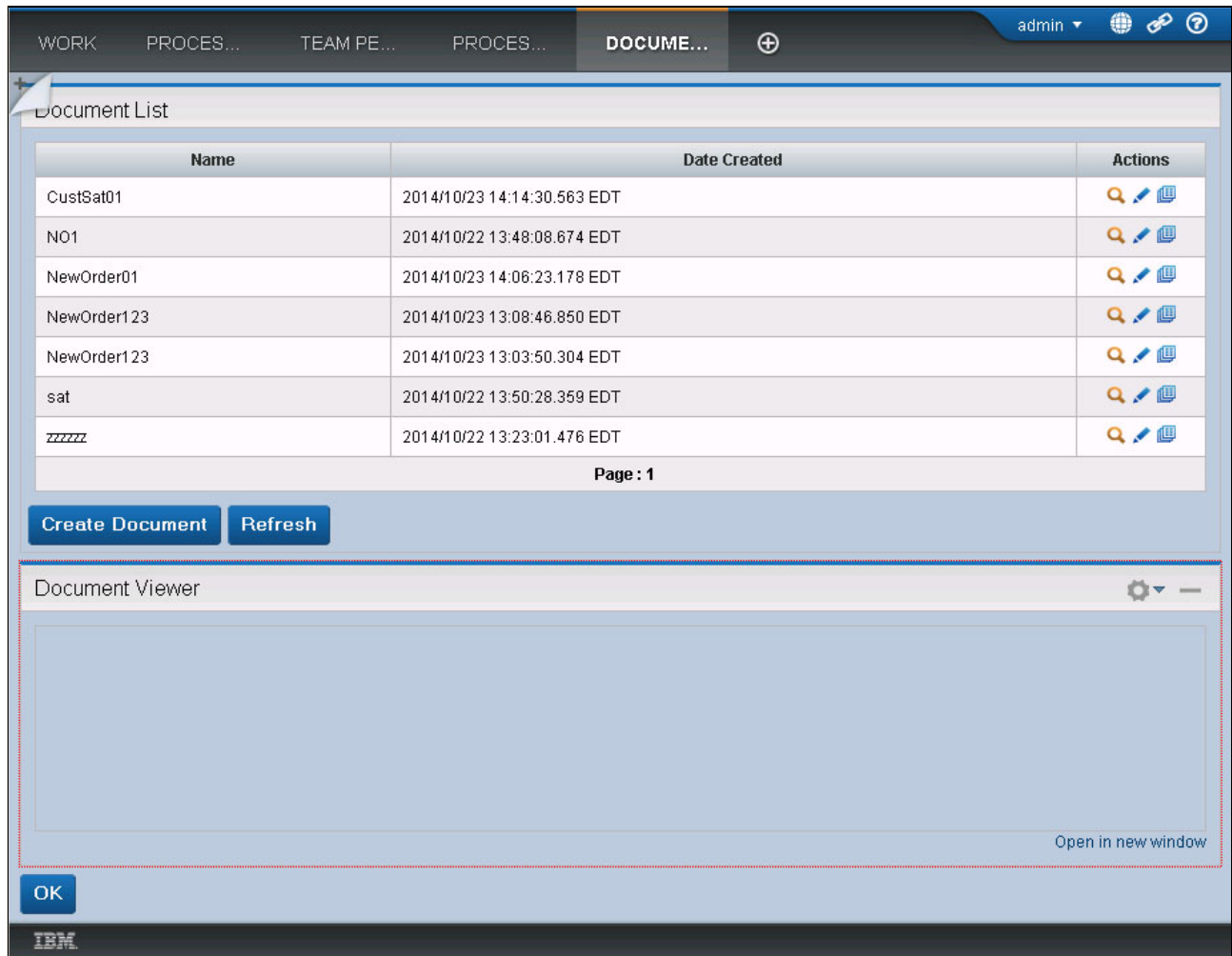


Figure 7-55 Document Dashboard

Upload a document to start a case instance:

1. Click **Create Document**.
2. Select the document to upload.

You must select the correct document type or the case instance will not be created. Therefore, choose a **New Order Document** document type.

Figure 7-56 on page 271 shows the Create Document window completed to upload a New Order Document.

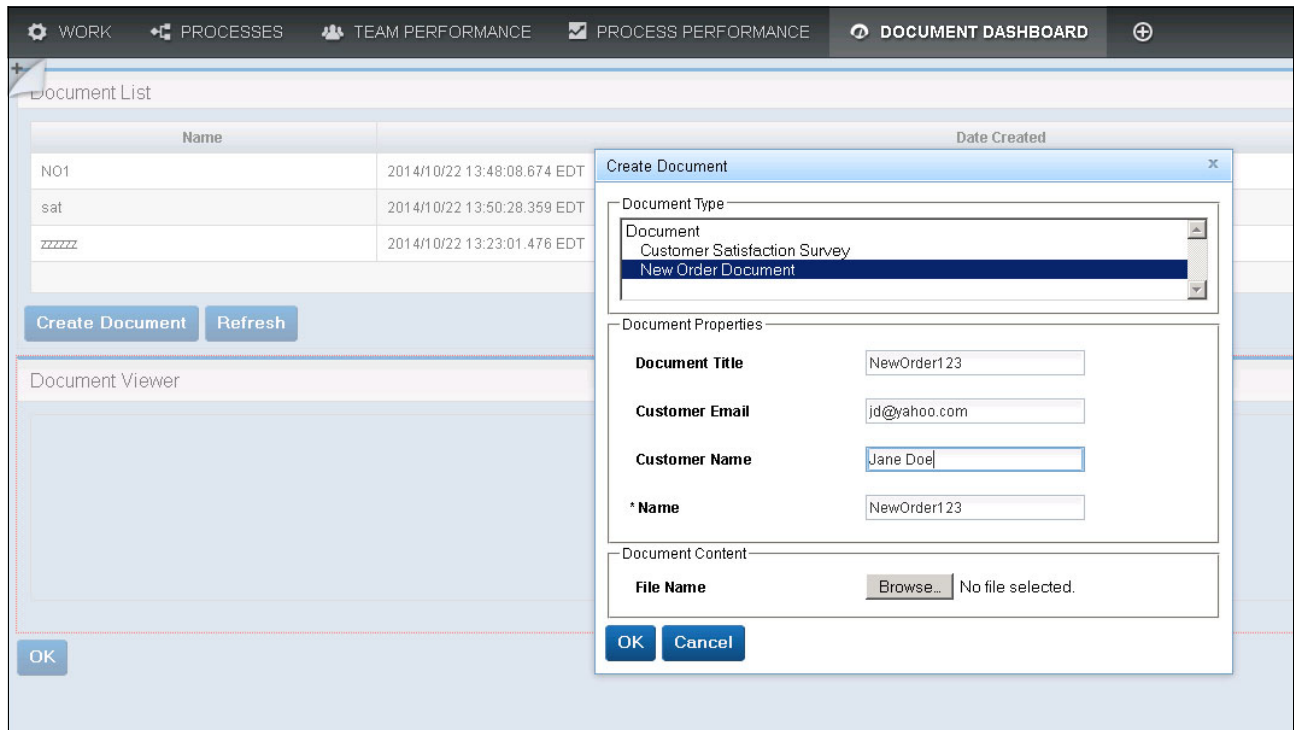


Figure 7-56 Document Dashboard: Create Document

Figure 7-57 shows that the document is listed after being uploaded.

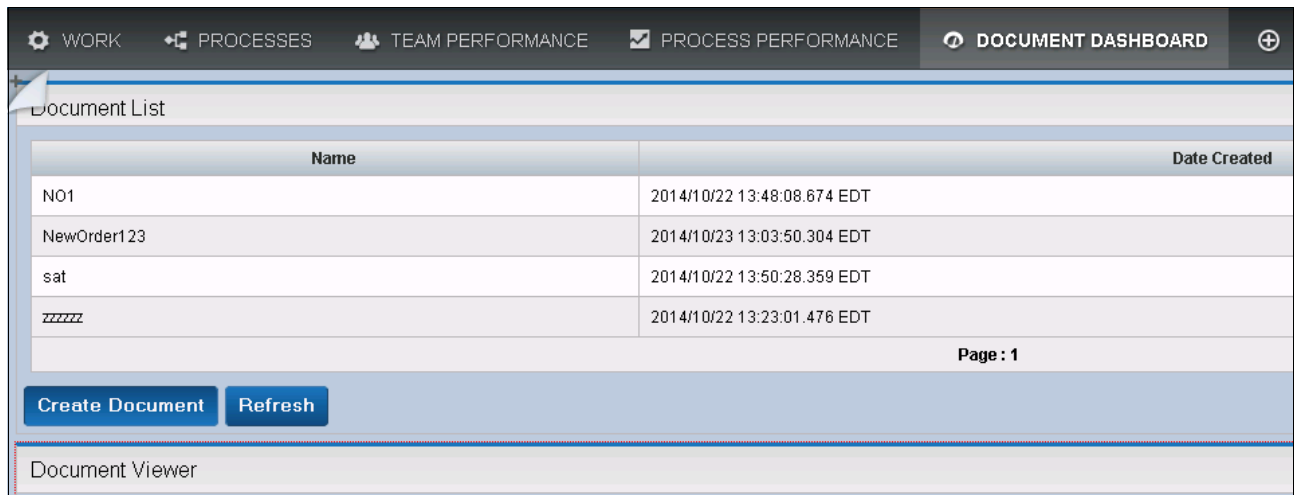


Figure 7-57 Document Dashboard: Document List

Figure 7-58 on page 272 shows that the case instance was indeed started.

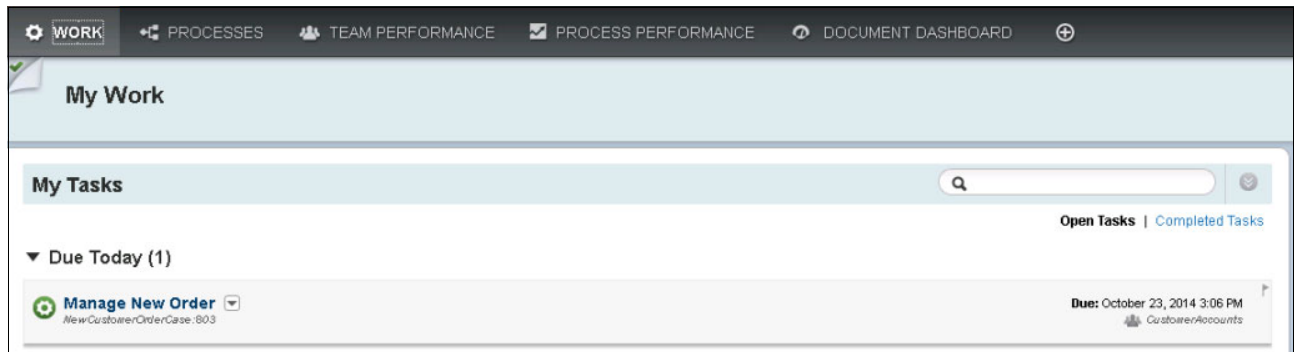


Figure 7-58 Manage New Order in Process Portal Task list

Figure 7-59 shows the Manage New Order coach form opened after it was claimed.

Figure 7-59 Manage New Order coach form

To test the IBM BPM responsive coach design changes, use the Responsive Design View option in the Firefox browser. The Responsive Design View makes it easy to see how a website or web app will look on different screen sizes. It helps to simulate various UI form factors.

To run the test, perform the following steps:

1. From the Firefox Tools menu, click **Web Developer** → **Responsive Design View**

2. Select the wanted screen size from the list.
3. Test the coach by switching from one screen size to the others, verifying the behavior of the adaptive UI components in the coach:
 - When emulating a desktop, use the large screen size: 1280x600.
 - When emulating a tablet, use the medium screen size: 800x1280.
 - When emulating a smartphone, use the small screen size: 360x640.

Note the following:

- ▶ As you make the screen size narrower, notice the various components of the horizontal sections rearranging themselves vertically.
- ▶ If you continue making the screen narrower, you will eventually see the specific sections being collapsed so that only the labels are visible.
- ▶ The sections can be expanded by clicking their labels. This behavior happens because of the mobile characteristics that were set on the coach view used in the design.

Figure 7-60 shows the Manage New Order coach form in the large format.

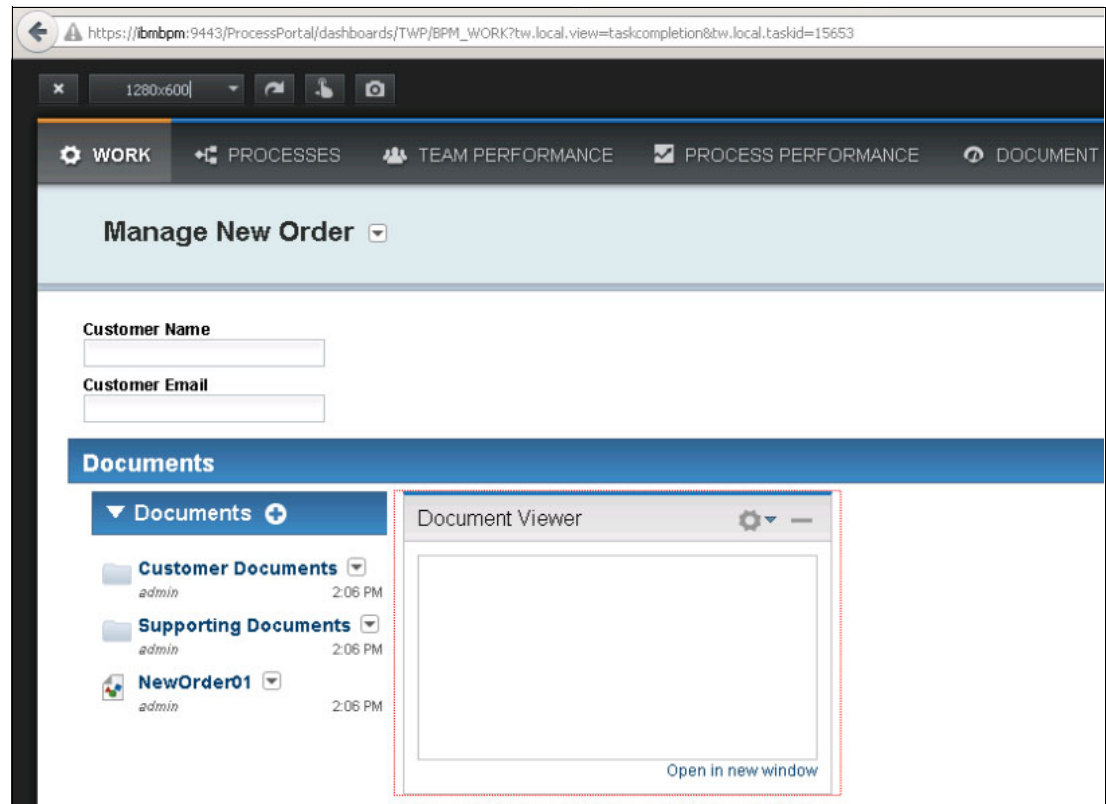


Figure 7-60 Manage New Order coach form in the large format

Figure 7-61 on page 274 shows the Manage New Order coach form in the medium format.

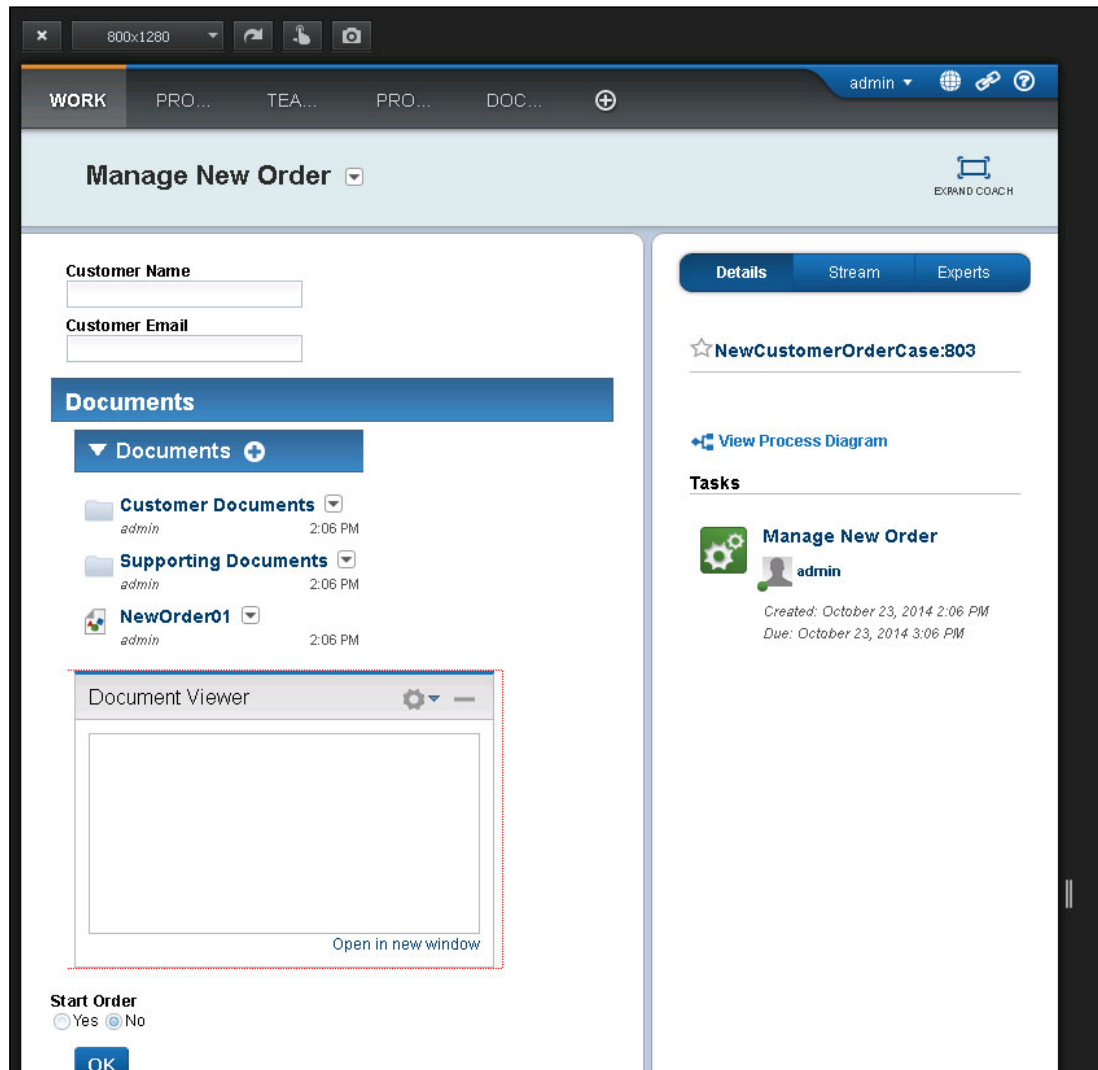


Figure 7-61 Manage New Order coach form in the medium format

Figure 7-62 on page 275 shows the Manage New Order coach form in the medium format with the Document Explorer coach view collapsed.

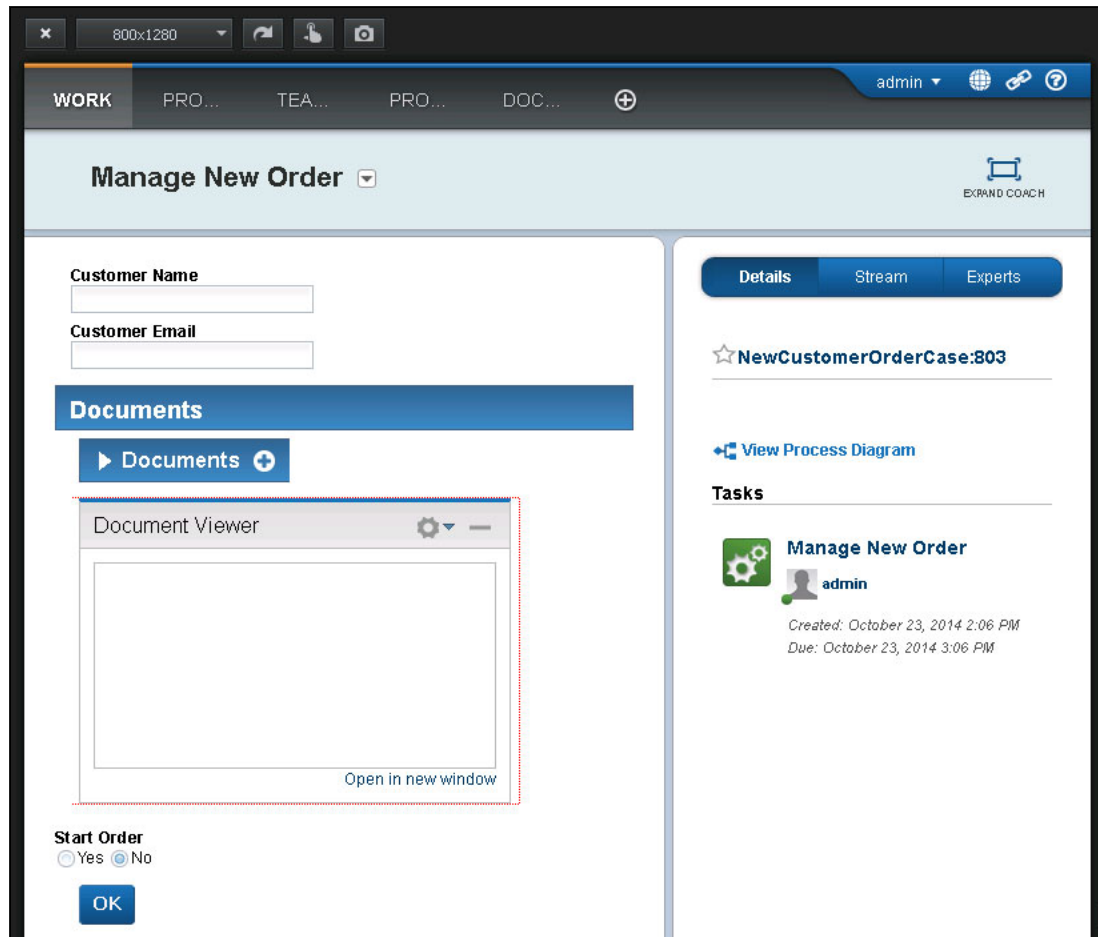


Figure 7-62 *Manage New Order coach form in the medium format with Document Explorer collapsed*

Figure 7-63 on page 276 shows the Manage New Order coach form in the medium format with document horizontal section collapsed.

Figure 7-63 Manage New Order coach form: Medium format with document horizontal section collapsed

Figure 7-64 shows the Manage New Order coach form in the small format. You cannot see the coach form because Process Portal takes up all of the screen space.

Figure 7-64 Manage New Order coach form in the small format but with Process Portal obscuring it

Figure 7-65 shows the Manage New Order coach form in the small format after clicking the full screen icon on the upper right Process Portal menu.

The screenshot shows a web application window with a title bar indicating a size of 320x480. The main content area is titled "Manage New Order" and includes a full-screen icon in the top right corner. The form contains the following elements:

- Customer Name**: A text input field.
- Customer Email**: A text input field.
- Documents**: A blue header for a section.
- Start Order**: A section with two radio buttons, "Yes" and "No". The "No" option is selected.
- OK**: A blue button at the bottom of the "Start Order" section.

Figure 7-65 *Manage New Order coach form in the small format in full screen mode*

Next, create a CustomerSatisfactionSurvey document type on the case. To add a document to the Manage New Order coach form, click **Add Document** as shown in Figure 7-66 on page 278.

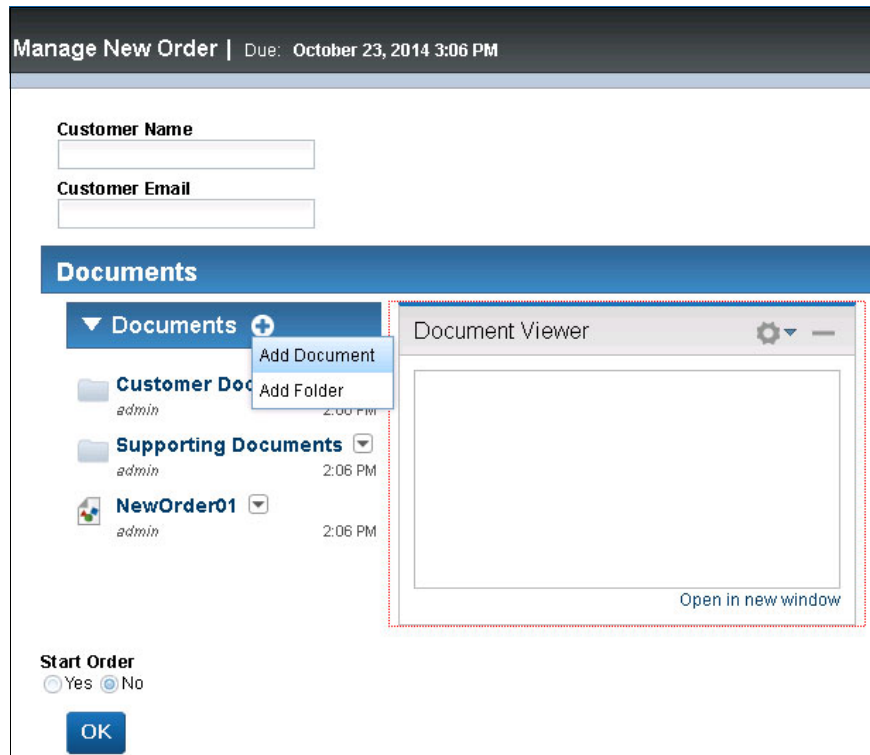


Figure 7-66 Add Document on the Manage New Order coach form

Figure 7-67 shows the Add Document panel.

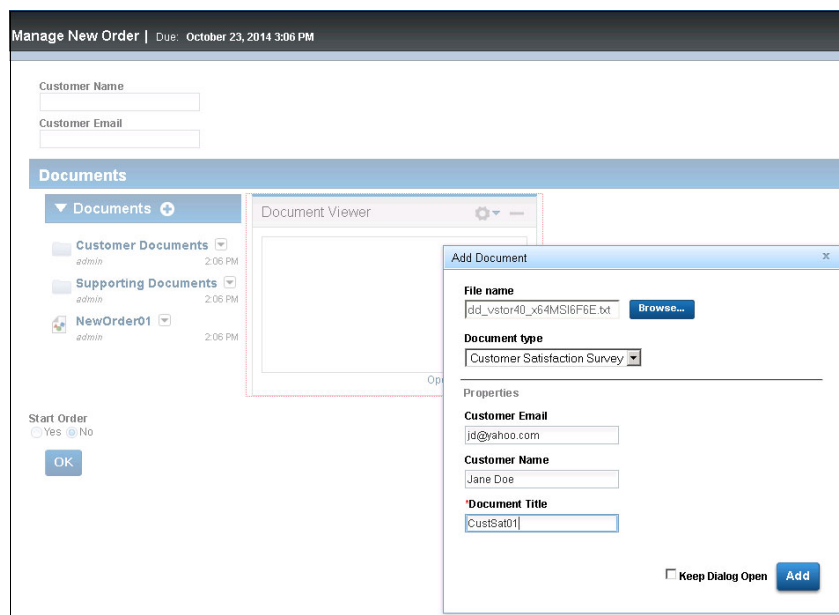


Figure 7-67 Add Document panel

Clicking **Add** uploads the document and another activity is triggered.

Figure 7-68 on page 279 shows the resulting Process Survey task in the Process Portal task list.

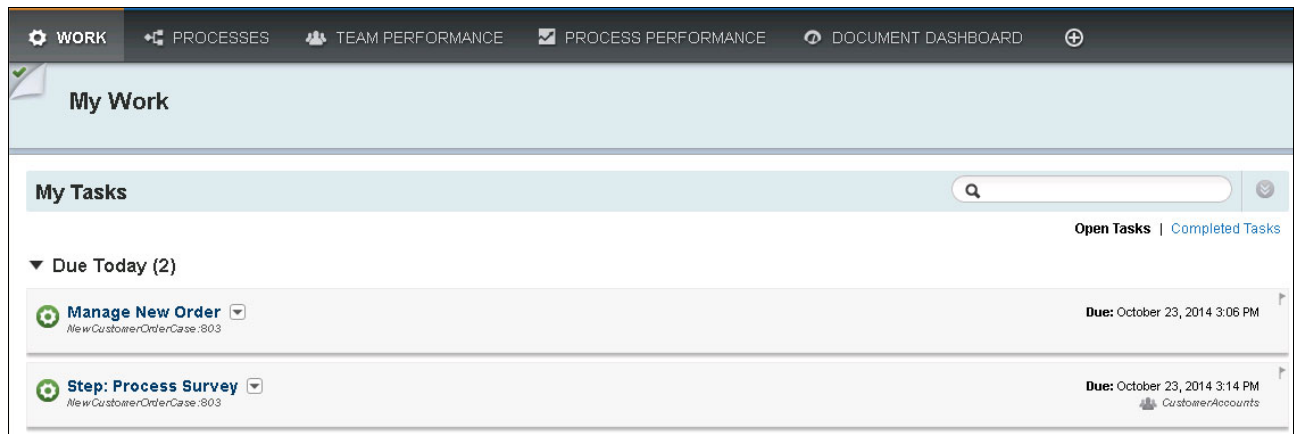


Figure 7-68 Process Survey Task in the Process Portal task list

The Manage New Order coach view shows the new document uploaded (see Figure 7-69).

The user can select the **Yes** radio button under Start Order and complete the Manage New Order activity. A new instance of the New Order Installation linked process is started then.

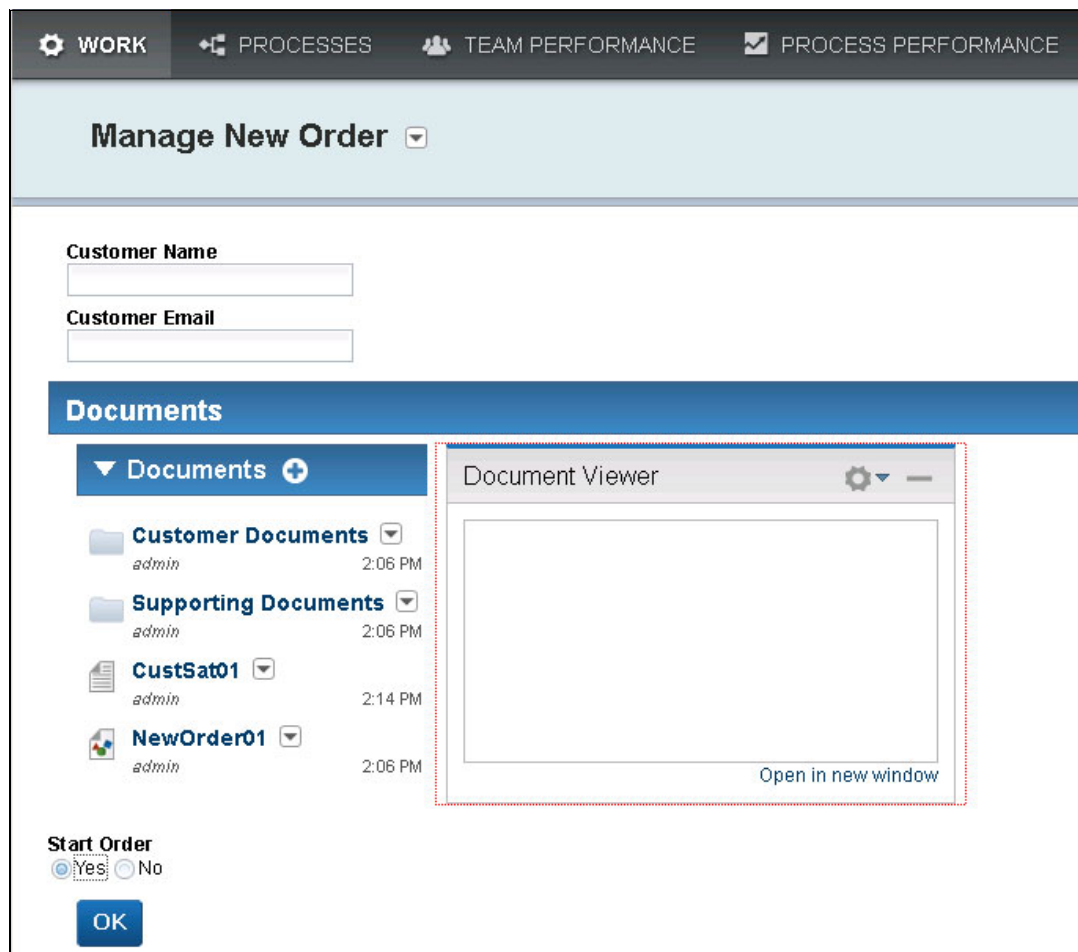


Figure 7-69 Process Survey Document added to the case

Figure 7-70 shows the task list for a different user, Allen. As a result of the new document being uploaded, the New Order Installation process was automatically started. And Allen is able to claim the first task in the New Order Installation process.

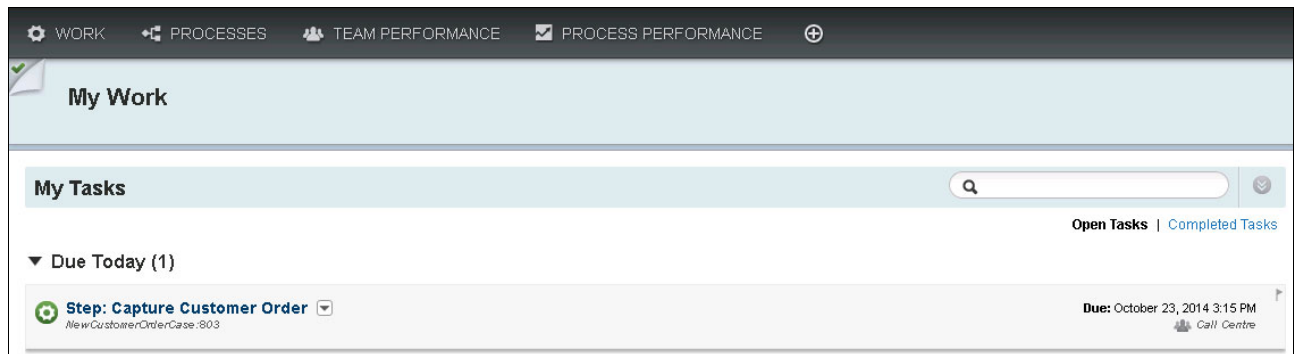


Figure 7-70 Task list for Allen

7.8.3 Accessing responsive coaches from mobile devices

Section 1.3.6, “Mobile technology adoption patterns” on page 17 discusses patterns that organizations can adopt for accessing IBM BPM processes from mobile apps.

This section provides examples of how to access responsive coaches from mobile devices.

Using Process Portal in a browser

You can access the coaches using the standard Process Portal running in the browser of a mobile device. The downside of this approach is that, on smartphones, the Process Portal itself takes up a large part of the screen. Aside from being able to manually tap the Expand View button, the expansion can be triggered automatically through code. To expand the coach to full mode:

1. Create a coach view with the following function on the View Event handler:

```

window.parent.document.querySelector('[dojoattachpoint="fullSizeControl"]').click();

```

Figure 7-71 on page 281 shows the coach view editor.

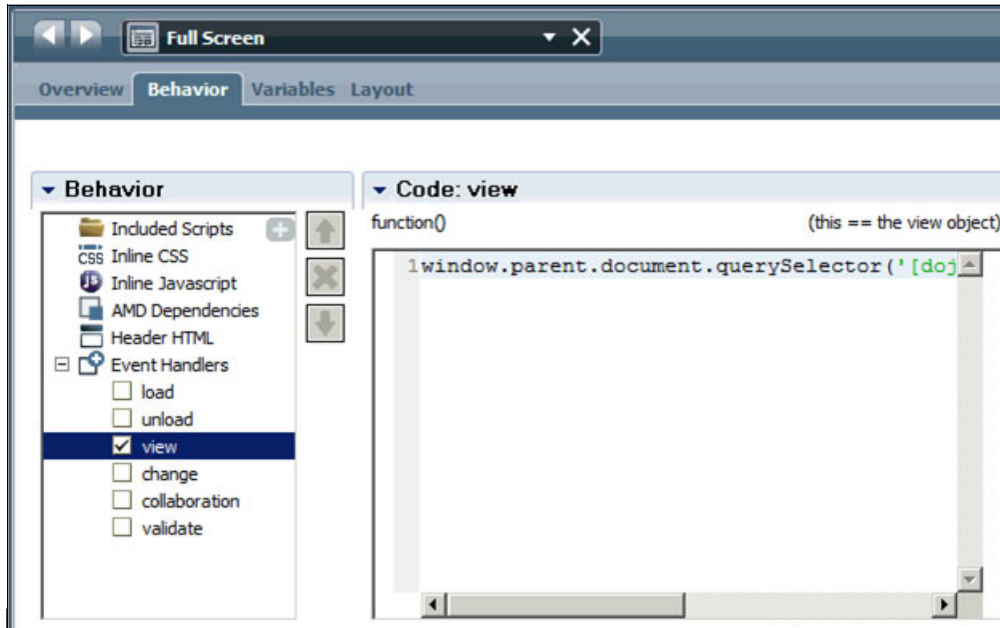


Figure 7-71 Coach view editor

- Drop the coach view created in step 1 on page 280 at the top of any coach that you want to force into full screen when it is opened in the Process Portal. Figure 7-72 shows the Full Screen coach view on a coach form.

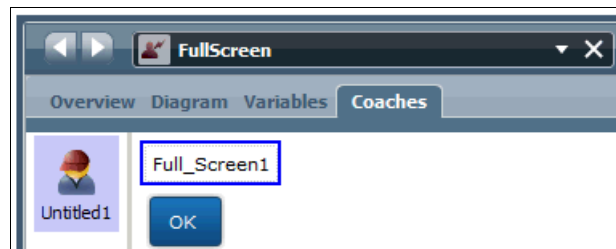


Figure 7-72 Full Screen coach view on a coach form

IBM BPM mobile apps samples

Several mobile app samples are provided in the IBM BPM Community — SAMPLE EXCHANGE at <http://wiki.bpmwiki.com/display/samples/Mobile+apps>

Under this link, you can find the following samples:

- ▶ CV - Mobile Signature - TK (SAMPLE EXCHANGE): The Mobile Signature control is a flexible easy to use coach view that captures a signature drawing using either a mouse or touch device (supports most web browsers, iOS, Android, and BlackBerry).
- ▶ Mobile BPM (Android sample): The purpose of this sample is to show how the IBM BPM REST API can be used by an external application.
- ▶ Mobile BPM (iOS) sample: This sample contains code from the IBM BPM app, showing how to use the IBM Business Process Manager and Blueworks Live REST APIs in a mobile application. Review this sample to gain knowledge about how to use the IBM Business Process Manager and Blueworks Live REST APIs for an iPhone or iPad environment.
- ▶ Mobile BPM (Web Sample).

- ▶ Worklight sample for BPM 8.0: This sample application allows iOS and Android devices to access an IBM BPM V8.0 server, using IBM Worklight.
- ▶ Worklight Sample for IBM Worklight V6.0 and IBM BPM 8.5 with Push Notifications.
- ▶ Worklight Sample for IBM Worklight 6.1 and IBM BPM 8.5.0.1 with Push Notifications.

IBM BPM mobile app for iOS

This app provides an out-of-the-box artifact for mobile interaction with IBM BPM processes from Apple iPhone or iPad devices. Refer to section 1.4.1, “IBM BPM mobile app for iOS” on page 19 for more information.



Deeper insight through IBM Business Process Manager and IBM Worklight analytics

This chapter provides an overview of the basic analytics capabilities available for IBM Business Process Manager (IBM BPM) and IBM Worklight.

This chapter discusses how IBM Business Monitor can be combined with IBM BPM to accelerate processes and facilitate business innovation.

This chapter provides an overview of the data displayed by the Process Admin Console to enable you to monitor the performance of the Process Servers in your environment. It helps to identify performance bottlenecks in Process Server and further analyze performance issues.

This chapter includes an overview of IBM Worklight operational analytics features and views.

8.1 IBM BPM business analytics

Monitoring is a key capability that must be available when organizations design and implement smarter processes.

IBM Business Monitor is a business activity monitoring environment that displays key performance indicators (KPIs) and metrics through dashboards. You can combine IBM Business Monitor with other Business Process Management (BPM) software to accelerate processes and facilitate business innovation.

IBM BPM governs cross-functional, core business process of an organization to achieve business objectives by directing resources from across the organization into efficient processes that create customer value. Use IBM Business Monitor in your business environment to:

- ▶ Monitor business transactions in real-time, allowing business stakeholders to visualize and gain insight into metrics, processes, business situations, and events to optimize business outcomes and take action while it matters.
- ▶ Gain end-to-end visibility and insight into business transactions that span applications and systems, enabling business stakeholders to take smarter business actions based on the full picture.
- ▶ Easily define user customizable dashboards, providing business and IT users with a targeted view of the relevant information needed to most effectively manage their business.
- ▶ Leverage advanced analytics to predict future values of KPIs based on historic and cyclic trends, empowering business stakeholders to proactively manage their business to take advantage of business opportunities or to avoid potential business threats.
- ▶ Automatically trigger alerts and actions when actual or predicted values detect a current or potential business situation, enabling business stakeholders to quickly react to changing business conditions.
- ▶ Define the actions to take when specified situations occur.
- ▶ Capture business-related data from business applications, based on the monitor model that you define and install.
- ▶ Extract the measurement variables from the data.
- ▶ Transform the variables into metric and KPI values.
- ▶ Display the measurement values on your dashboard. For example, you can display instance diagrams in context in any annotated Scalable Vector Graphics (SVG) diagrams. You can define elements to set colors, show and hide, associate metric and KPI values, and more.
- ▶ Provide business intelligence insight through dimensional analysis and reporting.
- ▶ Identify and notify you of operation failures for inspection and analysis.

Figure 8-1 on page 285 provides an overview of the key features in IBM Business Monitor, which provide visibility and insight to optimize business outcomes.



Figure 8-1 IBM Business Monitor: Providing visibility and insight to optimize business outcomes

Monitor models describe how events should be processed and how information should be collected for use with dashboards. The IBM Business Monitor development toolkit provides technical users with an environment for creating and testing monitor models.

The IBM Business Monitor server receives and processes events from business applications, and it can subscribe to business events from various sources, including IBM Business Process Manager and other applications in the business environment.

With IBM Business Monitor, you can monitor your business processes, regardless of what product hosts them. The monitor model can be designed to provide end-to-end monitoring of your business transactions, even if those transactions are made up of discrete processes from different products. You can use IBM Business Monitor to correlate events from IBM Business Process Manager and a broad range of IBM software for end-to-end monitoring, including the following products:

- ▶ IBM Operational Decision Management
- ▶ IBM Integration Bus
- ▶ IBM WebSphere Enterprise Service Bus
- ▶ IBM Operational Decision Manager
- ▶ IBM WebSphere DataPower X150
- ▶ IBM WebSphere Sensor Events
- ▶ IBM WebSphere MQ File Transfer Edition
- ▶ IBM Sterling Control Center
- ▶ IBM CICS®

- ▶ IBM IMS™
- ▶ IBM FileNet® P8

You can also monitor third-party environments, such as SAP, by using the IBM adapters.

The events that the server receives reflect your business activity. Information processed from the events is stored in the Business Monitor database. You can configure the IBM Business Monitor server to detect special business situations and manage the resulting actions.

8.1.1 Business Space widgets for business monitoring

Using the Web 2.0 IBM Business Monitor Business Space interface, users can create a personal business space that combines business data from multiple sources. Each business space consists of custom pages that display content in one or more views on each page. The views on a page are enabled by widgets that are tailored for different types of dynamic and static content, such as business process information, human task activities, process diagrams, KPIs, dimensional views, and documents (such as spreadsheets and presentations). The content sources can be local or remote. Each user can create multiple business spaces.

Using Business Space widgets business users can:

- ▶ Author reports
 - Based on Cognos 10 BI
 - Measures and dimensions
 - Rolling time periods
 - Choice of chart types
 - Rapidly Adaptive Visualization Engine (RAVE) reports. RAVE reports allow dynamic filtering in real time (for example, business users can move a slider and have the report graphically adjust)
- ▶ Author KPIs
 - Based on business data
 - Also supports prediction
- ▶ Author alerts
 - Reactive or Predictive
 - View full end-to-end reports
 - Correlates data across multiple systems (for example, IBM BPM with SAP with .NET)

Using a supported web browser, you can display one or more uniquely configured widgets, such as alerts, KPIs, or reports on a page of a dashboard space. You can group pages into one or more spaces. You can also have dashboards automatically generated when you deploy a monitor model, instead of configuring the dashboards manually.

Figure 8-2 on page 287 shows an example of an IBM Business Monitor business user dashboard.



Figure 8-2 IBM Business Monitor business user dashboard

Figure 8-3 on page 288 provides an overview of the Business Space widget palette for business monitoring. By using the widget palette, users can customize their Business Space.



Figure 8-3 Business Space widget palette for business monitoring

Figure 8-4 on page 289 takes a closer look to key performance indicators. KPIs compare real-time end-to-end data against user-defined targets.

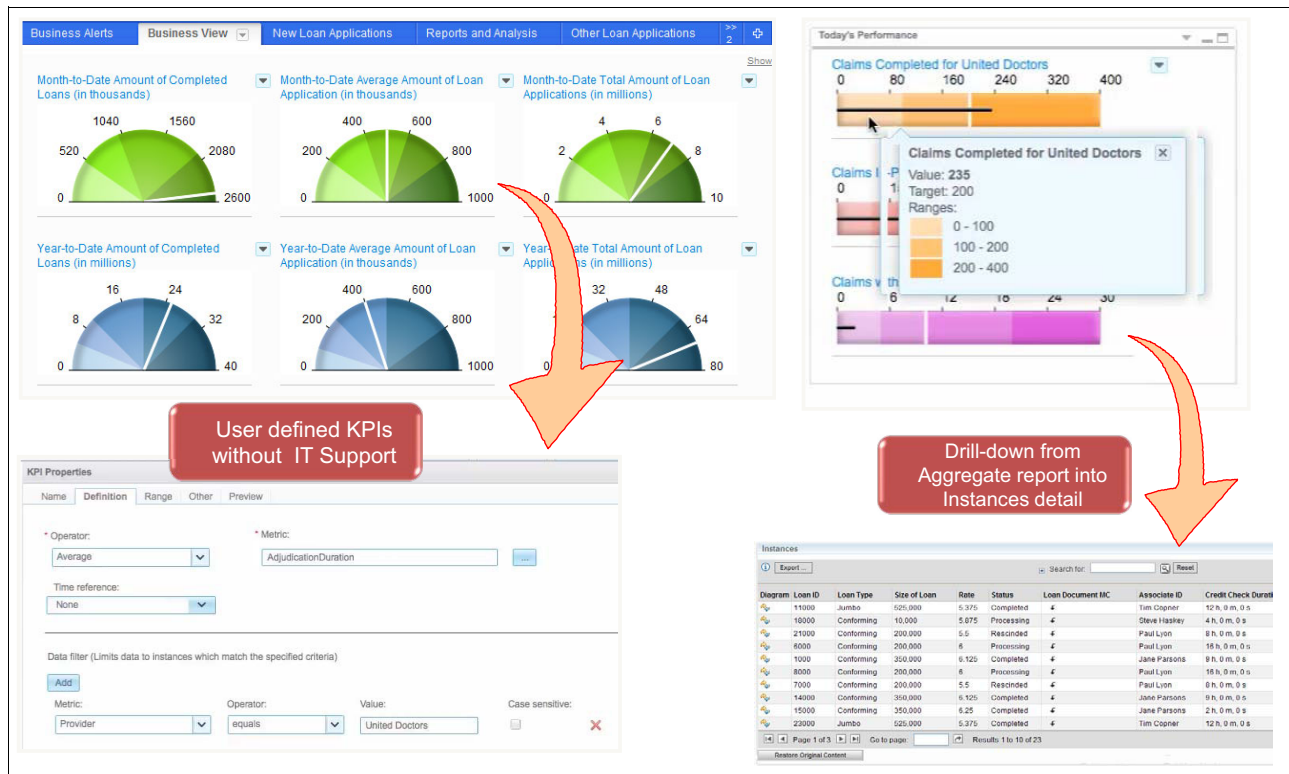


Figure 8-4 Key Performance Indicators details

Figure 8-5 on page 290 provides an overview of the Cognos reports editor included with Business Space.

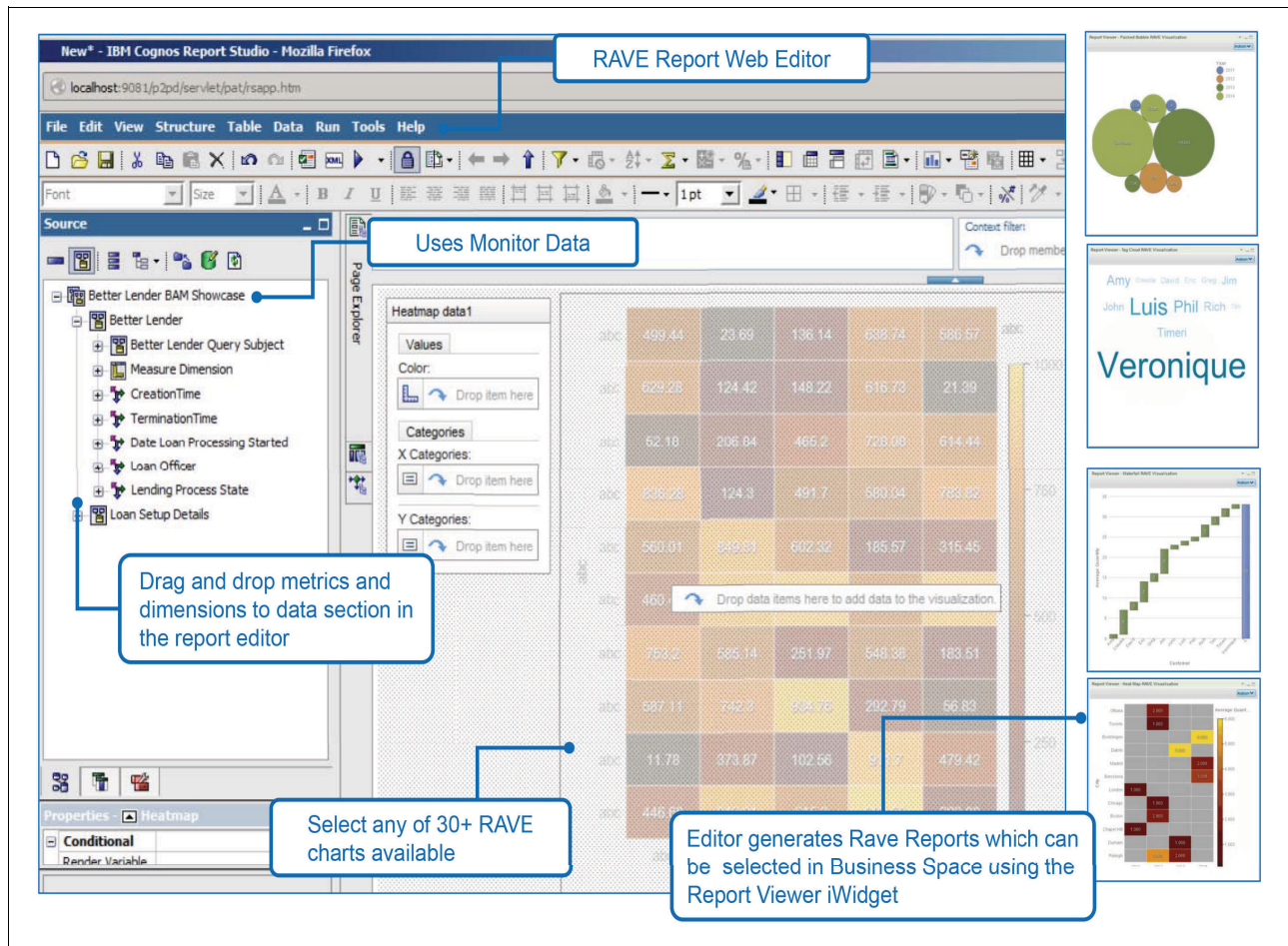


Figure 8-5 Cognos reports editors included with Business Space

8.1.2 IBM Business Monitor topologies

Figure 8-6 on page 291 shows the recommended three cluster topology for IBM Business Monitor.

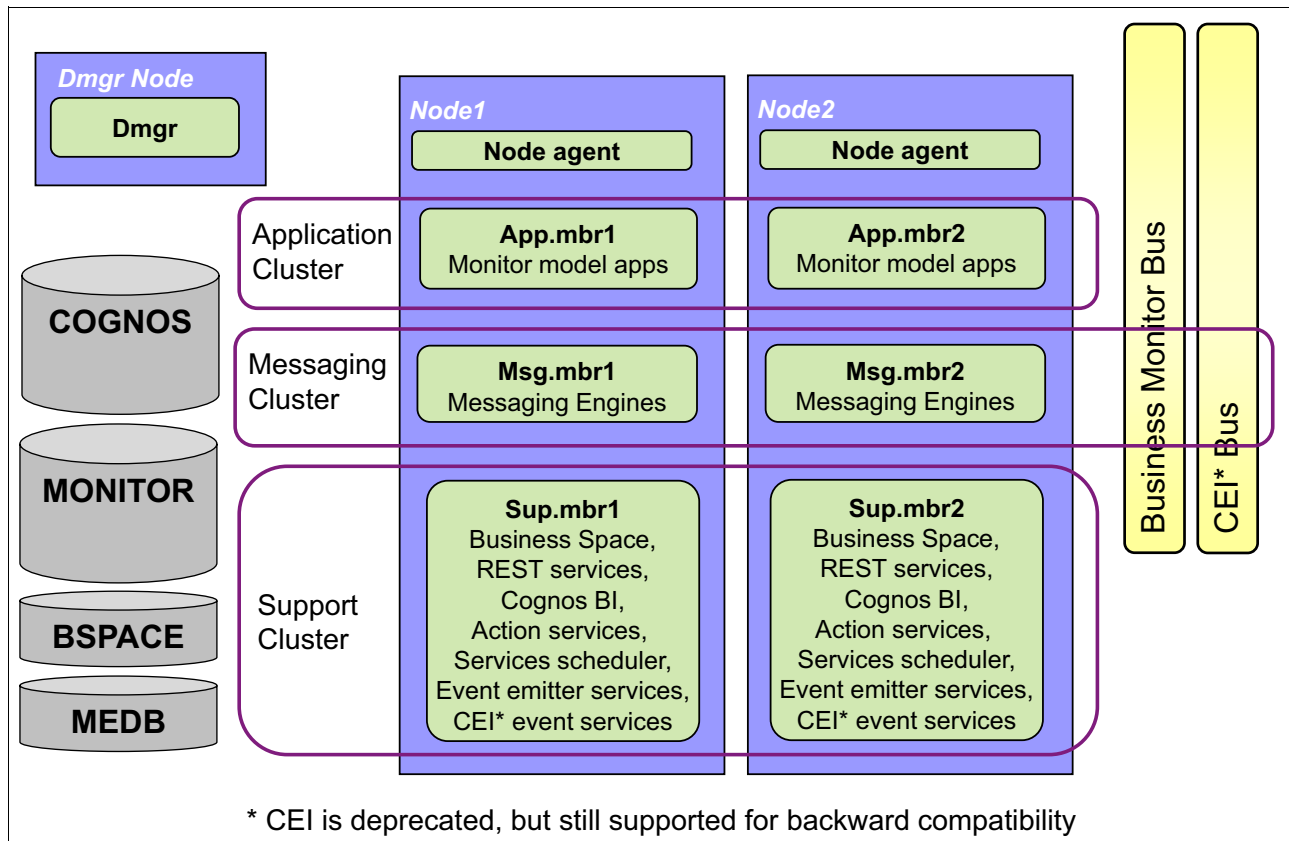


Figure 8-6 Three cluster golden topology

There are two topologies supported for Business Monitor:

- ▶ Single cluster topology
- ▶ Three cluster golden topology

A deployment environment is created by using the **monconfig** command.

The main MONITOR database stores the IBM Business Monitor configuration, monitor model metadata, and monitored data. The IBM Cognos Business Intelligence configuration is stored in a separate IBM Cognos BI content store database named COGNOSCS.

The MONITOR database is also used to store schemas for the following components during stand-alone profile creation:

- ▶ Business Space
- ▶ Common event infrastructure (CEI) messaging engine message store
- ▶ IBM Business Monitor messaging engine message store

If you are not using a stand-alone profile, you can use the same database or different databases for these components, and additionally for the CEI data store, which is not required and therefore is not created or enabled by default.

IBM Business Monitor and IBM BPM can share the same WebSphere Application Server installation directory location but cannot share the same cell. IBM BPM is not able to augment to an existing IBM Business Monitor profile and vice versa.

The advantages of having IBM Business Monitor in its own cell are many:

- ▶ You can administer IBM Business Monitor without impacting IBM BPM.

- ▶ You have independence in scheduling maintenance outages for:
 - Applying interim fixes or fix packs independently.
 - Deploying new monitor models.
 - Database maintenance activities, such as database back-ups and re-orgs.
- ▶ You can tune resources to perform optimally for IBM Business Monitor:
 - Event throughput processing for a plurality of different event sources, not just Business Process Modeling Notation (BPMN) or Business Process Execution Language (BPEL).
 - Dashboard response times.
- ▶ IBM Business Monitor exposes access to all its data through a well architected and rich REST-based API.
- ▶ The Business Monitor toolkit (WBMTK) widgets are encapsulated within coach views. The controls included are:
 - KPI Gauges: Show gauges representing the KPIs.
 - KPI Definition: Allow the user to create a new KPI definition.
 - KPI History: Display a chart of the KPI history.
 - Instances: Show a table of the instances for a model.
 - Show Models: Allow the user to pick a model.
 - Dashboard Alerts: Display a table of dashboard alerts for the user.
 - Diagram: Display the diagram associated with the monitor model.
 - Metrics Tree: Display a tree showing the metrics for selection.
 - MonitorREST: An encapsulated set of REST access functions.
 - Container: A dashboard container for monitor widgets.

Figure 8-7 on page 293 shows a coach form using the IBM Business Monitor toolkit.

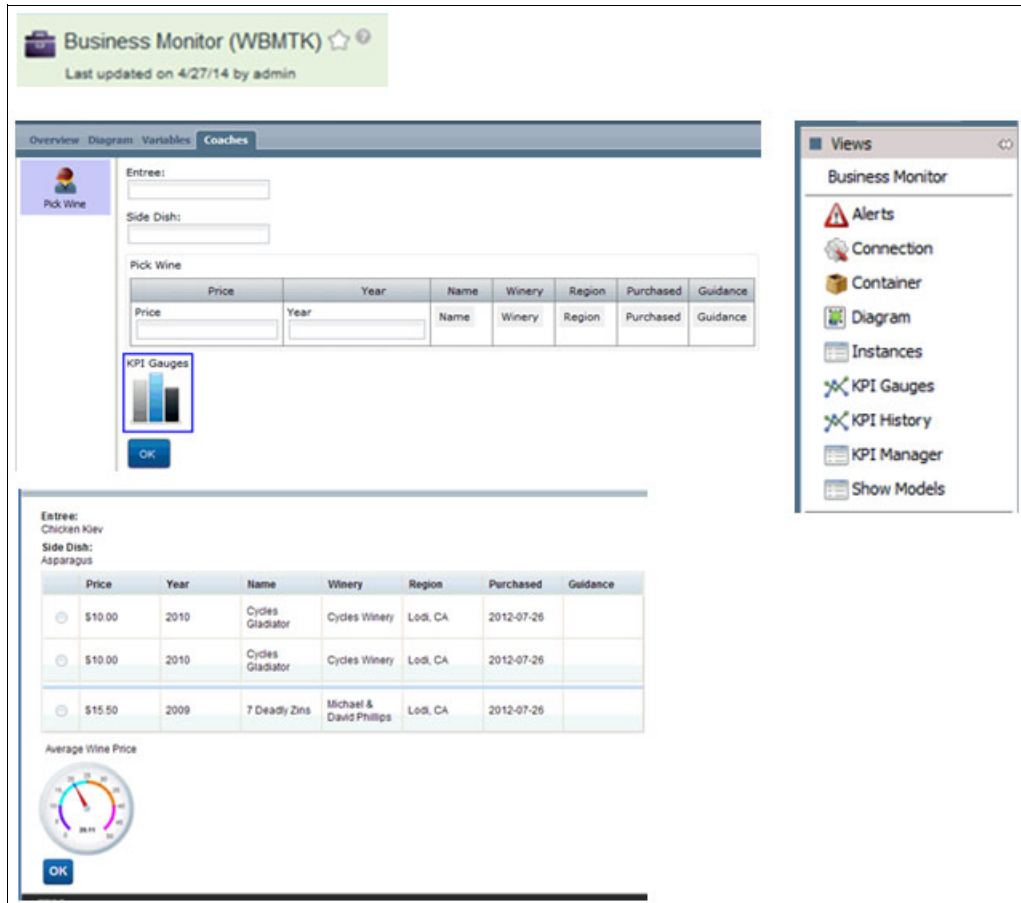


Figure 8-7 Coach form with Business Monitor toolkit coach views

8.1.3 What is new in IBM Business Monitor v8.5.5

IBM Business Monitor v8.5.5 supports the following new features:

- ▶ New Dynamic Event Framework, replacing CEI
- ▶ Simplified, faster, and more resilient
- ▶ Leverage new Cognos RAVE visualizations
- ▶ New chart types, like Gantt, Heat Map, and Tag Cloud
- ▶ Simplified configuration and migration
- ▶ Automatic configuration of remote event sources
- ▶ App-by-app migration
- ▶ Supports latest prerequisites and supporting products: WebSphere Application Server 8.5.5.2, IBM Cognos BI 10.2.1.2, IBM DB2 10.5, Internet Explorer 11

8.2 IBM BPM operational analytics

The Process Admin Console enables you to monitor the performance of the Process Servers in your environment. And, when necessary, you can view IBM Business Process Manager logs to help resolve issues.

The Process Admin Console includes an Instrumentation monitor to help identify performance bottlenecks in Process Server and to capture instrumentation data that you can use to further analyze any performance issues.

To identify performance issues with your process application, view the performance data available in the Process Monitor page of the Process Admin Console. Identify process applications that have bottlenecks, drill into the process application to identify the steps that are expensive, and learn how long it takes to run services.

8.2.1 Process Monitor Summary page

The Summary page shows you how many active services and processes are currently consuming processor resources in the current cluster. You can view the total time, total number of instances, and the total number of steps that are needed to run a service or process. You can identify the services and processes that are the most expensive, for example, a service that takes a long time to run, or a process that has many steps.

Table 8-1 shows the data displayed in the Summary page.

Table 8-1 Data displayed in the Process Monitor Summary page

Data displayed	Description
Active processes currently executing	Total number of process instances currently running on this server.
Active services currently executing	Total number of services currently running on this server that are potentially problematic.
Most expensive services	Name, total running time, and the number of steps that are required for each service that is deemed most costly on this server.
Most expensive processes	Process name, which includes the instance ID, total running time (which includes service execution time), and the number of steps that are required for each process that is deemed most costly on this server.
Most expensive service steps	Service name, step name, total running time, and total number of instances that are required to run each step that is deemed most costly on this server. If any subservices are associated with the step, the Process Monitor displays those subservice names as well.
Most expensive process steps	Process name, step name, total running time, and total number of instances that are required to run each step that is deemed most costly on this server. If any subprocesses are associated with the step, the Process Monitor displays those subprocess names as well.

8.2.2 Process Monitor Process page

The Processes page shows the data in Table 8-2 on page 295 for all processes on the server, in the current cluster.

Table 8-2 Data displayed in the Process Monitor Process page

Data displayed	Description
Active processes currently executing	Process name, which includes the instance ID, enter time (start time), duration (running time), and total number of steps for each process instance currently running on this server. In addition to processes that are running smoothly, this category also includes currently running processes that might have problems. For example, if a process instance is stuck in a repeating loop, it is shown in this list.
Active processes not currently executing	Name, last enter time (most recent start time), last duration (running time from most recent execution), total duration (cumulative running time), and total number of steps for processes that were previously started, but not currently active, on this server. This category includes process instances that are active but not running at this moment. For example, if a process instance is waiting for an event, it is included in this category.
Completed processes	Name, last enter time (most recent start time), last duration (running time from most recent execution), total duration (cumulative running time), and total number of steps for processes that ran successfully on this server.

8.2.3 Process Monitor Services page

The Services page shows the data in Table 8-3 for all services on this server:

Table 8-3 Data displayed in the Process Monitor Services page

Data displayed	Description
Active services currently executing	Name, enter time (start time), duration (running time), and total number of steps for each service currently running on this server. In addition to services that are running smoothly, this category also includes currently running services that might have problems. For example, if a service is stuck in a repeating loop, it is shown in this list.
Active services not currently executing/completed services	Name, last enter time (most recent start time), last duration (running time from most recent execution), total duration (cumulative running time), and total number of steps for services that were previously started, but not currently active, on this server and for services that ran successfully on this server. This category includes two types of services: <ul style="list-style-type: none"> ► Services that completed successfully. ► Services that were previously started but are not currently running. For example, if a service is waiting for an event, it is included in this category.

8.3 Worklight operational analytics

IBM Worklight includes a scalable operational analytics feature, the IBM Worklight Analytics Platform, which enables organizations to search logs and events collected from devices, applications, and servers for patterns, problems, and platform usage statistics.

Operational analytics data is obtained from several sources:

- ▶ Interactions of any app-to-server activity (anything that is supported by the Worklight client/server protocol, including push notification).
- ▶ Client-side logs and crashes.
- ▶ Server-side logs that are captured in traditional Worklight log files.

IBM Worklight operational analytics are feature rich, with numerous data views:

- ▶ **Dashboard view**
The dashboard view features include interaction support to see the full device usage across the platform for the last 30, 60, or 90 days. You can drill down to specific apps and app versions.
- ▶ **Devices view**
View device information, including session activity, network activity, and JSONStore analytics. Search is provided to view information about a particular device.
- ▶ **Adapters view**
View information about adapters such as invocation frequency and network latency. You can drill down to a specific adapter or procedure.
- ▶ **Servers view**
View analytics information about individual servers in a cluster and download server logs.
- ▶ **Activities view**
View analytics data on custom activities that are created with the client-side logging features.
- ▶ **Search view**
Provides a text search for client and server-side logs. Allows filtering by application, version, environment, severity, and so on.

Worklight 6.2 also continues to support the reports database feature, which was present in previous versions, but this feature represents a mere subset of the total data that is collected as part of the operational analytics feature.

The operational analytics feature is accessible from the Worklight Console and allows for near real-time analytics for client activity with the Worklight Server, analytics for adapter hits and network latency, search and download capabilities for both client and server-side logs, along with the ability to search crash and stack trace information.

In addition to an at-a-glance view of the mobile and web application analytics, the analytics feature includes the capability to perform raw searches against captured data. Additionally, Worklight provides client and server-side API function calls specifically designed so that users can feed custom data specific to their applications into the IBM Worklight Analytics Platform.

For more information about Worklight analytics, see the IBM Knowledge Center at:

http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.monit.or.doc/monitor/c_operational_analytics.html

8.3.1 Data capture

When the IBM Worklight Analytics Platform is deployed and properly configured, data flows from the Worklight Server to the IBM Worklight Analytics Platform. Some types of data are

captured and forwarded automatically, while other data set requires changes to be made in the client application to capture or forward the data to the Worklight Server.

All data forwarded to the IBM Worklight Analytics Platform is categorized by type. The different types of analytics data captured and analyzed are described in the following sections.

Note: No data is sent to the Worklight Server until the application is connected to the server. A connection occurs when a call to `WL.Client.connect()` is made or on the first successful call to an adapter in the Worklight Server.

App activities

All IBM Worklight client/server network communication is considered to be an app activity. An app activity is sent to the IBM Worklight Analytics Platform when:

- ▶ A client device begins a new session with the Worklight Server.
- ▶ A client device makes an adapter request.

When the client communicates with the Worklight Server through one of the previously mentioned events, it also sends metadata about the device, including:

- ▶ Device environment (Android, iOS, Windows Phone, and so on).
- ▶ Device model (iPhone, iPad, Nexus, and so on).
- ▶ Device OS version (6.2, 4.2.2, and so on).

Extra information that is captured during a client/network communication includes:

- ▶ Response times for adapter calls.
- ▶ Response payload sizes for adapter calls.

Server logs

Normal Worklight Server activity produces log messages that are saved to the disk. These messages are also forwarded to the IBM Worklight Analytics Platform and can be searched.

Client logs

Client devices can be configured to capture log data and crash events to be forwarded to the Worklight Server. For more information, see “Manually captured data” on page 297.

Notification activities

Upon a successful push notification, a notification activity is automatically sent to the IBM Worklight Analytics Platform.

Manually captured data

The following data must be captured manually by changing the client application.

Client logs

Client logs include custom data obtained using the client-side log capture capabilities in hybrid or native applications. All persisted client-side logs are sent automatically upon a successful initialization (successful session start) with the Worklight Server. An explicit API is provided if you want to send the logs more frequently.

Client network activities

Network activities for client devices are captured and persisted on the device automatically. However, they are only sent when the client successfully initializes with the Worklight Server (successful session start).

Analytics logging

The client-logging feature enables developers to create logs to help debug problems and capture errors. A separate API exists for creating logs that are not meant for problem detection and capture.

JSONStore analytics

You can collect key pieces of analytics information that are related to JSONStore with the Worklight platform.

8.3.2 Analytics views

The IBM Worklight Analytics Platform provides a range of analytics features contained in various views, each accessible from a tab. The data is segmented into six interactive views: dashboard, devices, adapters, servers, activities, and log search.

Dashboard view

The analytics dashboard view presents a range of measures about the IBM Worklight system. It is designed to give users a quick look of application usage.

The analytics dashboard displays the following information and graphs. See Figure 8-8 on page 299:

- ▶ Total, device, and web session data
- ▶ Application and environment usage
- ▶ Adapter calls
- ▶ Past 12-hour data
- ▶ New users

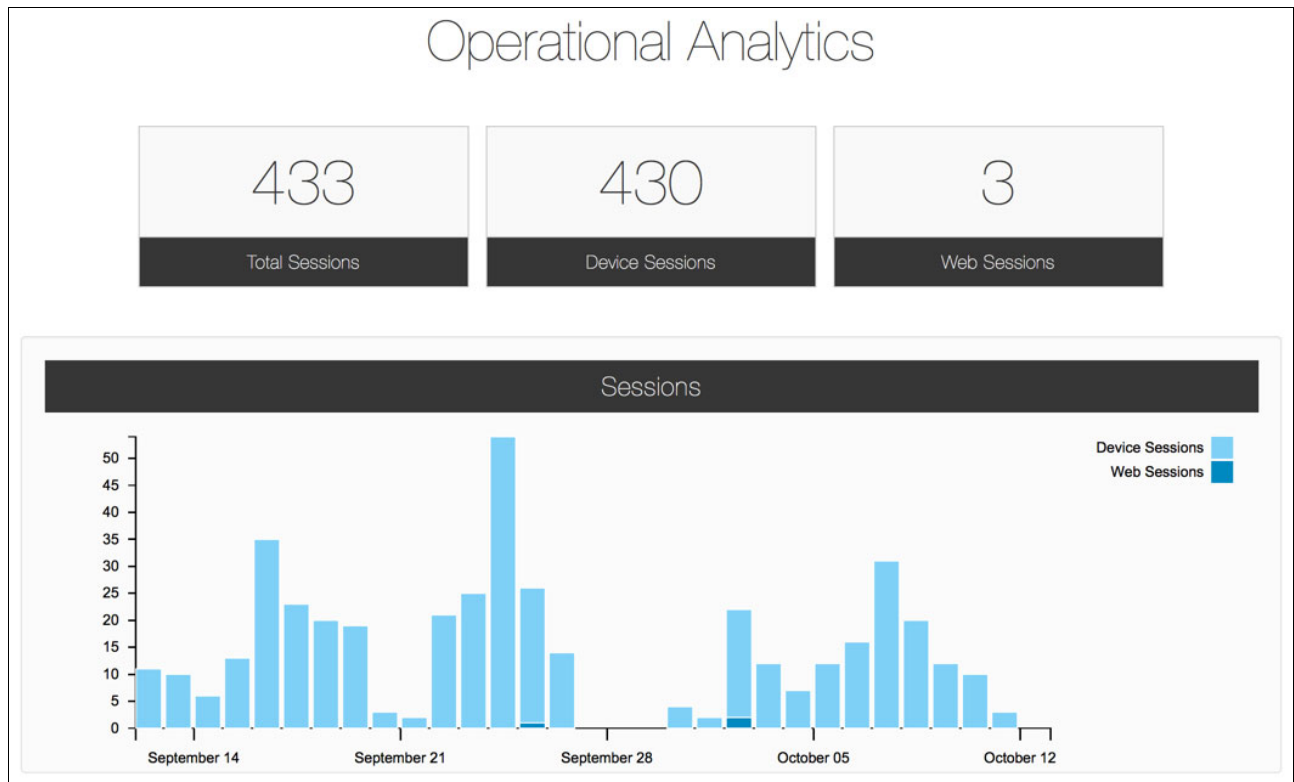


Figure 8-8 Operational analytics dashboard view

Devices view

The devices dashboard view is focused on device-specific analytics and displays the following information:

- Number and type of devices (including OS version). See Figure 8-9 on page 300.

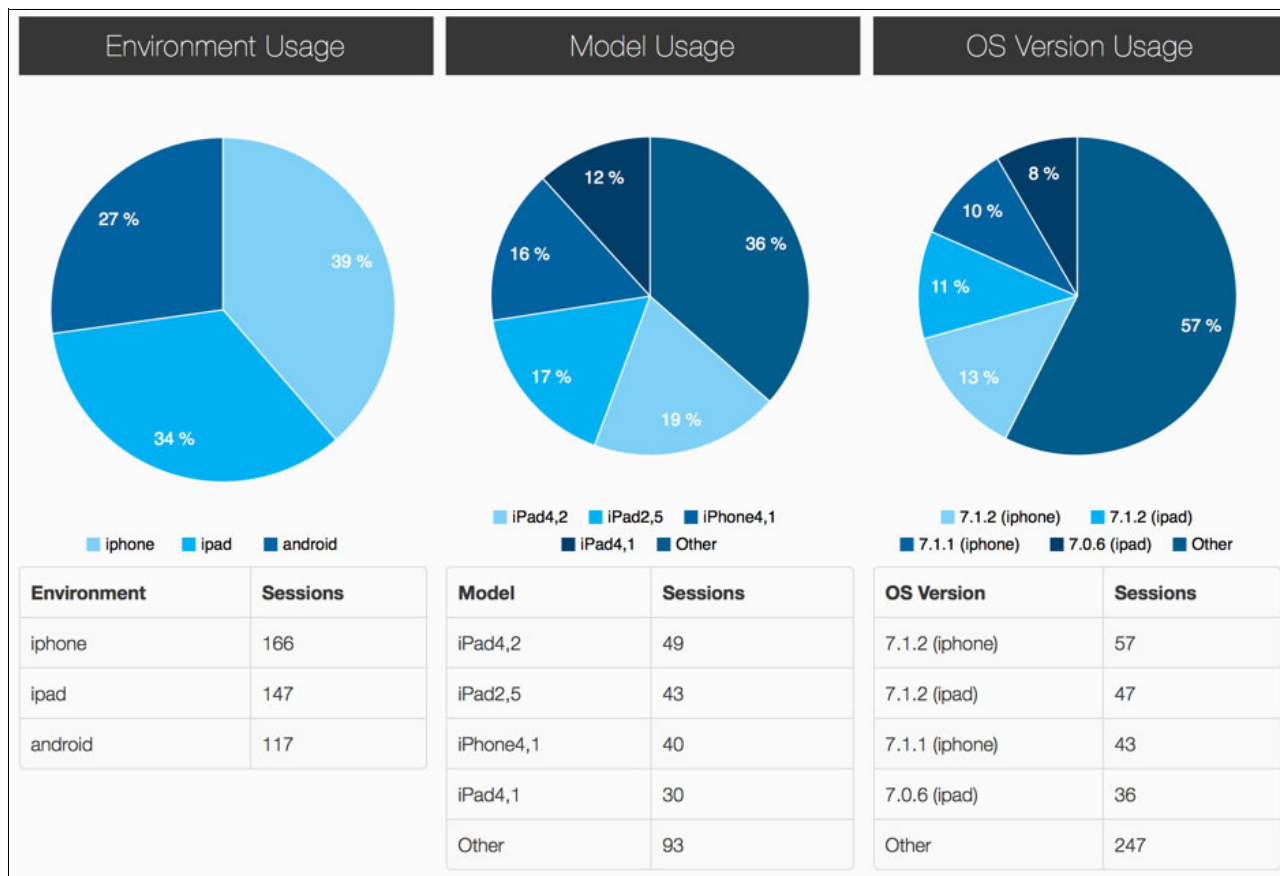


Figure 8-9 Device-specific analytics by type, model, and OS version

- Total session and adapter calls.
- Average response times (with embedded packet size). See Figure 8-10 on page 301.

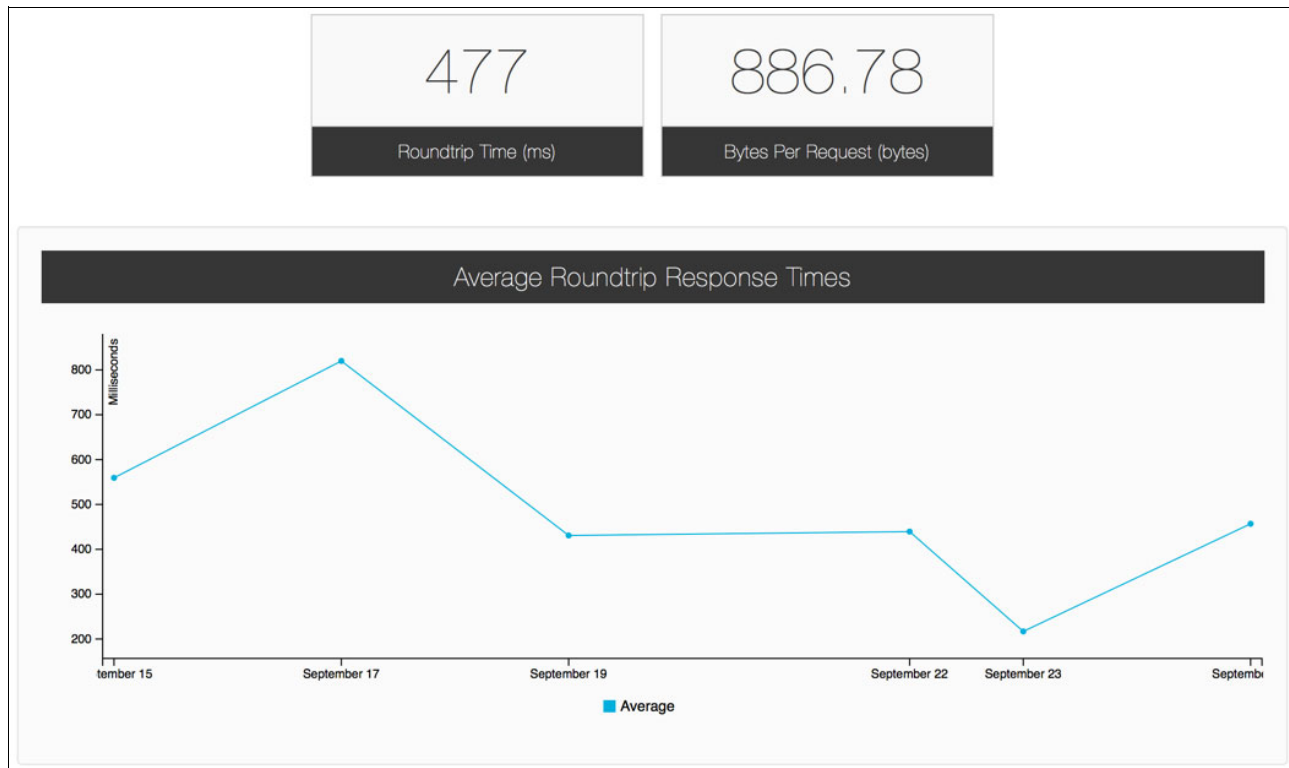


Figure 8-10 Network response time and response size

- JSONStore metrics
- Device search (with sessions, log download, and usage flow)

Adapters view

The adapters dashboard view is focused on adapter-based analytics and displays the following information:

- Adapter calls per server and procedure.
- Round-trip response times (with device and server distinction). See Figure 8-11.

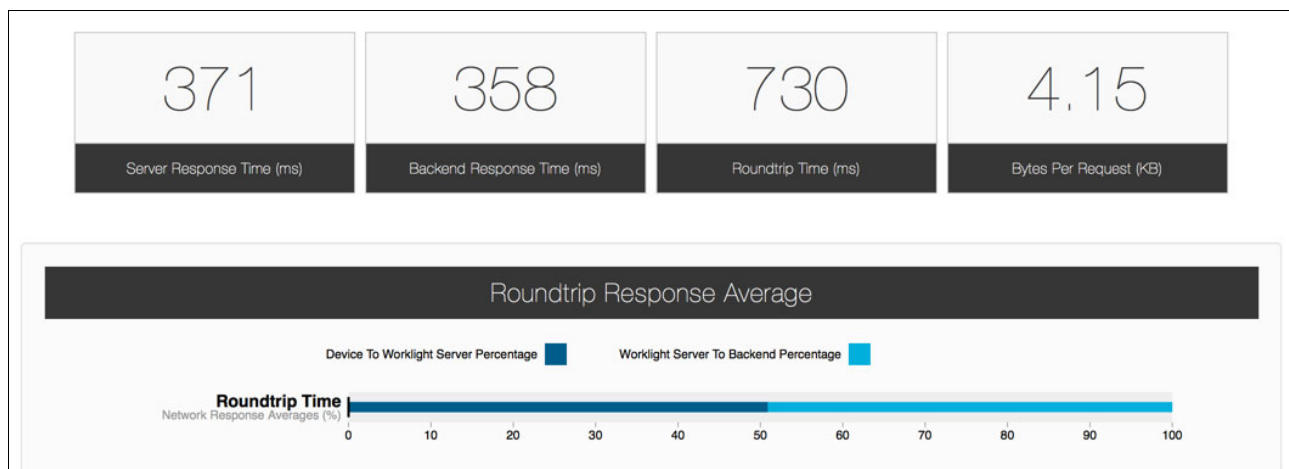


Figure 8-11 Adapter round-trip response time including network and server latency

- Total adapter calls and calls per user. See Figure 8-12 on page 302.
- Bytes per request.

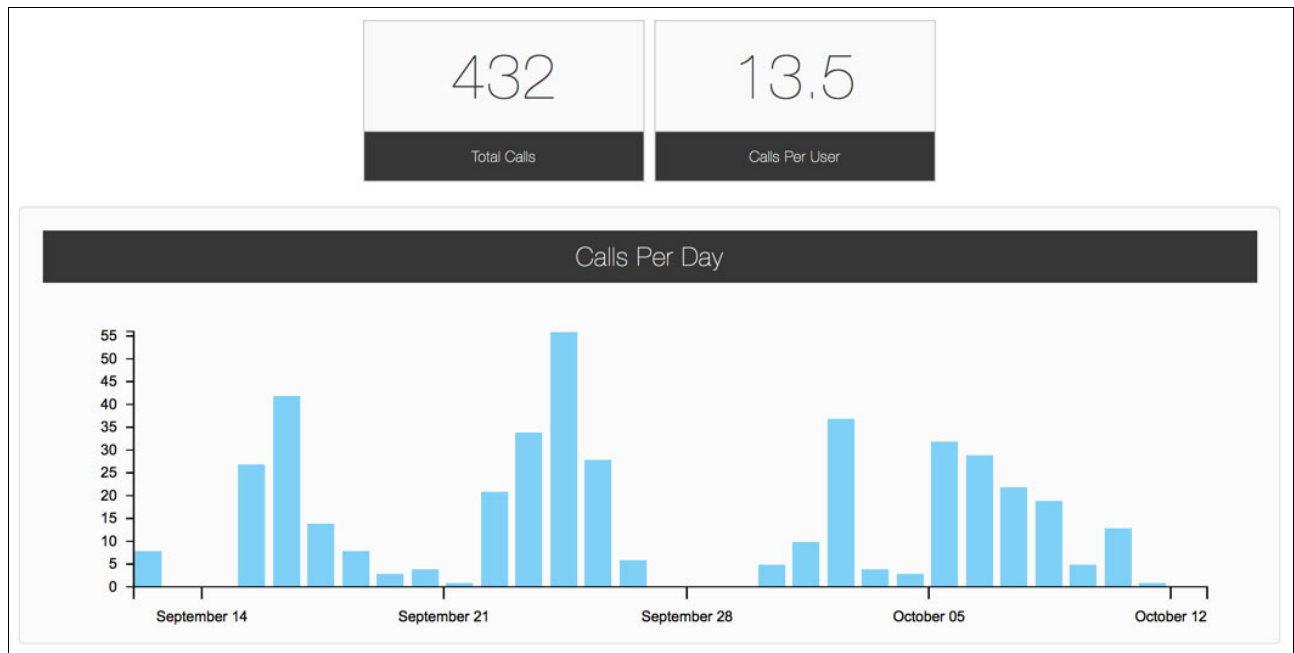


Figure 8-12 Adapter calls invoked per day

Servers view

The servers dashboard view is focused on server-side analytical data and displays the following information:

- Server sessions and usage. See Figure 8-13 on page 303.

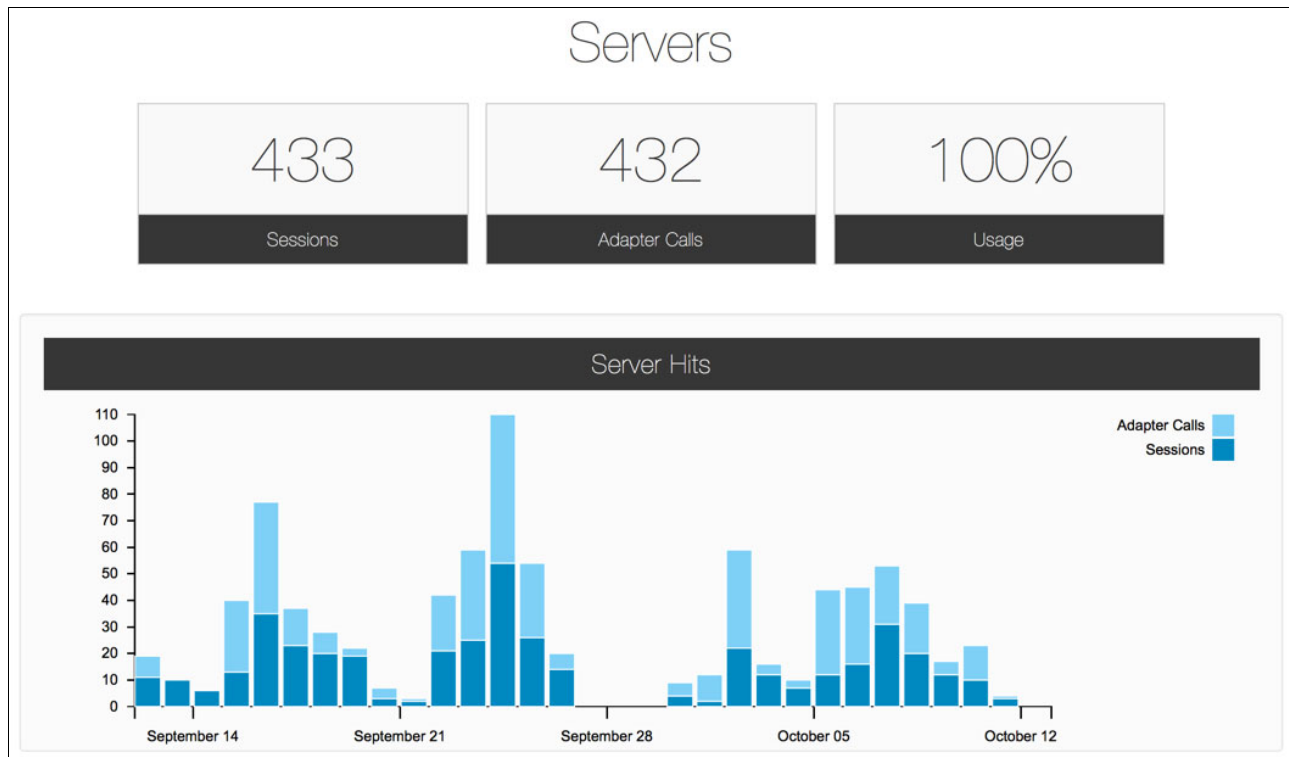


Figure 8-13 Server hits by day, distinguishing adapter, and session calls

- ▶ Server network usage flow
- ▶ Adapter hits
- ▶ Weekly log severities (info, fine, warn, severe). See Figure 8-14.

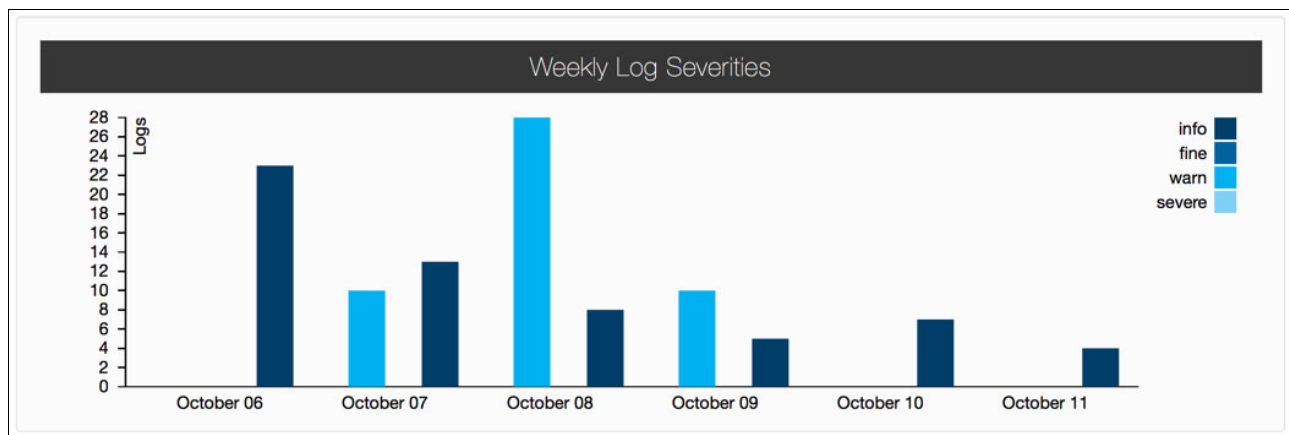


Figure 8-14 Weekly view of log severity data

- ▶ Server logs (including download)

Activities view

The activities dashboard view is focused on log and notification activities and displays the following information:

- ▶ Searchable activities
- ▶ Number of hits and calls per application. See Figure 8-15 on page 304.

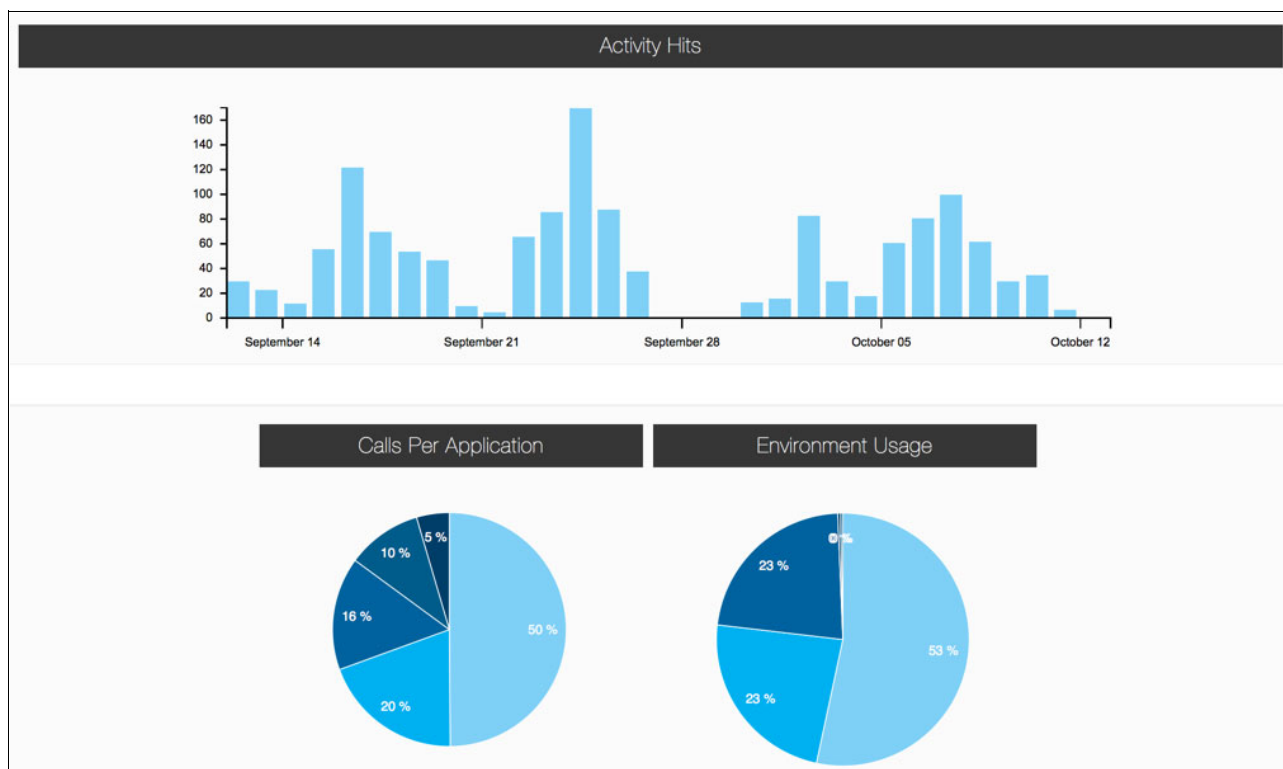


Figure 8-15 Activity hits by application and environment

- Notification hits per day. See Figure 8-16.

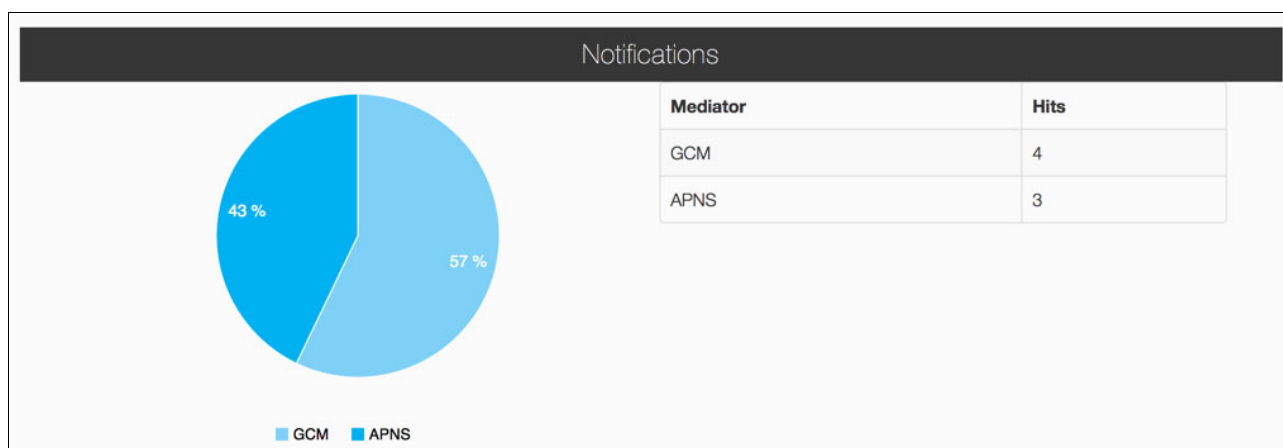


Figure 8-16 Notification report by mediator utilized

- Notification hits per service, for example, Apple Push Notification Services (APNS) and Google Cloud Messaging (GCM).
- Environment usage.

Log search view

The Log Search dashboard view is focused on client and server-side logging. In the log search view, users can search for logs based on exceptions, device type, application, or numerous other search parameters including wildcard searches. See Figure 8-17 on page 305.

Log Search

Client Logs
Server Logs

Search:

Show:
All Applications ▾
All Versions ▾
All Levels ▾
All Packages ▾

All Environments ▾
All Models ▾
All OS Versions ▾

From: All Days
To: All Days

Figure 8-17 Log searching capabilities for both client and server logs

When the search results have been narrowed down, the individual logs can be downloaded and viewed. See Figure 8-18.

Saturday, October 11, 2014 3:41 PM

Server: ext4lnx.demos.ibm.com/192.84.47.10 **Level:** INFO

FWLSE0085I: Deployed app 'AppCenter-iphone-2.06' successfully on Worklight Server [project worklight]

[View More](#)

Date: Saturday, October 11, 2014 3:41 PM

Server: ext4lnx.demos.ibm.com/192.84.47.10

Severity Level: INFO

Thread Id: 8

Source Class: com.worklight.gadgets.bean.WidgetServiceBean

Source Method Name: handleSingleEnvironmentDeployable

Logger Name: com.worklight.gadgets.bean.WidgetServiceBean

Message:

FWLSE0085I: Deployed app 'AppCenter-iphone-2.06' successfully on Worklight Server [project worklight]

Download logs for surrounding:

Figure 8-18 Log event overview and download



A

Samples included with this book

This appendix includes the summary information for the samples provided with this IBM Redbooks publication. This appendix walks through the steps required to configure, deploy, and run the samples. A reference regarding how to test mobile applications on an Android device is also included.

For information about downloading the samples, see Appendix B, “Additional material” on page 319.

Disclaimer: IBM DOES NOT WARRANT OR REPRESENT THAT THE CODE PROVIDED IS COMPLETE OR UP-TO-DATE. IBM DOES NOT WARRANT, REPRESENT OR IMPLY RELIABILITY, SERVICEABILITY OR FUNCTION OF THE CODE. IBM IS UNDER NO OBLIGATION TO UPDATE CONTENT NOR PROVIDE FURTHER SUPPORT.

ALL CODE IS PROVIDED "AS IS," WITH NO WARRANTIES OR GUARANTEES WHATSOEVER. IBM EXPRESSLY DISCLAIMS TO THE FULLEST EXTENT PERMITTED BY LAW ALL EXPRESS, IMPLIED, STATUTORY AND OTHER WARRANTIES, GUARANTEES, OR REPRESENTATIONS, INCLUDING, WITHOUT LIMITATION, THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF PROPRIETARY AND INTELLECTUAL PROPERTY RIGHTS.

YOU UNDERSTAND AND AGREE THAT YOU USE THESE MATERIALS, INFORMATION, PRODUCTS, SOFTWARE, PROGRAMS, AND SERVICES, AT YOUR OWN DISCRETION AND RISK AND THAT YOU WILL BE SOLELY RESPONSIBLE FOR ANY DAMAGES THAT MAY RESULT, INCLUDING LOSS OF DATA OR DAMAGE TO YOUR COMPUTER SYSTEM.

IN NO EVENT WILL IBM BE LIABLE TO ANY PARTY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES OF ANY TYPE WHATSOEVER RELATED TO OR ARISING FROM USE OF THE CODE FOUND HEREIN, WITHOUT LIMITATION, ANY LOST PROFITS, BUSINESS INTERRUPTION, LOST SAVINGS, LOSS OF PROGRAMS OR OTHER DATA, EVEN IF IBM IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

THIS EXCLUSION AND WAIVER OF LIABILITY APPLIES TO ALL CAUSES OF ACTION, WHETHER BASED ON CONTRACT, WARRANTY, TORT OR ANY OTHER LEGAL THEORIES.

Samples compressed file content

Perform the following steps to extract the sample projects included with this book:

1. Download the SG248240.zip file on your desktop as described in Appendix B, “Additional material” on page 319.
2. Extract the SG248240.zip. You should have a directory structure as described Table A-1.

Table A-1 Description of files bundled in the sample

Scenario	File Path	Description
Scenario 1	/Scenario1/Bluemix/CustomerOrder.json	JSON file used to create a data class in the Bluemix Mobile Data service in scenario 1 step 4 on page 184 of Section 6.6.6, “Implementing the Schedule Work Order system activity” on page 184.
Scenario 1	/Scenario1/Bluemix/CustomerProfile.json	JSON file used to create a data class in the Bluemix Mobile Data service in scenario 1 step 9 on page 177 of section 6.6.5, “Implementing the Create Customer Profile system activity” on page 175.
Scenario 1	/Scenario1/Bluemix/Location.json	JSON file used to create a data class in the Bluemix Mobile Data service in scenario 1 Step 14 on page 178 of section 6.6.5, “Implementing the Create Customer Profile system activity” on page 175.
Scenario 1	/Scenario1/Bluemix/app.js	Node.js source code for Bluemix Mobile Cloud project in scenario 1 step17 on page 187 section 6.6.6, “Implementing the Schedule Work Order system activity” on page 184.
Scenario 1	/Scenario1/Worklight/Scenario1_Worklight_Project.zip	Worklight Project interchange file for scenario 1 section 6.4.1, “Worklight project” on page 128.
Scenario 1	/Scenario1/BPM/Field_Force_Automation - Scenario_1.twx	IBM BPM artifacts for scenario 1.

Scenario	File Path	Description
Scenario 1	/Scenario1/BPM/PushNotificationAdapter.zip	<p>Eclipse Java project to be used to generate a JAR file containing the Java integration adapter that can be imported into IBM BPM for scenario 1 used in section 6.6.7, “Implementing the Send Notification to Field Staff activity” on page 195.</p> <p>Note: The hostname and application identifier/secret for Bluemix for your environment must be inserted into the source code before compiling the JAR file.</p>
Scenario 1	/Scenario1/BPM/BluemixAdapters.zip	<p>Eclipse Java project to be used to generate JAR file containing the Java integration adapter that can be imported into IBM BPM for scenario 1 “Create the IBM BPM Bluemix integration service” on page 179 and “Create the Bluemix integration service” on page 191.</p> <p>Note: The hostname and port number for the Worklight Server for your environment must be inserted into the source code before compiling the JAR file.</p>
Scenario 2	/Scenario2/Bluemix/parts-bluemix.json	JSON file used to create the parts database in Bluemix Mobile Data used in 7.6.1, “Creating the Parts database in Bluemix” on page 230.
Scenario 2	/Scenario2/Worklight/Scenario2_Worklight_Project.zip	Worklight project interchange file for scenario 2 in 7.2, “Parts app” on page 210.
Scenario 2	/Scenario2/BPM/Field_Force_Automation - Scenario_2_v1.twx	IBM BPM artifact for scenario 2. This .twx file also includes all the artifacts created for scenario 1.
Scenario 2	/Scenario2/LTPA/login.html /LTPA/loginError.html	Files to be included in Worklight WAR file root level for authentication.

Scenario 1 sample instructions

This section describes how to deploy the Worklight and IBM BPM artifacts to run the sample for scenario 1 in your environment.

IBM BPM scenario 1 sample artifacts

Perform the following steps to deploy the IBM BPM artifacts for scenario 1 described in Chapter 6, “Scenario 1: Getting started” on page 111:

1. Log in to IBM Process Center at `https://<hostname>:<port>/ProcessCenter/login.jsp`. Enter your user name and password and click **Log In** (see Figure A-1).

Figure A-1 IBM Process Center login page

2. Select the **Process Apps** tab.
3. Click the **Import Process App** option (link on the upper right). See Figure A-2 on page 311.

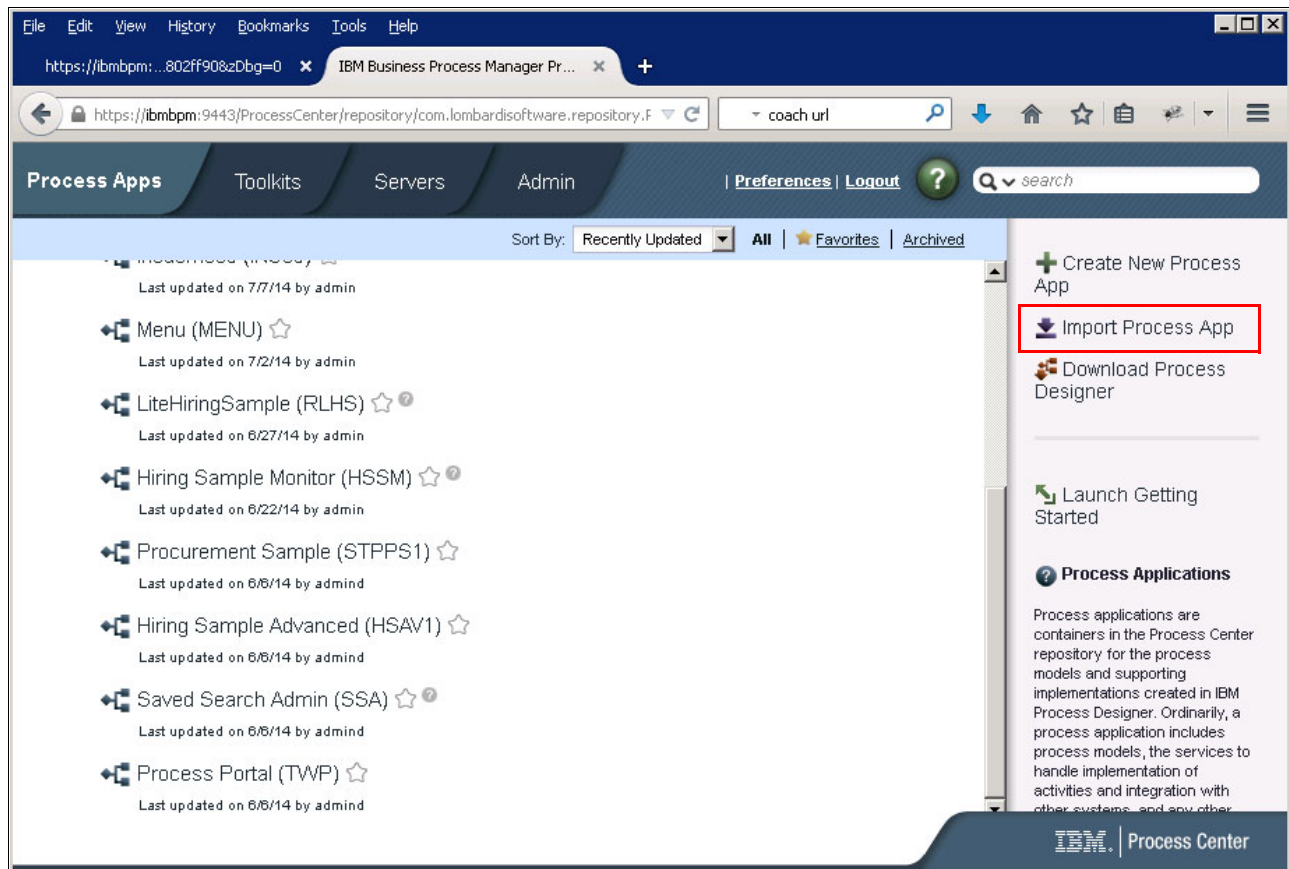


Figure A-2 Top page of Process Center

4. Browse to find the .twx file to import (Figure A-3 on page 312).

The path for the .twx file for scenario 1 is /Scenario1/BPM/Field_Force_Automation - Scenario_1.twx. After you select the .twx file to import, click **OK**.

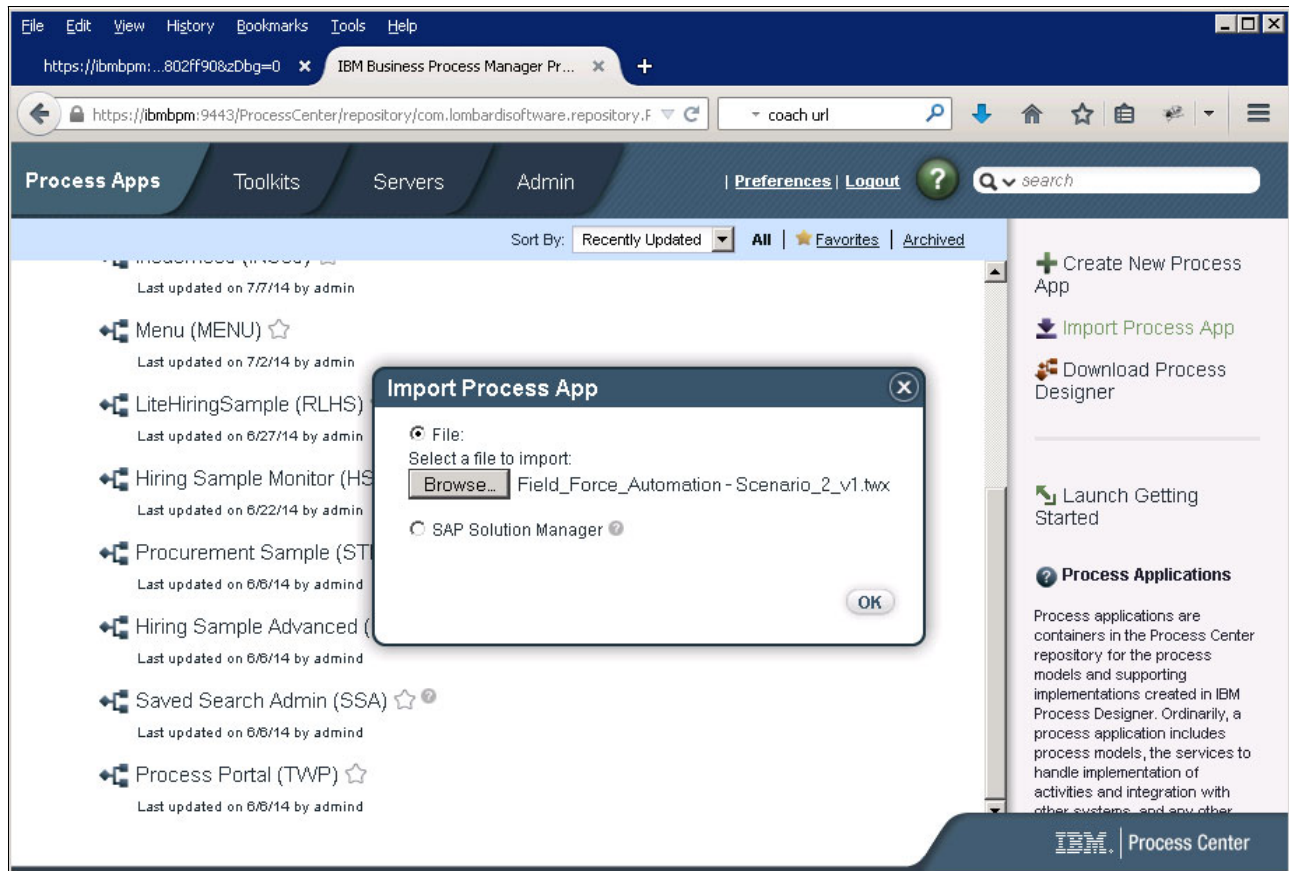


Figure A-3 Select the .twx file to import

5. Click **Import** to import the selected file.

The sample .twx file contains two Java integration adapters that have been compiled with the host names, ports, and application identifiers configured for the IBM Redbooks test system.

6. Replace the host names, ports, and application identifiers for the appropriate values in your environment.
7. Recompile the Java integration adapter source code, generate new JAR files, and reimport them into the IBM BPM application.

The source code included in the sample files has placeholders for where these items need to be inserted.

See section “IBM BPM integration services and IBM Bluemix” on page 312 for a description of preparing the Java integration adapters

IBM BPM integration services and IBM Bluemix

This section describes how to implement the IBM BPM and Bluemix integration services described in scenario 1, Chapter 6, “Scenario 1: Getting started” on page 111.

IBM BPM Java integration adapters

Perform the following steps:

1. Launch Eclipse and select **File** → **Import** → **Existing Projects into Workspace**.

2. For Select archive file, select the **BluemixAdapter.zip** file provided with the samples.
3. Click **Finish**.

The Java project requires the Apache HTTP Client library. It can be downloaded from *The Apache Software Foundation* at <http://hc.apache.org>.

4. In Project Explorer, right-click the imported project and select **Properties** → **Java Build Path**.
5. Select the **Libraries** tab.
6. The Apache http-client libraries might need to be reattached if the path is different from the target path where they will be stored. Select each library item, click **Edit**, and select the file from its revised location.
7. Make the necessary changes to the placeholders in the code denoted by the pattern {insert here}.
8. Follow the instructions in section “Create the IBM BPM Bluemix integration service” on page 179 to compile and export the JAR file.
9. The exported JAR file can be imported into the IBM BPM project to replace the JAR file that had been imported by the authors of this book.

In IBM Process Designer, select **Files** and double-click the file **BluemixMobileDataService.jar**. Ensure that the file name is the same so that IBM BPM can replace the file.

10. Click **Browse** under the File section and select the JAR that you just exported (see Figure A-4).
11. Click **Save**.

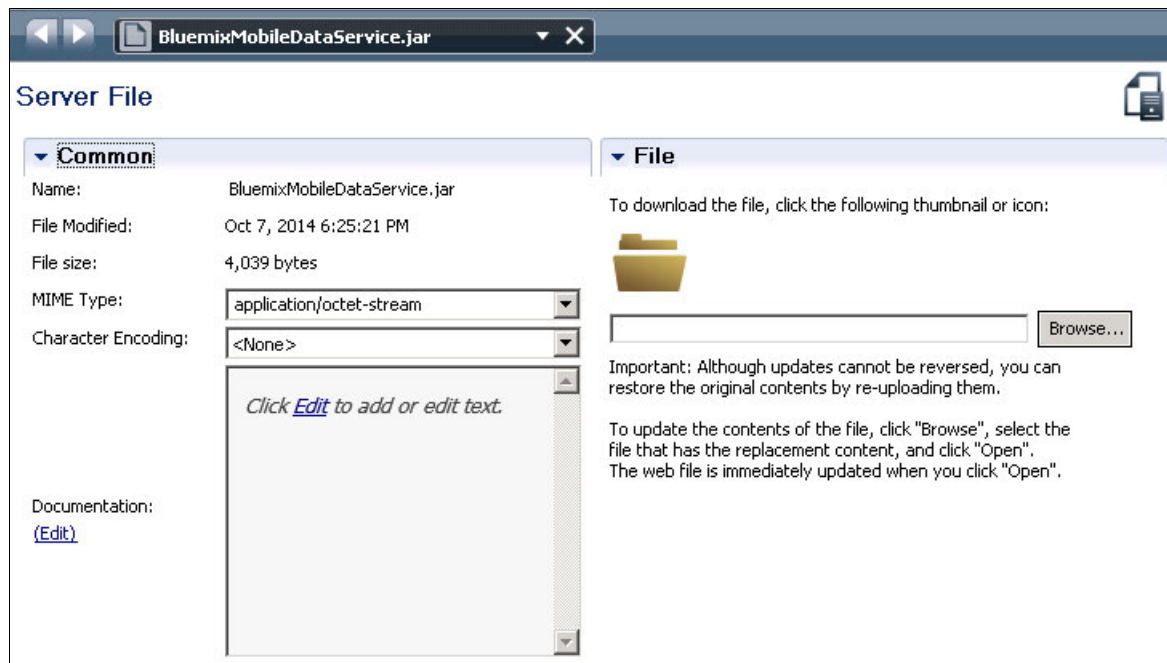


Figure A-4 Replacing the *BluemixMobileDataService.jar* file that contains the IBM BPM Java integration adapter

12. Repeat step 1 on page 312 through step 11 for the *PushNotificationAdapter.zip* file containing the other Java integration adapter used by IBM BPM in scenario 1. After making the changes to the code and recompiling to generate a new .JAR file, ensure that

the .JAR file name matches the file name in IBM BPM that is being replaced (WorklightService.jar).

IBM Bluemix Node.js application

Follow the instructions as outlined in section “Create the Bluemix Node.js service” on page 184 to create the Bluemix application.

As part of step 17 on page 187 in section “Create the Bluemix Node.js service” on page 184, paste the source code /Scenario1/Bluemix/app.js referenced in Table A-1 on page 308 into the text editor being used to edit the app.js once the Node.js project has been cloned locally.

IBM Worklight scenario 1 sample artifacts

This section describes the configuration and deployment of the Worklight artifacts included in the scenario 1 sample.

Import the sample project interchange file into Worklight Studio

Perform the following steps:

1. Create a new workspace in Worklight Studio for scenario 1.
2. In Worklight Studio, navigate to **File** → **Import** → **Existing Projects into Workspace** and click **Next**.
3. Select **Select archive file**.
4. Click **Browse**. Select the archive file **Scenario1/Worklight/Scenario1_Worklight_Project.zip** and click **Finish**.

The project WL_BPM_Project is imported.

Note: To build the project successfully, you *might* need to delete the generated Android projects WL_BPM_ProjectFieldServiceAndroid and WL_BPM_ProjectPartsAndroid, and then build all environments again to regenerate them.

To delete the project:

1. From the Project Explorer view in Worklight Studio, right-click the Android project **WL_BPM_ProjectFieldServiceAndroid** or **WL_BPM_ProjectPartsAndroid**.
2. Click **Delete**.
3. Select **Delete project contents on disk (cannot be undone)**.
4. Click **OK**.

After deleting the projects, build all environments for each app:

1. Right-click **/apps/Parts** and **/apps/FieldService**.
2. Click **Run As** → **Build All Environments**.

Update the IBM BPM connection information

The connection information such as IP addresses and ports in the sample adapters must be updated with the corresponding information for your environment.

In Worklight Studio, WL_BPM_Project project, perform the following changes:

1. Navigate to the folder **adapters/AuthenticationAdapter**.
2. Open the file **AuthenticationAdapter.xml**.

3. Change the IP address and port number to the IP address and port number of the REST interface for the IBM BPM Server. Write the IP address in `<domain>0.0.0.0</domain>` and write the port number in the `<port>0000</port>`.
4. Change the IP address and ports in the same way for the file **adapters/OrderAdapter/OrderAdapter.xml**.

Update the Bluemix connection information

In LocationAdapter, the Bluemix connection information for geolocation service must be updated as follows:

1. In the Worklight Studio, WL_BPM_Project project, open the **adapters/LocationAdapter/LocationAdapter.xml** file.
2. Change the `<domain/>` tag in the connection information to the domain of the application you created in Bluemix for the geolocation service, for example, `myapp.mybluemix.net` or `myapp.ng.bluemix.net`.

Update the push notification GCM information

Create a senderID and key pair for the application to receive push notifications. The details to set up push notifications for Android devices are documented in *Setting up push notifications for Android* at the following IBM Knowledge Center link:

http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.dev.doc/devref/t_setting_up_push_notification_android.html?lang=en

Note: The Worklight Server must have access to Google Cloud Messaging (GCM) servers to send push notifications to devices. Make sure the ports mentioned in the instructions in *Setting up push notifications for Android* are open for the Worklight Server.

1. In Worklight Studio for the WL_BPM_Project project, open the **apps/FieldService/application-descriptor.xml** file for the Field Service app.
2. Under the **Android environment** tag. Change the values in the tag `<pushSender key="key" senderId="senderId"/>` to the key and senderID you obtained following the instructions in *Setting up push notifications for Android*.

Deploy the Worklight project on the Worklight Server

The WL_BPM_Project project is now ready for deployment on Worklight Server. Follow the steps described in *Deploying an application from development to a test or production environment* at the following IBM Knowledge Center link:

http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.deploy.doc/devref/t_transporting_the_app.html?lang=en

Scenario 2 sample instructions

This section describes how to deploy the Worklight and IBM BPM artifacts to run the sample for scenario 2 (described in Chapter 7, “Scenario 2: Advanced features” on page 207) in your environment.

Note: The push notification feature from scenario 1 was not tested in the sample for scenario 2. Scenario 2 sample focuses on highlighting the features of LTPA authentication, device single-sign on, simple data sharing, and communications with Bluemix from Worklight adapters.

IBM BPM scenario 2 sample artifacts

The same instructions documented for scenario 1 in section “IBM BPM scenario 1 sample artifacts” on page 310 apply to scenario 2 but using the .twx file under directory /Scenario2/BPM/.

IBM Bluemix

For the Parts database, follow instructions in 7.6.1, “Creating the Parts database in Bluemix” on page 230 to create the data service on Bluemix.

IBM Worklight scenario 2 sample artifacts

This section describes the configuration and deployment of the Worklight artifacts for the scenario 2 sample.

Import the sample project interchange into Worklight Studio

Perform the following steps:

1. Create a new workspace in Worklight Studio for scenario 2.
2. In Worklight Studio, navigate to **File** → **Import Existing Projects into Workspace** and click **Next**.
3. Select **Select archive file**.
4. Click **Browse**. Select the archive file **Scenario2/Worklight/Scenario2_Worklight_Project.zip** and click **Finish**.

The project WL_BPM_Project is imported.

Update the IBM BPM connection information

The connection information such as IP addresses and ports in the sample adapters must be updated with the corresponding information in your environment.

In Worklight Studio, WL_BPM_Project project, perform the following changes:

1. Navigate to the folder **adapters/OrderAdapter**.
2. Open the file **OrderAdapter.xml**.
3. Change the IP address and port number to the IP address and port of the REST interface for the IBM BPM Server. Write the IP address in **<domain>0.0.0.0</domain>** and write the port number in **<port>0000</port>**.
4. Change the IP address and ports in the same way for the file **adapters/PartOrderAdapter/PartOrderAdapter.xml**.

Update Bluemix connection information

The Bluemix connection information for the parts data service must be updated as follows:

1. In Worklight Studio, WL_BPM_Project project, open the file **adapters/PartAdapter/PartAdapter.xml**.
2. Change the <domain/> tag in the connection information to the domain of the application you created in Bluemix for the parts data service.

Note: Since Bluemix Mobile Data Services are used in this sample, the default domain is `mobile.ng.bluemix.net` and the port is 443.

The application ID and secret must be updated inside the adapter code:

1. Open **adapters/PartAdapter/PartAdapter-impl.js**.
2. At the top of the file, change the `appId` and `secret` variables to the values you got when creating the Bluemix application, as in Example A-1.

Example A-1 Placeholders for Bluemix application ID and secret

```
var appId = 'appId';  
var secret = 'secret';
```

LTPA Configuration

Follow instructions in 7.4, “SSO configuration with LTPA implementation” on page 217 to configure LTPA between Worklight and the IBM BPM server. Samples of the required `login.html` and `loginError.html` are included in the compressed file package under the `/Scenario2/LTPA` directory.

Deploy the Worklight project on Worklight Server

The WL_BPM_Project project is now ready for deployment on Worklight Server. Follow the steps described in *Deploying an application from development to a test or production environment* at the following IBM Knowledge Center link:

http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/com.ibm.worklight.deploy.doc/devref/t_transporting_the_app.html?lang=en

Testing on an Android device

Follow the instructions in this section to test the mobile applications on an Android device. Make sure that the Android device is set up for development. For information about how to set up your Android device for development, see *Setting up a Device for Development* at <http://developer.android.com/tools/device.html>

1. Connect your device to your workstation using a USB cable.
2. In Eclipse, make sure that the device is recognized in the Devices view: Press **Ctrl+3** and search for **Devices** to add the panel.
3. Right-click your Worklight application and select **Run As** → **Build All and Deploy**.
4. Right-click the generated Android project (it should appear below the Worklight project) and select **Run As** → **Android Applications**.
5. An unsigned .apk file is generated. Install it in the device.
6. Watch LogCat for errors. Press **Ctrl+3** and search for **LogCat** to add the panel.

Software version reference

The software versions used in the samples of this IBM Redbooks publication are:

- ▶ IBM Worklight Consumer Edition v6.2 with interim fix 6.2.0.0-IF201410080322. Note that as of v6.2, IBM Worklight Developer Edition, IBM Worklight Consumer Edition, and IBM Worklight Enterprise Edition provide the same instance of Worklight Studio.
- ▶ IBM Business Process Manager Advanced v8.5.5.



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/SG248240>

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, SG248240.

Using the Web material

The additional Web material that accompanies this book includes the following files:

<i>File name</i>	<i>Description</i>
SG248240.zip	Compressed Code Samples

System requirements for downloading the Web material

The Web material requires the following system configuration:

Hard disk space:	100 MB minimum
Operating System:	Windows/Linux

Downloading and extracting the Web material

Create a subdirectory (folder) on your workstation, and extract the contents of the Web material .zip file into this folder. For information about the contents of the file and how to deploy the samples included, see Appendix A, “Samples included with this book” on page 307.

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.

- ▶ *IBM MobileFirst Strategy Software Approach*, SG24-8191
- ▶ *Business Process Management Deployment Guide Using IBM Business Process Manager V8.5*, SG24-8175
- ▶ *Securing Your Mobile Business with IBM Worklight*, SG24-8179
- ▶ *Extending SAP Solutions to the Mobile Enterprise with IBM MobileFirst Platform Foundation*, REDP-5136-00

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 Worklight Foundation V6.2.0 documentation
http://www-01.ibm.com/support/knowledgecenter/SSZH4A_6.2.0/wl_welcome.html
- ▶ IBM Business Process Manager V8.5.5 documentation
http://www-01.ibm.com/support/knowledgecenter/SSFPJS_8.5.5/com.ibm.wbpm.main.doc/kc-homepage-bpm.html
- ▶ Digital reinvention: Preparing for a very different tomorrow
http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=XB&infotype=PM&appname=GBSE_GB_SC_USEN&htmlfid=GBE03583USEN

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Extending IBM Business Process Manager to the Mobile Enterprise with IBM Worklight

(0.5" spine)

0.475" <-> 0.873"

250 <-> 459 pages



Extending IBM Business Process Manager to the Mobile Enterprise with IBM Worklight

Unleash the benefits of IBM Mobile Smarter Process

Accelerate mobile app development by example

Discover IBM BPM features to mobile-enable processes

In today's business in motion environments, workers expect to be connected to their critical business processes while on-the-go. It is imperative to deliver more meaningful user engagements by extending business processes to the mobile working environments.

This IBM Redbooks publication provides an overview of the market forces that push organizations to reinvent their process with Mobile in mind. It describes IBM Mobile Smarter Process and explains how the capabilities provided by the offering help organizations to mobile-enable their processes.

This book outlines an approach that organizations can use to identify where within the organization mobile technologies can offer the greatest benefits. It provides a high-level overview of the IBM Business Process Manager and IBM Worklight features that can be leveraged to mobile-enable processes and accelerate the adoption of mobile technologies, improving time-to-value. Key IBM Worklight and IBM Business Process Manager capabilities are showcased in the examples included in this book. The examples show how to integrate with IBM Bluemix as the platform to implement various supporting processes.

This publication discusses architectural patterns for exposing business processes to mobile environments. It includes an overview of the IBM MobileFirst reference architecture and deployment considerations.

Through use cases and usage scenarios, this book explains how to build and deliver a business process using IBM Business Process Manager and how to develop a mobile app that enables remote users to interact with the business process while on-the-go, using the IBM Worklight Platform. The target audience for this book consists of solution architects, developers, and technical consultants.

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

SG24-8240-00

ISBN 0738440329