



Mobile connectivity with IBM Integration Bus v9

Dave Gorman – IBM Integration Bus Performance Team Lead
IBM Hursley – gormand@uk.ibm.com

13th August 2013
13294



Agenda



- **Introduction to Worklight**
- Worklight Adapters
- Integration Bus Mobile Patterns
 - Mobile enablement for Microsoft .NET applications
 - Mobile Services
 - Push Notifications
 - Resource handler including security and caching
 - MessageSight
- Demo

Worklight Overview



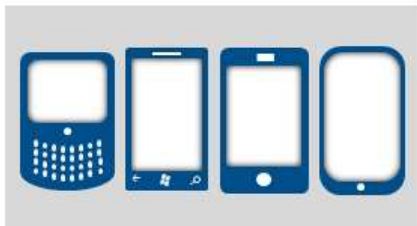
Worklight Studio

The most complete, extensible environment with maximum code reuse and per-device optimization



Worklight Server

Unified notifications, runtime skinning, version management, security, integration and delivery



Worklight Runtime Components

Extensive libraries and client APIs that expose and interface with native device functionality



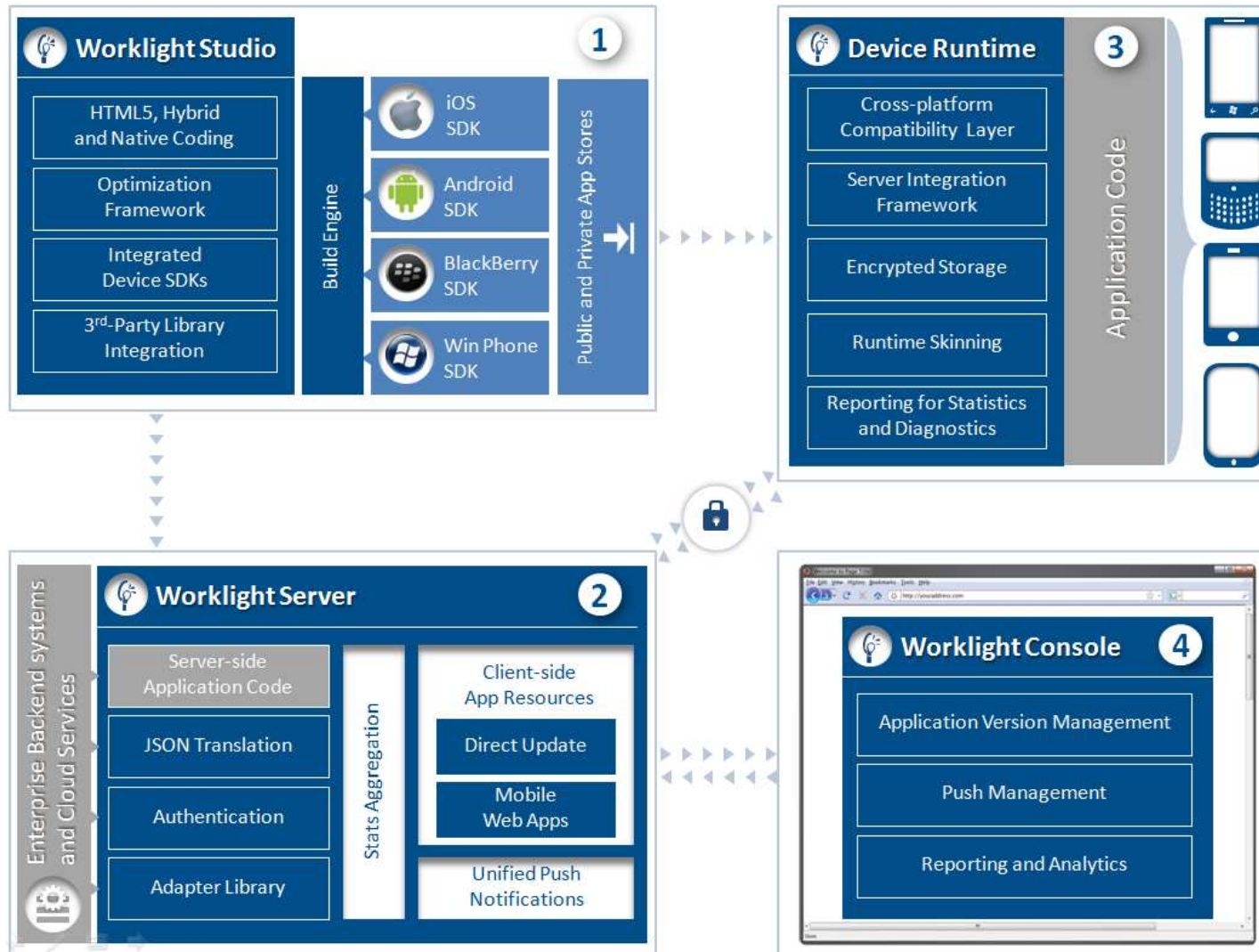
Worklight Console

A web-based console for real-time analytics and control of your mobile apps and infrastructure

Complete your sessions evaluation online at SHARE.org/BostonEval



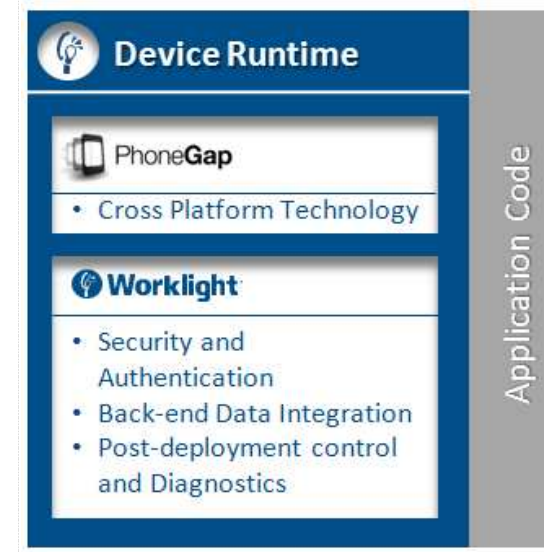
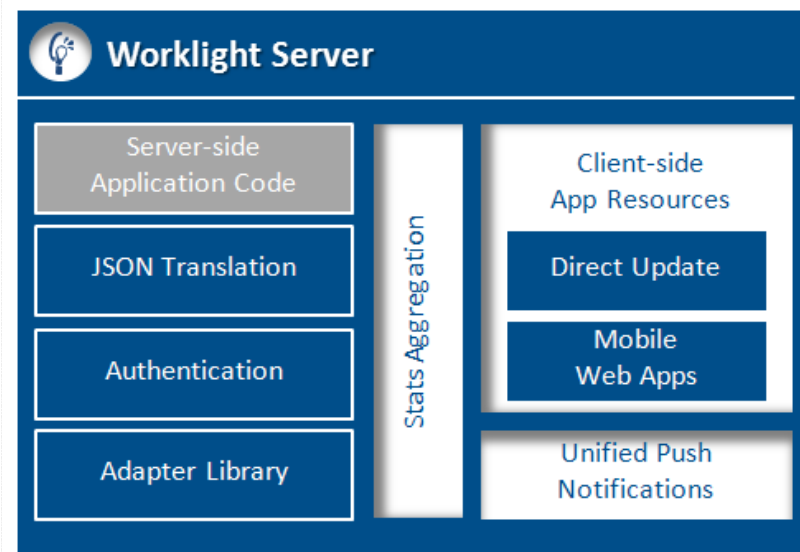
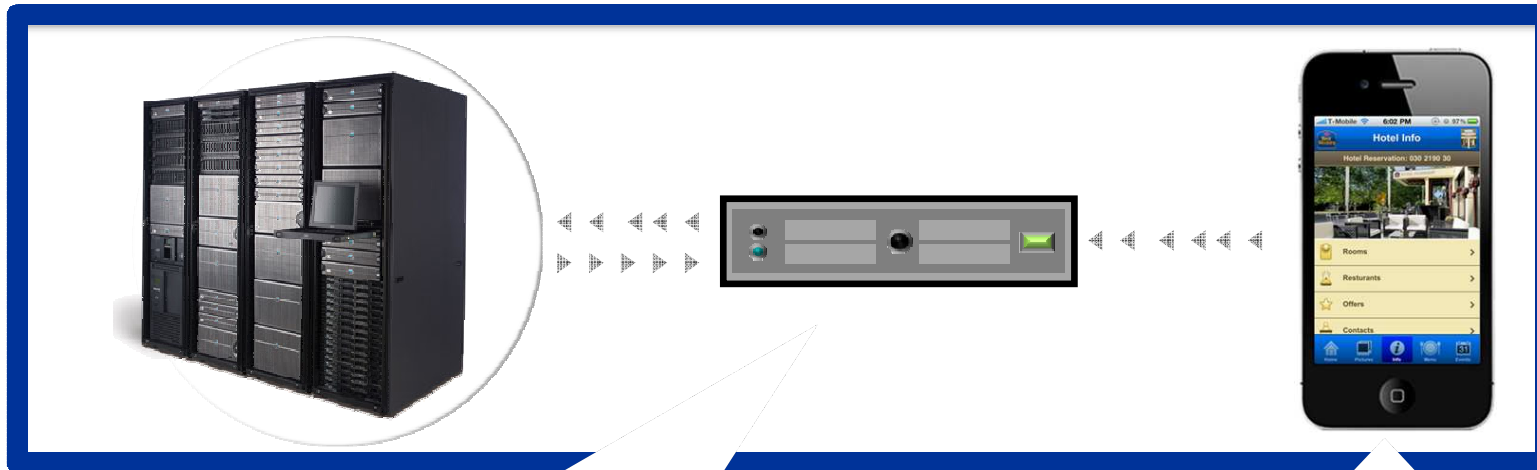
Worklight Architecture



Complete your sessions evaluation online at SHARE.org/BostonEval



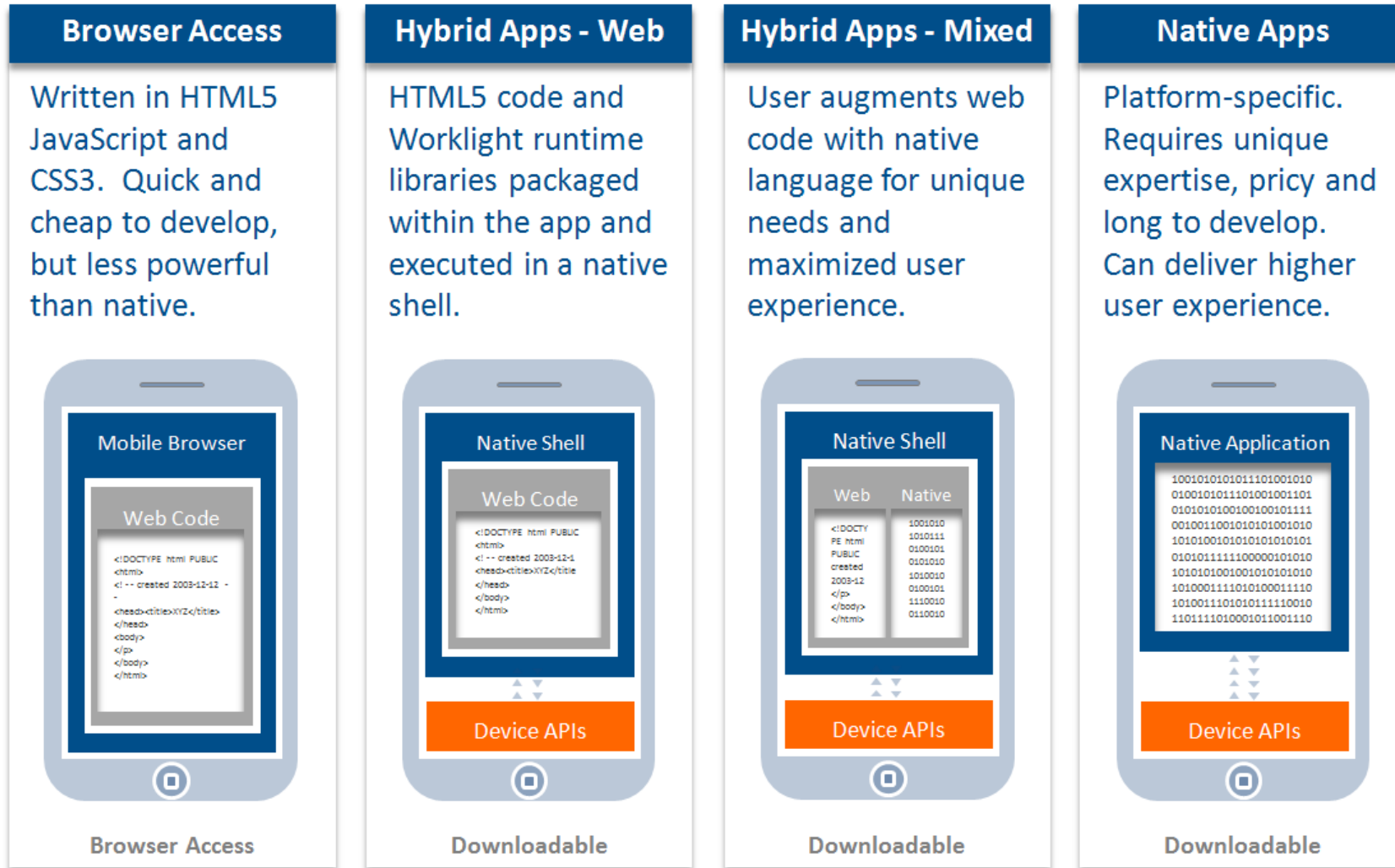
Worklight Overview



Complete your sessions evaluation online at SHARE.org/BostonEval



Types of Mobile Application



Agenda

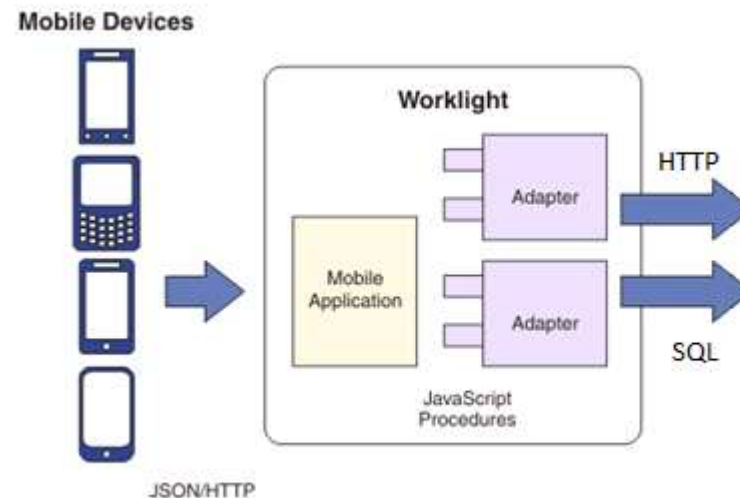


- Introduction to Worklight
- **Worklight Adapters**
- Integration Bus Mobile Patterns
 - Mobile enablement for Microsoft .NET applications
 - Mobile Services
 - Push Notifications
 - Resource handler including security and caching
 - MessageSight
- Demo

Worklight Adapters



- Adapters provide the glue between Worklight and back-end applications
 - Provides the extensibility mechanism for Worklight to call out to back-end systems
- Worklight has two built-in interfaces that adapters can use (HTTP and SQL)
 - Worklight has client-side JavaScript APIs so that applications can invoke services
 - Likewise, server-side JavaScript APIs are available to implement procedures (adapters)




Worklight Adapters



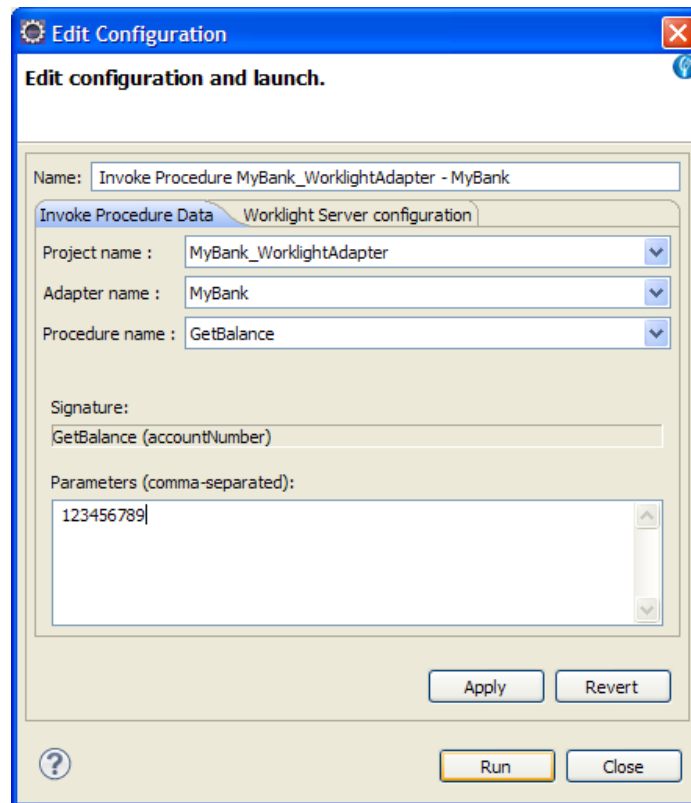
- An adapter contains two files for configuration and implementation
 - The first file is XML and contains the overall metadata (procedure names, protocol etc)
 - Second file is JavaScript and contains one function (procedure) for each entry point
- Adapters are uploaded to Worklight Server ready for mobile applications
 - Once deployed, adapters are managed through the Worklight Console

The screenshot shows the Worklight Console interface. At the top, there is a navigation bar with tabs for 'Catalog', 'Push Notifications', 'Reports', and 'Active Users'. Below this is a deployment area with a 'Deploy application or adapter:' label, a 'Choose File' button, the text 'No file chosen', and a 'Submit' button. The main content area displays the configuration for an adapter named 'MyBank'. On the right side of this section are 'Export' and 'Delete' buttons. The configuration details are as follows:

	Last updated at:	2012-06-20 14:34	
	Worklight integration adapter		
Connectivity:	Type:	HTTP	
	Protocol:	http	
	Domain:	localhost	
	Port:	7800	
	Use Proxy:	false	
Procedures:	GetBalance, TransferMoney, FindMissingAccount		
	Hide details ▲		

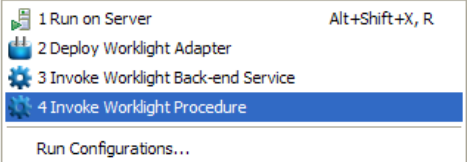
Invoking Worklight adapters

- Adapters are invoked from mobile applications using HTTP/JSON
 - This convention makes Worklight adapters easy to test using web browsers
 - Client side applications use the XMLHttpRequest object for asynchronous calls
 - Mobile toolkits (jQuery, Dojo and Sencha) wrap this in a device independent layer



Invocation Result of procedure: 'GetBalance' from the Worklight Server:

```
{
  "errors": [],
  "info": [],
  "isSuccessful": true,
  "result": {
    "NS1": "urn://bankingapplication/retailbank_V1",
    "lastUpdated": "20/06/2012 15:08:19",
    "returnValue": "1000.00"
  },
  "statusCode": 200,
  "statusReason": "OK",
  "warnings": []
}
```



Run Configurations dialog box showing a list of configurations:

- 1 Run on Server (Alt+Shift+X, R)
- 2 Deploy Worklight Adapter
- 3 Invoke Worklight Back-end Service
- 4 Invoke Worklight Procedure (highlighted)

Run Configurations...

Agenda

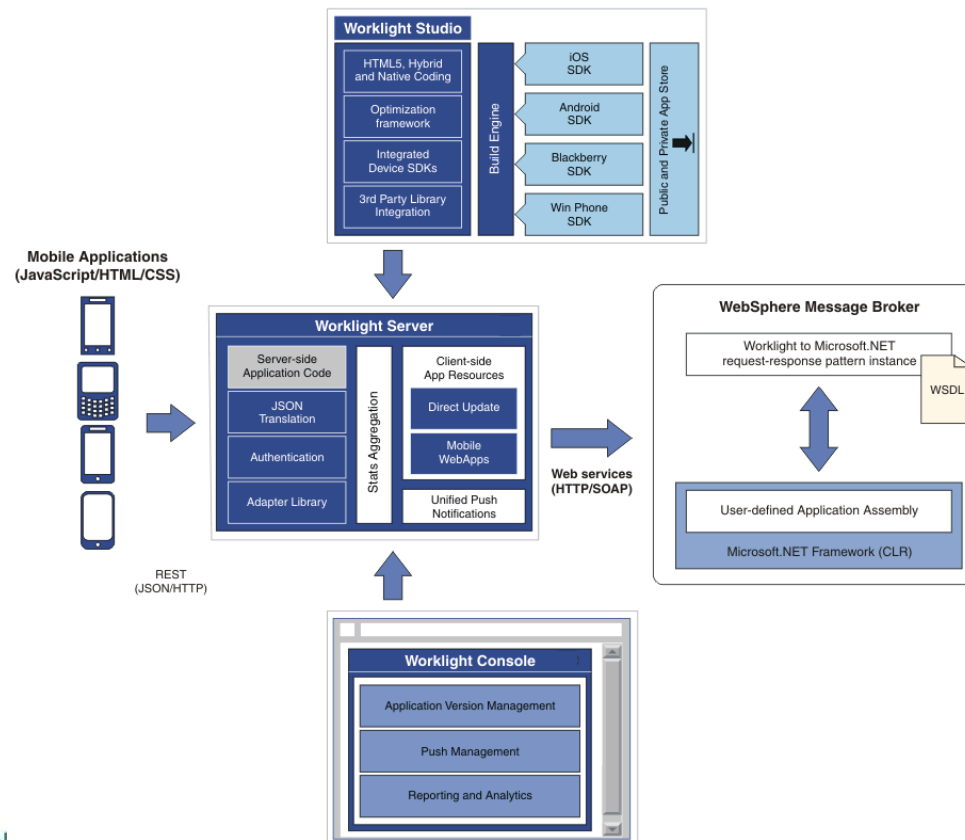


- Introduction to Worklight
- Worklight Adapters
- **Integration Bus Mobile Patterns**
 - **Mobile enablement for Microsoft .NET applications**
 - Mobile Services
 - Push Notifications
 - Resource handler including security and caching
 - MessageSight
- Demo

Worklight to Microsoft .NET Service Enablement



- Creates a mobile-ready service around a Microsoft .NET application
 - Generates a web service implementation which is deployed to Integration Bus
 - Builds a Worklight integration adapter and a sample mobile application
 - Inbound data from the mobile application is sent to Worklight as JSON/HTTP
 - The adapter converts the JSON data into/from SOAP/HTTP for the .NET web service



Complete your sessions eval



Configuring the Pattern Instance



- Pattern is configured with Microsoft .NET and Worklight information
 - Server address is a key field as it is used to configure both ends of the connection!
 - Standard set of error handling and logging options are provided by the pattern
 - Adapter configured with the maximum number of concurrent (HTTP) connections
 - Once this limit is reached, Worklight will queue inbound requests from applications

The screenshot displays a configuration interface for a Pattern Instance, organized into several sections:

- Worklight:** Configure the Worklight integration adapter. Fields include Adapter description (Worklight integration adapter), Maximum concurrent connections * (99), and Enable audit * (checked).
- Microsoft .NET assembly:** Configure the .NET assembly that implements the service calls. The Class name is BankinoApplication.RetailBank. Below this are expandable sections for Logging, Error handling, and General, each with a checkmark.
- Service information:** Service configuration information. Fields include Major version * (1), Minor version * (0), Enterprise domain * (BankingApplication), Service domain (empty), Service name * (RetailBank), Enable support for query WSDL * (checked), and Server address * (http://localhost:7800).

Complete your sessions evaluation online at SHARE.org/BostonEval



Configuring the Microsoft .NET Assembly



- User-defined editor allows the pattern user to select their .NET assembly
 - Selection proceeds to a class and the (static) methods available in that class
 - Assembly can be developed in any .NET language (for example, VB.NET or C#)
 - Return value and parameters are reflected on and displayed by the user-defined editor

Configure Microsoft .NET Service

Configure Microsoft .NET service
Configure your .NET assembly that the service invokes.

Assembly file name:

Assembly Information

Class name:

Methods on the class that the service will invoke:

Method Name	Abstract	Static	Public	Private	Return Type	Nullable	Web Method
<input checked="" type="checkbox"/> GetBalance	No	Yes	Yes	No	System.String	No	No
<input checked="" type="checkbox"/> TransferMoney	No	Yes	Yes	No	System.String	No	No
<input checked="" type="checkbox"/> FindMissingAccount	No	Yes	Yes	No	System.String	No	No
<input type="checkbox"/> ToString	No	No	Yes	No	System.String	No	No
<input type="checkbox"/> Equals	No	No	Yes	No	System.Boolean	No	No
<input type="checkbox"/> GetHashCode	No	No	Yes	No	System.Int32	No	No
<input type="checkbox"/> GetType	No	No	Yes	No	System.Type	No	No

Select All Clear All

Parameters:

Parameter Name	Type	Input	Output	Reference	Optional	Nullable
accountNumber	System.String	Yes	No	No	No	No
lastUpdated	System.String	No	Yes	No	No	No

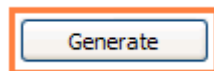
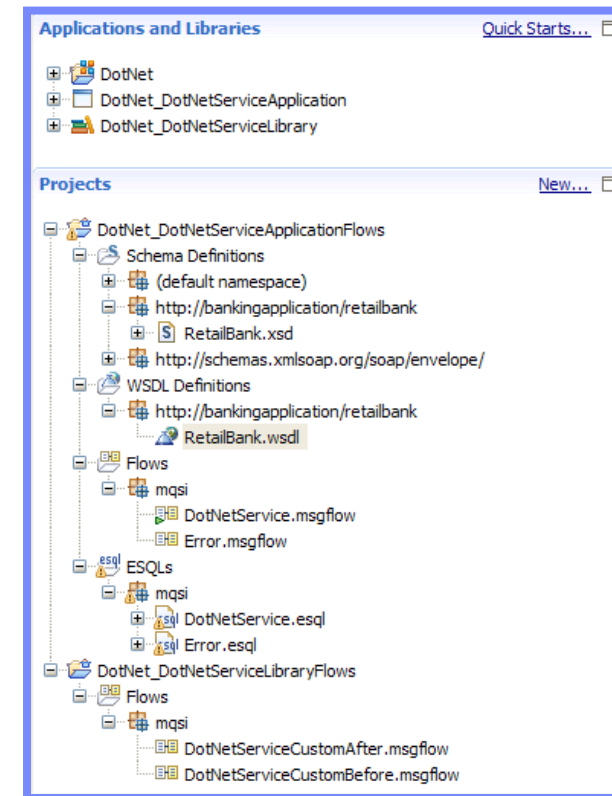
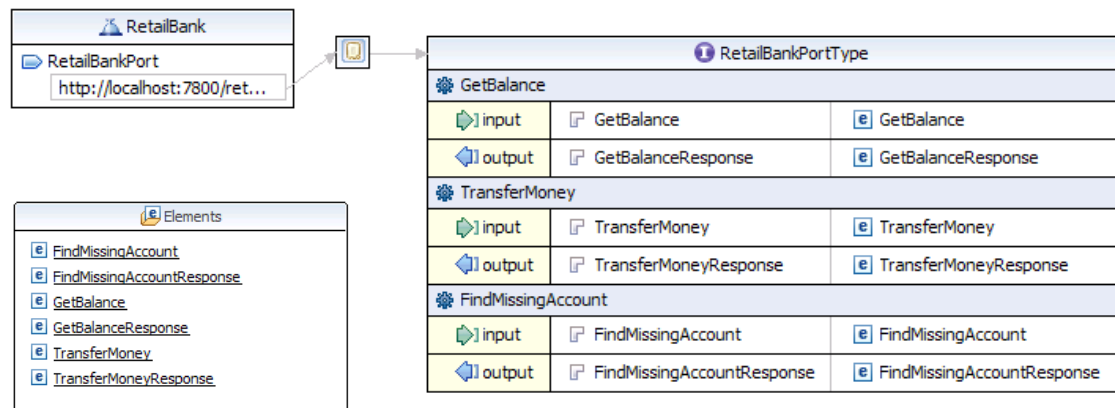
OK Cancel



Generated Integration Bus Projects



- The pattern generates an application and a library
 - Application contains the mechanics of the pattern instance
 - Library contains subflows for user customizations
 - Customizations are never deleted on re-generation!
- WSDL represents the selected .NET methods
 - One WSDL operation for each .NET (static) method
 - Likewise one message part defined per operation
 - WSDL types are defined in a separate XML schema file
 - WSDL and XSD are deployed directly to Integration Bus



Worklight Adapter



- Worklight adapter generated which reflects the web service methods
 - Integrates the mobile application with Integration Bus .NET web service
 - One procedure is generated for each operation (method) on the web service
 - Adapter manages the conversion between JSON and SOAP/XML data formats
 - Adapter generated in a separate project so it can be deployed to Worklight Server

The screenshot shows the Worklight management console interface. At the top, there is a navigation bar with 'Catalog', 'Push Notifications', 'Reports', and 'Active Users'. Below this is a deployment bar with a 'Choose File' button, 'No file chosen', and a 'Submit' button. The main content area displays a table with one entry for 'MyBank'. The entry includes a 'Last updated at' timestamp of '2012-06-20 14:34', a 'Worklight integration adapter' icon, and a table of connectivity details. The connectivity details table lists: Type: HTTP, Protocol: http, Domain: localhost, Port: 7800, and Use Proxy: false. Below the connectivity table, the 'Procedures' section lists 'GetBalance, TransferMoney, FindMissingAccount'. A 'Hide details' link with an upward arrow is at the bottom of the entry.

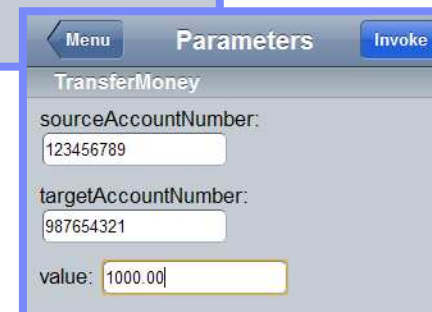
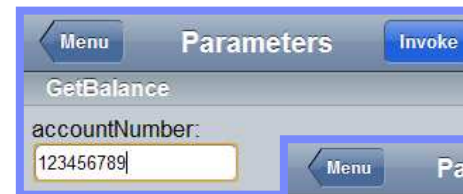
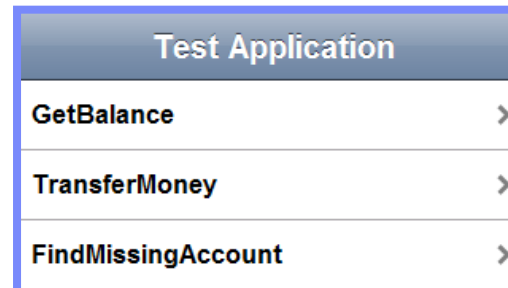
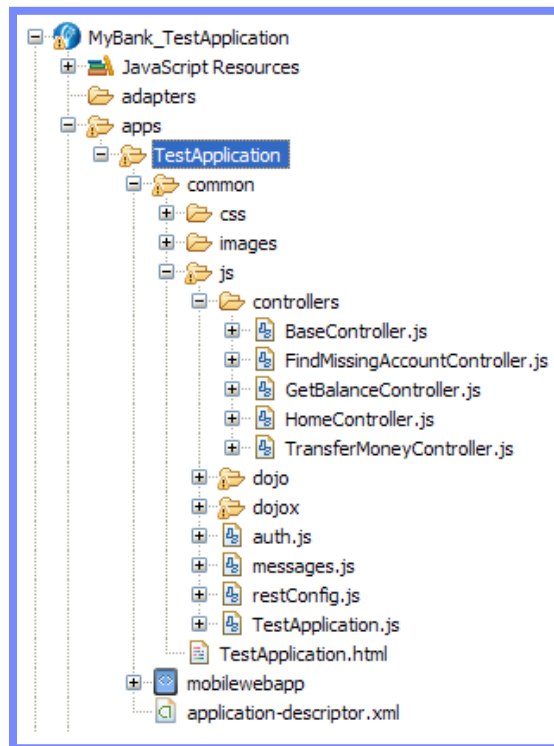
Connectivity:	
Type:	HTTP
Protocol:	http
Domain:	localhost
Port:	7800
Use Proxy:	false

Procedures: [GetBalance, TransferMoney, FindMissingAccount](#)

[Hide details](#) ▲

Mobile Application

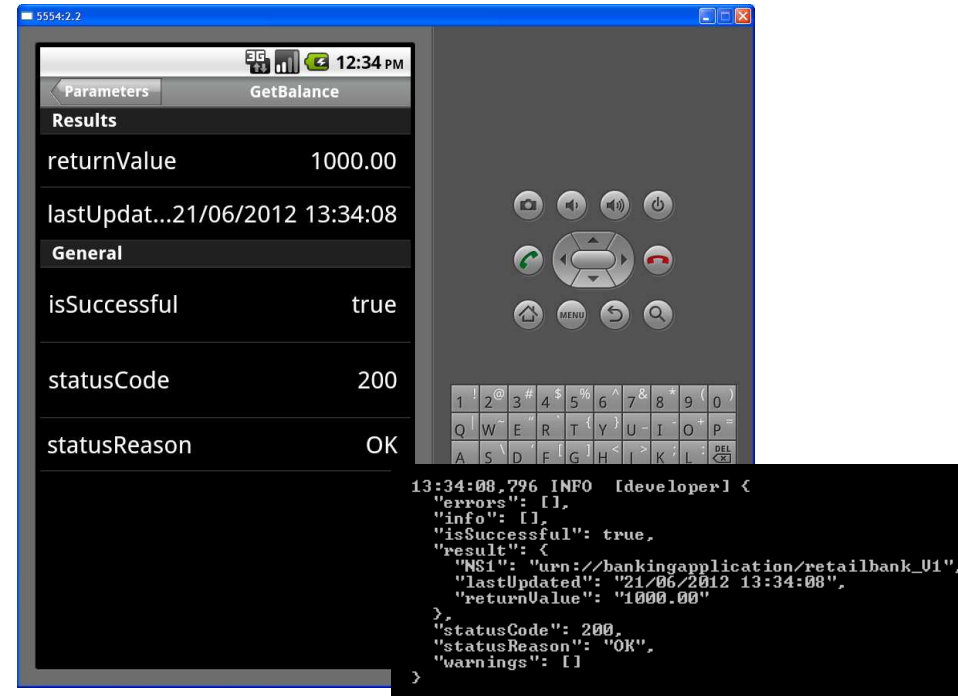
- Pattern also creates a mobile application to test the Worklight adapter
 - Each operation has views (pages) to configure and invoke the back-end service
 - Application is built using Dojo Mobile (ensures it is device independent)
 - More information on the Dojo mobile toolkit here: <http://dojotoolkit.org/features/mobile>



Parameters		GetBalance
Results		
returnValue		1000.00
lastUpdated		21/06/2012 13:26:53
General		
isSuccessful		true
statusCode		200
statusReason		OK

Mobile Application

- The mobile application has a single mobile web environment
 - Application is best suited for browsers on small screen mobile devices
 - Easy to add extra environments for iOS, Android and many more!
- Android development requires a separate download (Android SDK)
 - Pick and choose your target Android versions from Android SDK Manager



Agenda

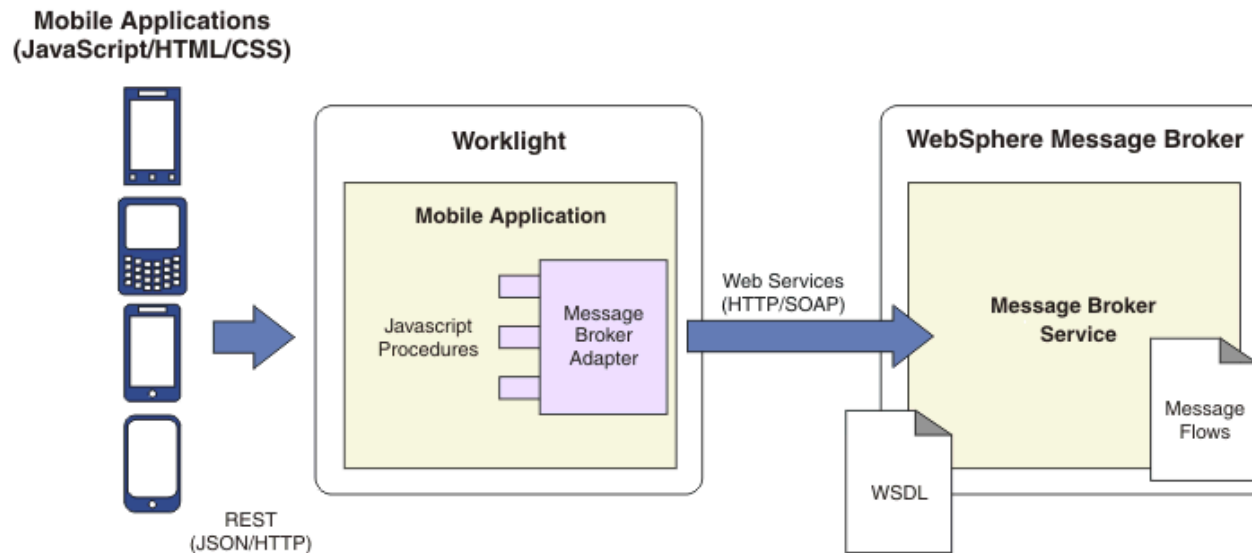


- Introduction to Worklight
- Worklight Adapters
- **Integration Bus Mobile Patterns**
 - Mobile enablement for Microsoft .NET applications
 - **Mobile Services**
 - Push Notifications
 - Resource handler including security and caching
 - MessageSight
- Demo

Worklight Mobile Services



- Creates a mobile-ready interface around an Integration Bus service
 - Services are a first class artifact in Integration Bus alongside applications and libraries
 - Builds an adapter to integrate Worklight and Integration Bus services
 - Inbound data from the mobile application is sent to Worklight as JSON/HTTP
- Makes it very simple to mobile enable an Integration Bus service!
 - The adapter passes the inbound request straight through to the service
 - Pattern adds an HTTP/JSON message flow (binding) to the service project



Configuring the Pattern Instance



- Create an Integration Bus service and then instantiate the pattern
 - You choose which operations in the service are available to mobile applications
 - Standard set of Worklight pattern parameters provided to configure the adapter

Worklight Mobile Service

Configure Worklight service

Select the operations that will be configured for your Worklight service

Service Information

Service name: Refresh

Select the web service operations that the Worklight service will invoke:

Operation Name	Input Type Name	Output type Name	One-way?
<input checked="" type="checkbox"/> SaveAddress	Person	SaveAddressResponse	No
<input checked="" type="checkbox"/> FindAddress	Name		

Worklight Configuration

Configure the Worklight integration adapter

Worklight version:

Adapter description:

Maximum concurrent connections *:

Enable audit *:

Interface

Configuration

Name: AddressBook

Namespace: http://addressbook.com/

Operations

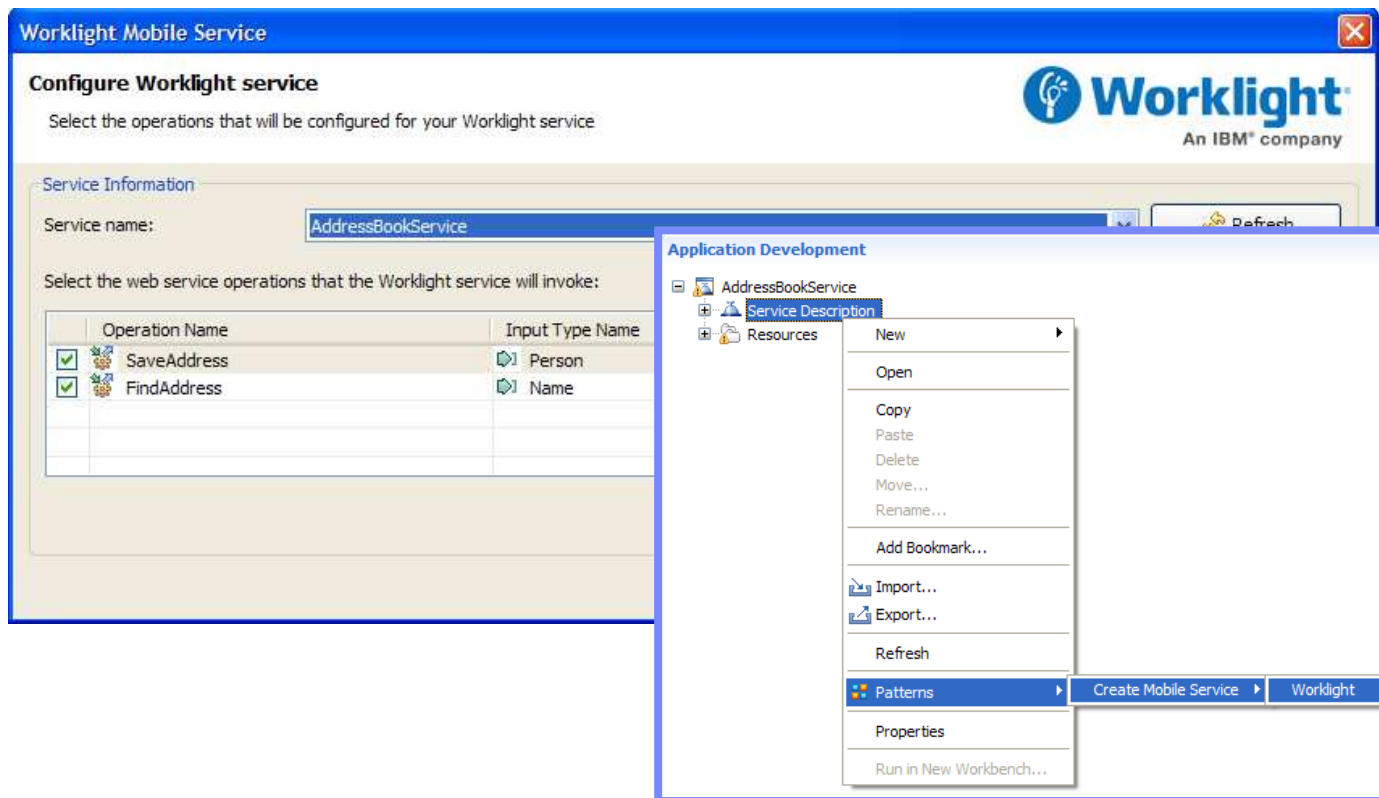
Operations and their parameters

Message Type	Name	Type
SaveAddress		
Person	Person	PersonType
SaveAddressResponse	SaveAddressResponse	boolean
FindAddress		
Name	Name	string
Address	Address	AddressType
FindAddressFault	FindAddressFault	FindAddressFaultType



Configuring the Pattern Instance

- The mobile service pattern can also be launched from the Navigator
 - Intuitive user experience for mobile enablement of Integration Bus services
 - The selected service name is passed to the pattern as the launch configuration
 - Pattern instance is configured automatically and can be immediately generated



The screenshot displays the 'Worklight Mobile Service' configuration window. The main window is titled 'Configure Worklight service' and contains a 'Service Information' section with a 'Service name' field set to 'AddressBookService'. Below this is a table for selecting web service operations:

	Operation Name	Input Type Name
<input checked="" type="checkbox"/>	SaveAddress	Person
<input checked="" type="checkbox"/>	FindAddress	Name

An 'Application Development' navigator window is overlaid on the main window, showing a tree view with 'AddressBookService' selected. A context menu is open over 'AddressBookService', with the 'Patterns' option selected. The 'Patterns' submenu is visible, showing 'Create Mobile Service' and 'Worklight' options.

Worklight Adapter




- Generates a Worklight adapter which reflects the web service methods
 - Integrates the mobile application with the Integration Bus web service
 - One procedure is generated for each selected operation in the service
 - Request-response and one-way interactions for the service are supported 🍷 🍷

IBM Worklight Console

Catalog | Push Notifications | Active Users

Deploy application or adapter: No file chosen

Address Service

 Last updated at: 2012-07-10 16:49
Message Broker service adapter

Connectivity:	Type:	HTTP
	Protocol:	http
	Domain:	localhost
	Port:	7800
	Use Proxy:	false

Procedures: SaveAddress, FindAddress

Hide details ▲

Test Application!

Operations

- SaveAddress >
- FindAddress >

Agenda



- Introduction to Worklight
- Worklight Adapters
- **Integration Bus Mobile Patterns**
 - Mobile enablement for Microsoft .NET applications
 - Mobile Services
 - **Push Notifications**
 - Resource handler including security and caching
 - MessageSight
- Demo

Worklight Push Notification Services



- Worklight supports asynchronous push notifications to mobile applications
 - Push notifications have a measurable impact on the success of mobile applications
 - There are many IT challenges in supporting push notifications (devices, delivery etc)
- Push notifications are applicable across many industry verticals
 - Healthcare, retail, travel, transportation, government, insurance and more!
- All the major mobile platforms support push notification services
 - Apple iOS 3, Google Android 2.2, RIM Blackberry 5 and Windows Phone 7



Complete your sessions evaluation online at SHARE.org/BostonEval



Worklight Push Notification Services



- Users receive notifications when the mobile application is not active
 - Efficiency gain as application does not need to issue constant queries
 - Saves battery life and also reduces network bandwidth (communication fees)
- Notifications are not always appropriate and have disadvantages
 - Users need to subscribe on their device to receive push notifications
 - Notifications are limited in the size of their payload (for example, 256 bytes on iOS)
 - No quality of service is guaranteed and there is no delivery notification
 - No guarantee either that the end-to-end delivery chain is secure

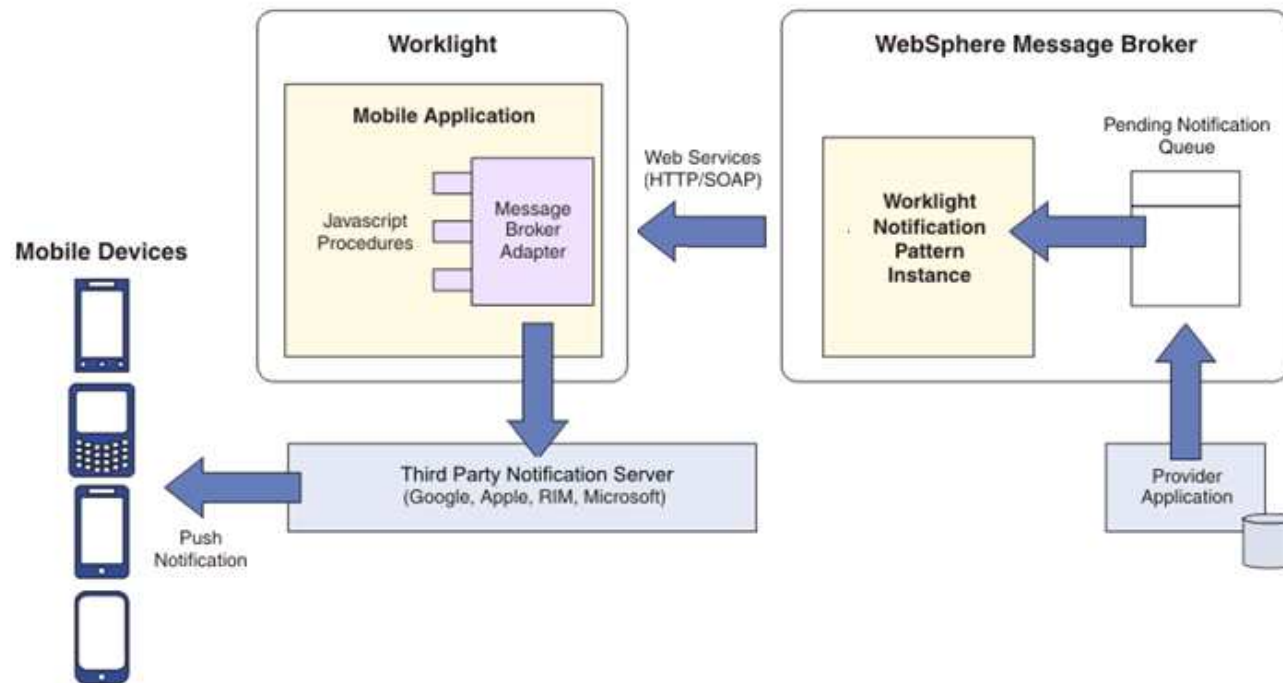


Complete your sessions evaluation online at SHARE.org/BostonEval



Worklight Push Notification from WebSphere MQ

- Creates a push notification adapter from a WebSphere MQ queue
 - Generates a web service implementation which is deployed to Integration Bus
 - Builds a Worklight integration adapter which polls for pending notifications
 - Pending notifications are written to a WebSphere MQ queue by a provider application
 - The adapter converts the notifications into JSON and arranges delivery to the mobile



Configuring the Pattern Instance



- Pattern is configured with Worklight and Integration Bus information
 - Server address is a key field as it is used to configure both ends of the connection!
 - Standard set of error handling and logging options are provided by the pattern
- Application specific fields can be delivered in the push notification
 - Configured as part of the pattern instance so that an accurate schema can be created

The screenshot displays the configuration interface for a Worklight push notification pattern instance. It is divided into several sections:

- Worklight** (checked):
 - Worklight version: Worklight v5.0
 - Adapter description: Worklight push notification adapter
 - Event source: HealthcareAppointments
 - Payload: A table with columns for Name and values for TimeOfAppointment and PhysicianName. Buttons for Add..., Edit..., and Delete are visible.
 - Polling interval *: 30
- Logging** (expanded):
- Error handling** (expanded):
- General** (checked):
 - Service information** (checked):
 - Service configuration information
 - Service name: notifications
 - Enable support for query WSDL *:
 - Notification queue name *: NTFY
 - Server address *: http://localhost:7800

Worklight Adapter



- Worklight adapter generated which periodically checks for notifications
 - Integrates Worklight with a queue of notifications managed by Integration Bus
 - Generated pattern instance project includes a schema for the notification messages
 - Adapter manages the conversion from XML to JSON for the Worklight server-side calls
- Polling interval for pending notifications is configurable in the pattern
 - Adapter greedily processes all pending notifications each time it wakes up

```
<xsd:complexType name="Payload">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">Application specific data in the notification messages.</xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element minOccurs="0" name="TimeOfAppointment" type="xsd:string"/>
    <xsd:element minOccurs="0" name="PhysicianName" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="Notification">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">Response message for notification messages.</xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="UserId" type="xsd:string"/>
      <xsd:element minOccurs="0" name="Badge" type="xsd:string"/>
      <xsd:element minOccurs="0" name="Sound" type="xsd:string"/>
      <xsd:element minOccurs="0" name="ActivateButtonLabel" type="xsd:string"/>
      <xsd:element minOccurs="0" name="NotificationText" type="xsd:string"/>
      <xsd:element name="Payload" type="tns:Payload"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

NotificationPortType		
GetNotification		
input	GetNotification	GetNotification
output	GetNotificationResponse	GetNotificationResponse
PutNotification		
input	PutNotification	PutNotification
output	PutNotificationResponse	PutNotificationResponse

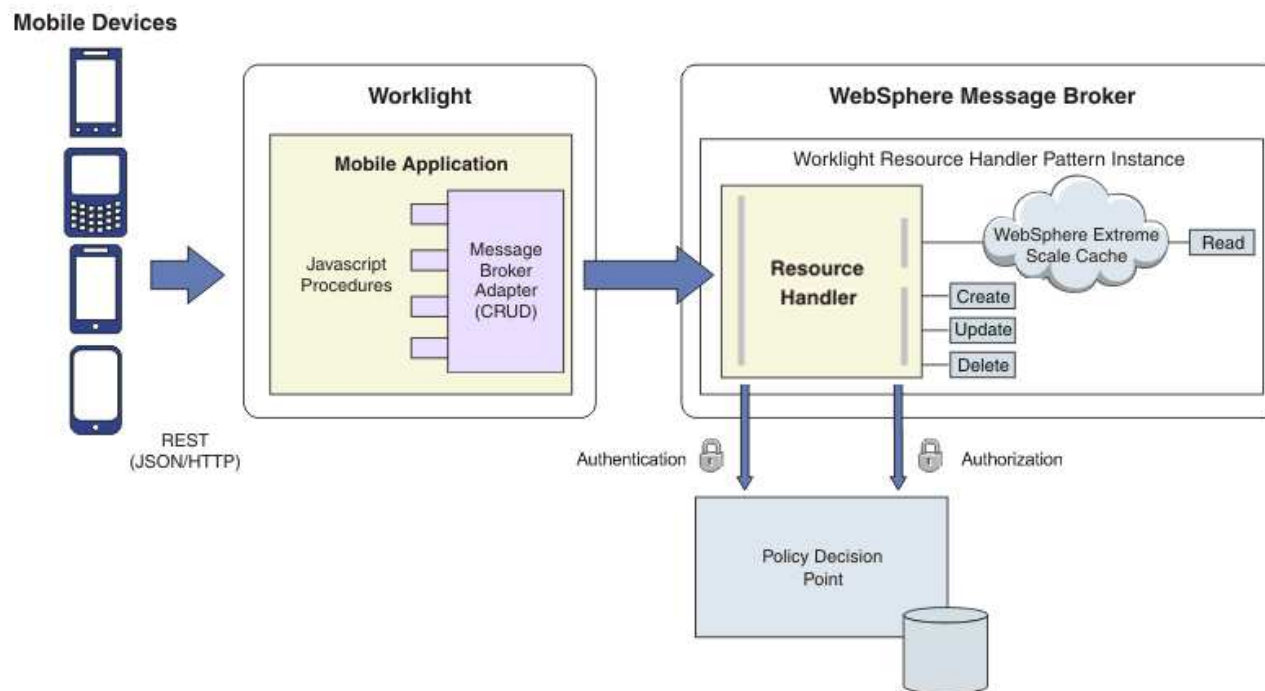
Agenda



- Introduction to Worklight
- Worklight Adapters
- **Integration Bus Mobile Patterns**
 - Mobile enablement for Microsoft .NET applications
 - Mobile Services
 - Push Notifications
 - **Resource handler including security and caching**
 - MessageSight
- Demo

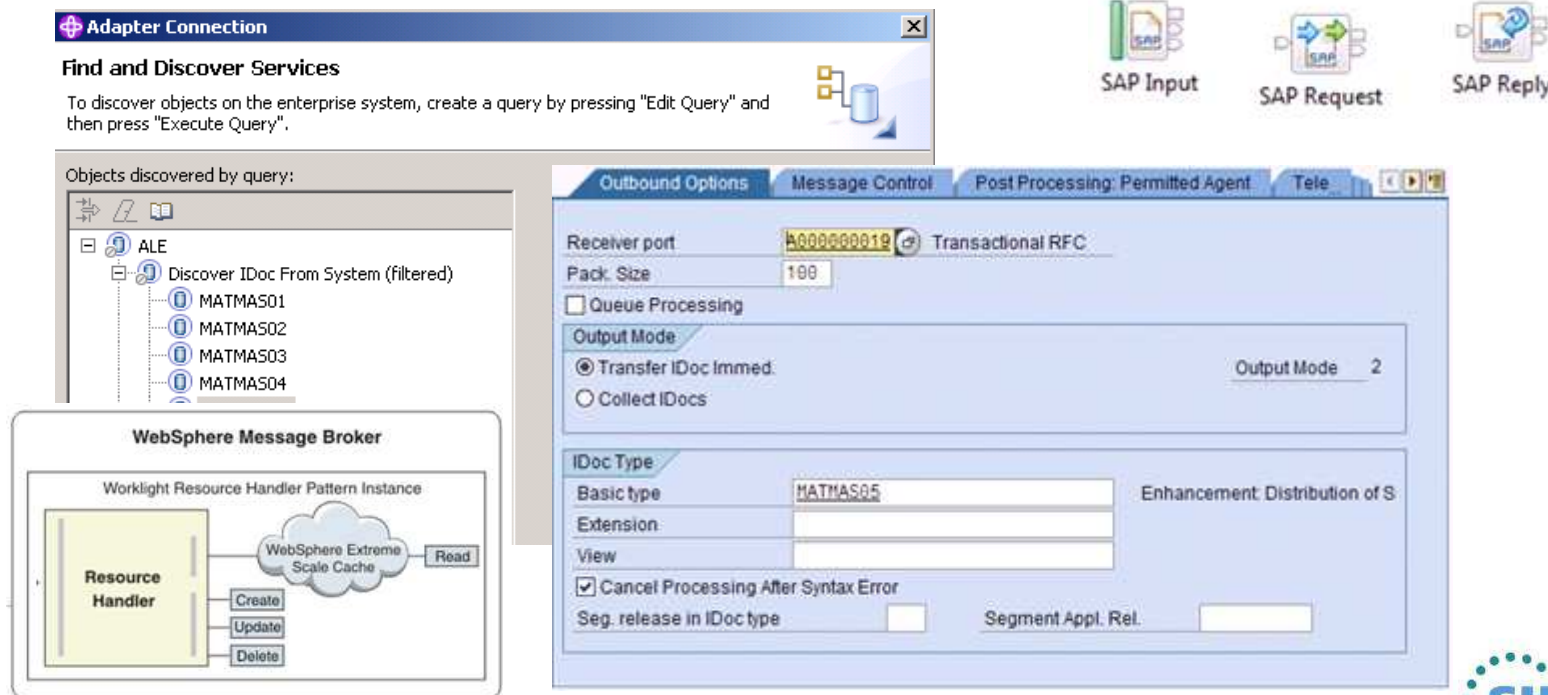
Worklight Resource Handler

- Resource oriented architecture is a well known implementation pattern
 - Provides a common set of functions (CRUD – Create Read Update and Delete)
 - This pattern provides an adapter which implements CRUD operations
 - An Integration Bus service is generated with subflows for each operation
 - The service integrates security authorization and authentication (LDAP)
 - Operations optionally integrate with the Integration Bus Global Cache (WXS)



Implementing Resource Handlers

- Complete the pattern instance by implementing the resource handlers
 - Subflows are generated for each CRUD operation in a customization project
 - Pattern generates a reference implementation of a back end system in ESQL
- Integration Bus has excellent support for enterprise applications
 - Common design pattern to integrate with SAP, Siebel, JDEdwards and PeopleSoft
 - Wizards makes it easy to discover the application content (for example, SAP iDocs)
 - Rich SAP support includes iDocs, ALE, BAPI and query SAP tables (QISS)



The image displays several screenshots related to SAP integration configuration:

- Adapter Connection - Find and Discover Services:** A window with instructions: "To discover objects on the enterprise system, create a query by pressing 'Edit Query' and then press 'Execute Query'". Below, a tree view shows "Objects discovered by query:" including ALE, Discover IDoc From System (filtered), and four MATMAS instances (MATMAS01 to MATMAS04).
- SAP Integration Icons:** Three icons labeled "SAP Input", "SAP Request", and "SAP Reply".
- WebSphere Message Broker Diagram:** A diagram titled "Worklight Resource Handler Pattern Instance" showing a "Resource Handler" box connected to a "WebSphere Extreme Scale Cache" cloud, which is linked to a "Read" operation. Below the cache are "Create", "Update", and "Delete" buttons.
- Outbound Options Configuration:** A configuration window for "Transactional RFC" with fields for "Receiver port" (A000000019), "Pack. Size" (100), and "Output Mode" (Transfer IDoc Immed. selected, Output Mode 2). The "IDoc Type" section shows "Basic type" as MATMAS05 and "Extension" as empty.

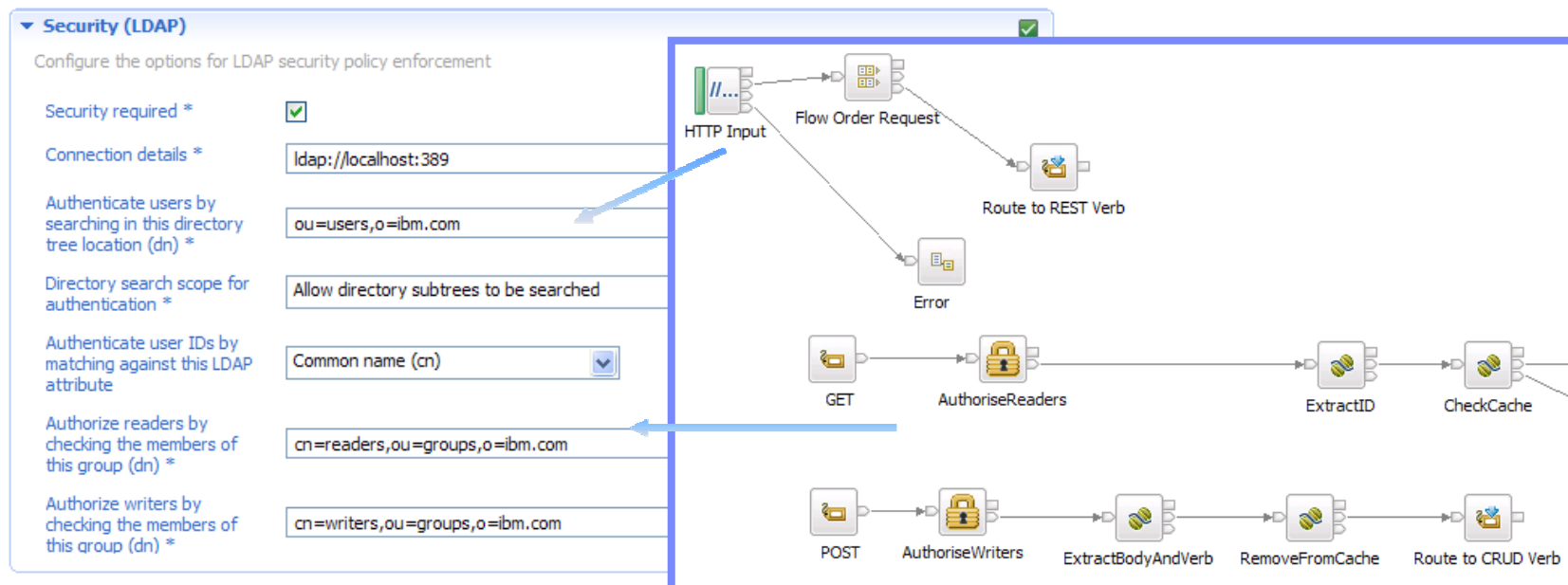
WebSphere Extreme Scale (WXS)

- WebSphere Extreme Scale is tightly integrated with Integration Bus
 - Provides a highly scalable, fault tolerant, elastic in-memory data grid
 - One or more execution groups manage a single logical cache of key-value data
 - WXS components are hosted within the execution group processes
 - Default scope is one cache per broker but this can be extended to multiple brokers
- Vital for mobile applications where the number of devices can be huge
 - Caching fits perfectly with a CRUD model of many readers and (generally) few writers
 - Integration Bus activity log shows the cache activity as CRUD operations complete

Message Nu...	Timestamp	MSGFLOW	Message Summary	ThreadID	CACHEKEY	CACHENAME
i BIP1150II	23-Jul-2012 21:55:01.000 BST	RESTProviderFlow	Received data from input node 'HTTPInput'.	4960		
i BIP11506I	23-Jul-2012 21:55:01.000 BST	RESTProviderFlow	Committed a local transaction.	4960		
i BIP1150II	23-Jul-2012 21:55:05.000 BST	RESTProviderFlow	Received data from input node 'HTTPInput'.	4960		
i BIP11103I	23-Jul-2012 21:55:19.000 BST	RESTProviderFlow	Got data from map 'rhWorklightCache'	4960	1	WMB
i BIP1110II	23-Jul-2012 21:55:19.000 BST	RESTProviderFlow	Put data into map 'rhWorklightCache'	4960	1	WMB
i BIP11506I	23-Jul-2012 21:55:19.000 BST	RESTProviderFlow	Committed a local transaction.	4960		
i BIP1150II	23-Jul-2012 21:55:35.000 BST	RESTProviderFlow	Received data from input node 'HTTPInput'.	4960		
i BIP11103I	23-Jul-2012 21:55:35.000 BST	RESTProviderFlow	Got data from map 'rhWorklightCache'	4960	1	WMB
i BIP11506I	23-Jul-2012 21:55:35.000 BST	RESTProviderFlow	Committed a local transaction.	4960		
i BIP1150II	23-Jul-2012 21:55:39.000 BST	RESTProviderFlow	Received data from input node 'HTTPInput'.	4960		
i BIP11103I	23-Jul-2012 21:55:39.000 BST	RESTProviderFlow	Got data from map 'rhWorklightCache'	4960	1	WMB
i BIP11506I	23-Jul-2012 21:55:39.000 BST	RESTProviderFlow	Committed a local transaction.	4960		
i BIP1150II	23-Jul-2012 21:56:26.000 BST	RESTProviderFlow	Received data from input node 'HTTPInput'.	4960		
i BIP11103I	23-Jul-2012 21:56:26.000 BST	RESTProviderFlow	Got data from map 'rhWorklightCache'	4960	1	WMB
i BIP11506I	23-Jul-2012 21:56:26.000 BST	RESTProviderFlow	Committed a local transaction.	4960		

Authorization and Authentication

- Patterns provides a security model based around LDAP
 - Caching fits perfectly with a CRUD model of many readers and (generally) few writers
 - Users are authenticated using HTTP basic authentication by the HTTP Input node
 - Authorization is then done by splitting the users into two groups (readers/writers)
 - A user is authorized if they are a member of the group in the LDAP directory
 - The LDAP queries are issued by the message flow using the Security PEP node
 - Caching changes are made through WXS after the user has cleared security



Agenda

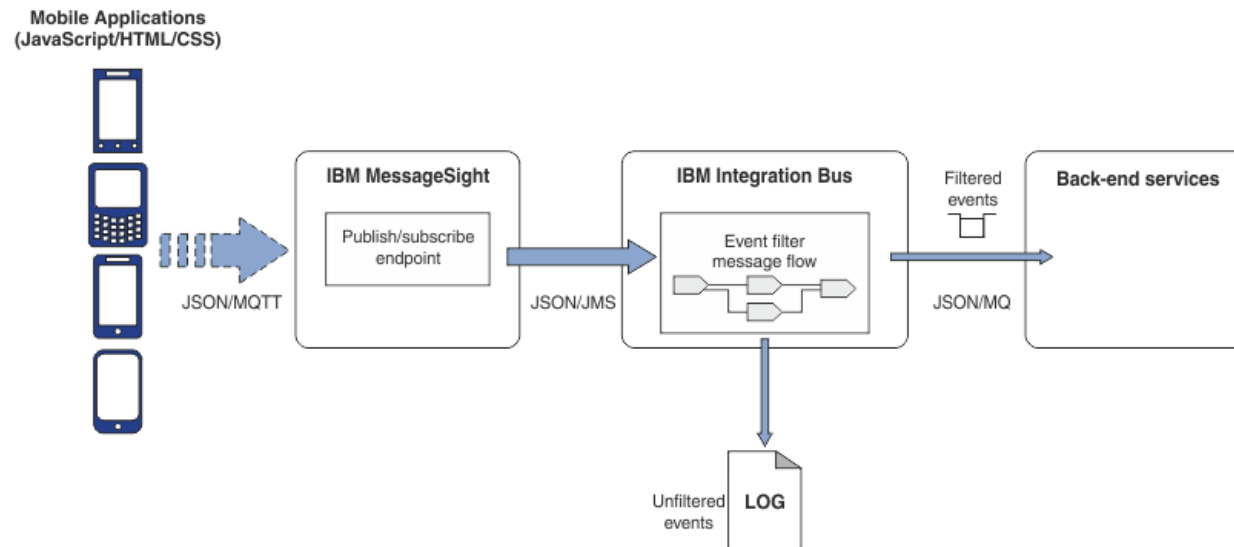


- Introduction to Worklight
- Worklight Adapters
- **Integration Bus Mobile Patterns**
 - Mobile enablement for Microsoft .NET applications
 - Mobile Services
 - Push Notifications
 - Resource handler including security and caching
 - **MessageSight**
- Demo

MessageSight : Event filter



- Use in scenarios where mobile and pervasive devices publish events
 - Forwards relevant events to back-end and discards or logs others.
 - Protects backend application and services from high volume of event.



Configuring event filter



▼ IBM MessageSight Endpoint Configuration

Pattern Parameters

Server *

Port *

Single port for JMS/MQTT/Console UI

▼ Event Filtering

Configure the events to subscribe to and the xpath used to filter them

Topic *

Filter * XPath

Topic can include wildcards

▼ Filtered Event Output

Configure the queue manager and queue used to receive filtered events

Queue manager name

Queue name *



Demo client for event filtering



The screenshot shows a web browser window titled "Pacemaker Event Publisher" with the URL "PacemakerClient.html". The page has a dark background with red text and a red heart icon with an ECG line. It is divided into three main sections: "Connection to event gateway endpoint", "Publish to event gateway", and "Log".

Connection to event gateway endpoint: Server: Port:

Publish to event gateway: Topic: Event Type:

Message:

```
{
  "device": {"id": "Pacemaker45237", "type": "pacemaker"},
  "date": "6/24/2013",
  "time": "11:10:45 AM",
  "event": {"type": "information", "detail": "Functioning normally"}
}
```

Log:

Connect to message sight as a MQTT client

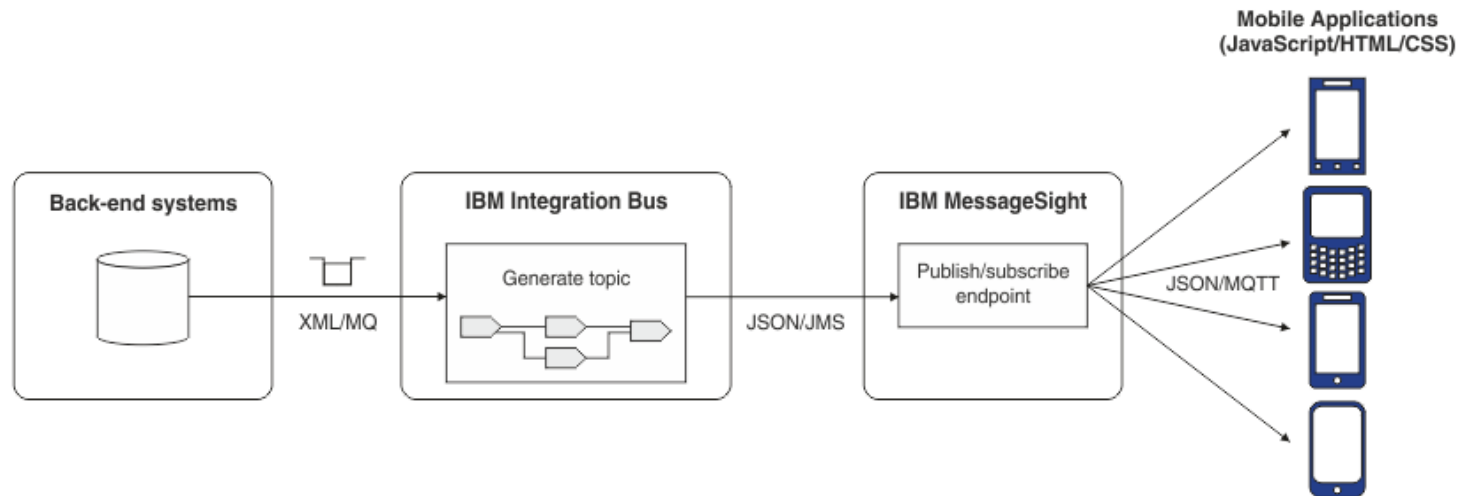
Publish JSON message to specific topic

View log



MessageSight : Event notification

- Use to notify devices of events from back-end systems or applications
 - Transform XML to JSON for ease of consumption
 - Unbounded set of topic names
 - Topic name dynamically generated based on message content



Configuring event notification



▼ **IBM MessageSight Endpoint Configuration** ✓

Pattern Parameters

Server *

Port *

▼ **Input Configuration** ✓

Pattern Parameters

Queue name

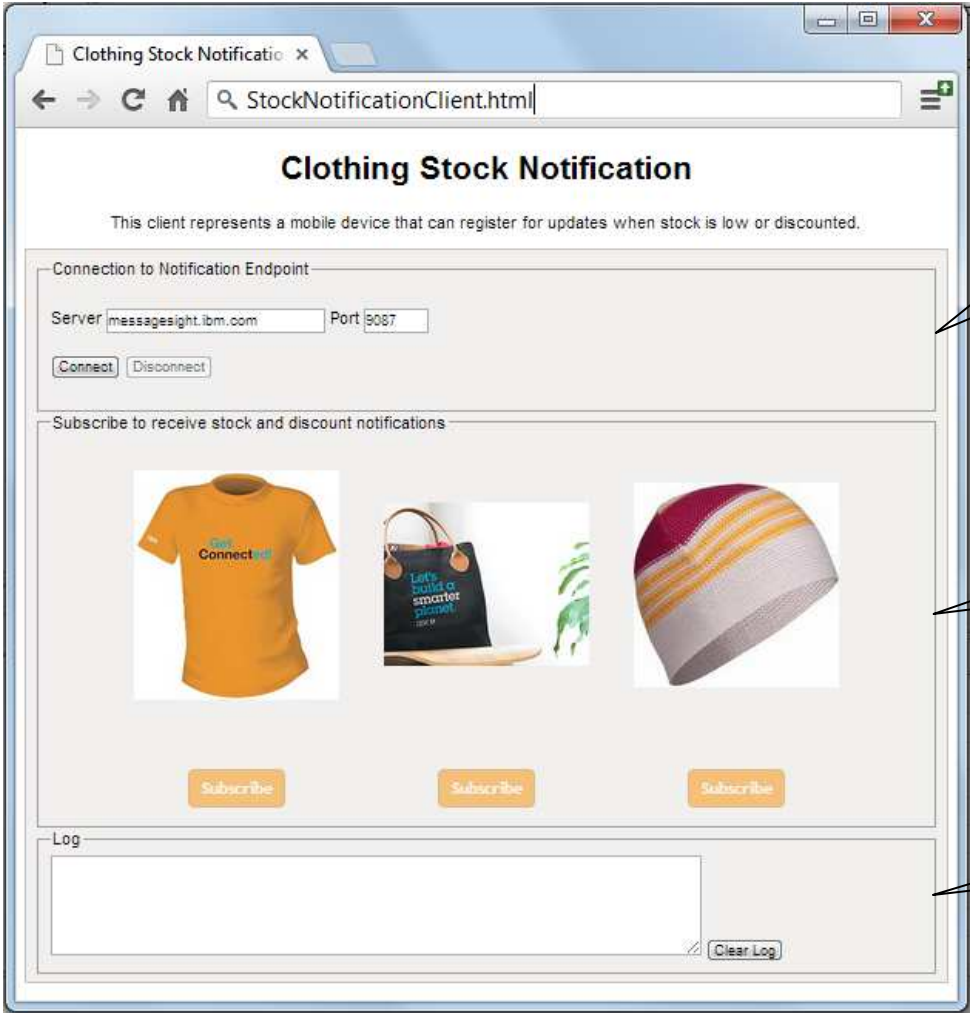
▼ **Event Notification Configuration** ✓

Pattern Parameters

Topic	XPath
	<input type="text" value="IBM/testTopic"/> <input type="button" value="Edit..."/>



Demo client for event notification



Connect to message sight as a MQTT client

Subscribe to topic(s)

View log




Agenda



- Introduction to Worklight
- Worklight Adapters
- Integration Bus Mobile Patterns
 - Mobile enablement for Microsoft .NET applications
 - Mobile Services
 - Push Notifications
 - Resource handler including security and caching
 - MessageSight
- **Demo**

This was session 13294 - The rest of the week



	Monday	Tuesday	Wednesday	Thursday	Friday
08:00				Extending IBM WebSphere MQ and WebSphere Integration Bus to the Cloud	CICS and WMQ - The Resurrection of Useful
09:30	Introduction to MQ				Can I Consolidate My Queue Managers and Brokers?
11:00		MQ on z/OS - Vivisection	Hands-on Lab for MQ - take your pick!	MOBILE connectivity with Integration Bus	Migration and Maintenance, the Necessary Evil. Into the Dark for MQ and Integration Bus
12:15					
01:30	MQ Parallel Sysplex Exploitation, Getting the Best Availability From MQ on z/OS by Using Shared Queues	What's New in the MQ Family	MQ Clustering - The basics, advances and what's new	Using IBM WebSphere Application Server and IBM WebSphere MQ Together	
03:00	First Steps With Integration Bus: Application Integration for the Messy	What's New in Integration Bus	BIG Connectivity with mobile MQ	WebSphere MQ CHINIT Internals	
04:30	What's available in MQ and Broker for high availability and disaster recovery?	The Dark Side of Monitoring MQ - SMF 115 and 116 Record Reading and Interpretation	MQ & DB2 – MQ Verbs in DB2 & Q-Replication performance	Big Data Sharing with the Cloud - WebSphere eXtreme Scale and IBM Integration Bus	
06:00				WebSphere MQ Channel Authentication Records	

Complete your sessions evaluation online at SHARE.org/BostonEval





SHARE
Technology · Connections · Results



Complete your sessions evaluation online at SHARE.org/BostonEval

