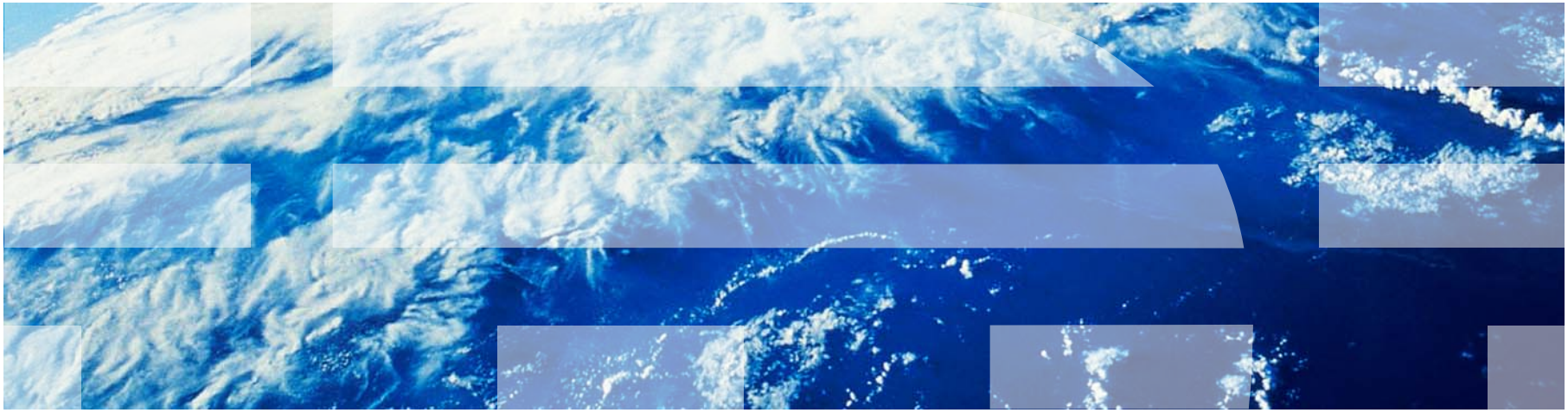
IBM Worklight V6.1.0
Getting Started

Developing dynamic, collaborative mobile applications with MQ Telemetry Transport



17 March 2014

Trademarks

- IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Worklight is a trademark or registered trademark of Worklight, an IBM Company. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Microsoft, Windows, Windows NT, 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 products or service names may be trademarks or service marks of others.
- This document may not be reproduced in whole or in part without the prior written permission of IBM.

About IBM®

- See http://www.ibm.com/ibm/us/en/

Agenda

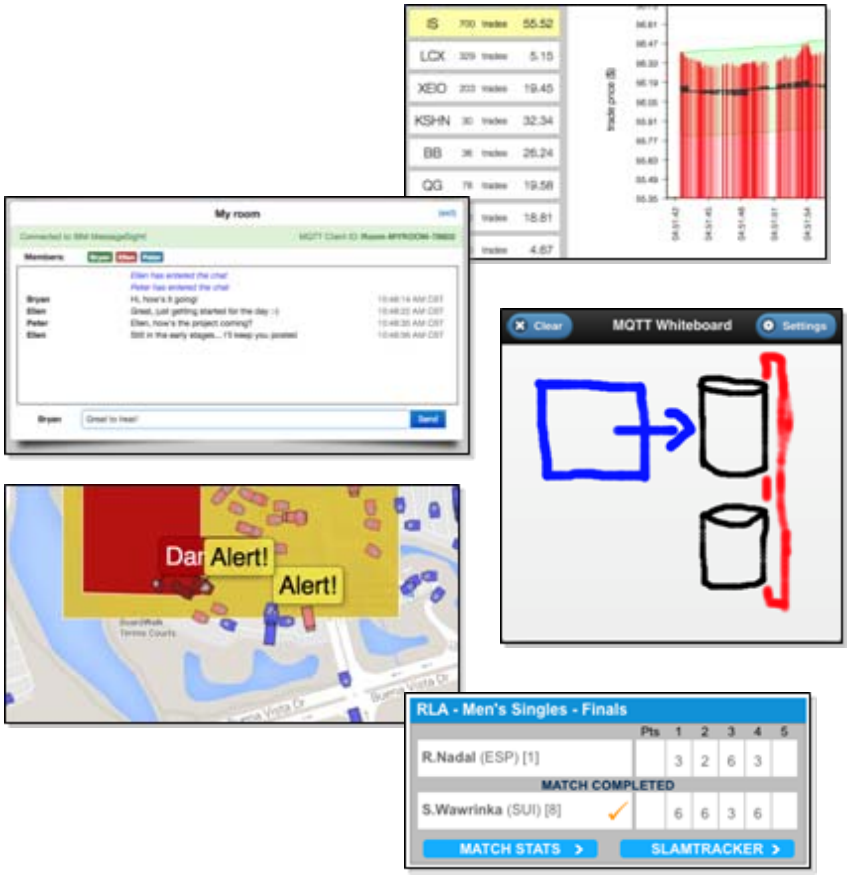
- Getting started with MQ Telemetry Transport
- Building a collaborative application (Whiteboard)
- Connecting to an MQTT broker
  - Mosquitto
  - IBM MessageSight

Getting started with MQ Telemetry Transport

- MQ Telemetry Transport (MQTT) is a lightweight messaging protocol that is designed for Internet of Things (IoT) and mobile connectivity.
- MQ Telemetry Transport provides:
  - Reliable message delivery over unreliable connections
  - Secure message delivery to the enterprise
  - One-to-many message delivery (publish/subscribe)
  - Near real-time push of data from server (no polling)
  - Minimal footprint on-the-wire (only 2-byte header)
  - Reduced battery usage

Getting started with MQ Telemetry Transport (1 of 3)

- MQ Telemetry Transport allows for real-time data push from server to mobile devices, making it ideal for dynamic mobile applications.
- Examples:
  - Stock ticker
  - Chat application
  - Collaboration apps
    - (Whiteboard sharing)
  - Real-time emergency alerts
  - Live match score updates



Getting started with MQ Telemetry Transport (2 of 3)

- MQTT client libraries are available in Java™, C, JavaScript™, C++, Python, Objective-C, and many other programming languages.
- The Eclipse Paho project (http://www.eclipse.org/paho/) provides many open source MQTT clients (http://git.eclipse.org/c/paho).
- IBM provides MQTT clients in a Mobile Messaging & M2M Client Pack, available from the IBM Messaging community on developerWorks.
- This module uses the Eclipse Paho JavaScript MQTT client for publish/subscribe messaging in the IBM Worklight® application.

Getting started with MQ Telemetry Transport (3 of 3)

- The MQTT clients have a simple API for publish/subscribe messaging
- The following examples use the Eclipse Paho JavaScript MQTT client to **Connect** to an MQTT broker, **Subscribe** to an MQTT topic, **Receive** and process messages from the broker, and **Publish** a message on an MQTT topic.

Connect

```
function connect() {
  client = new Messaging.Client(hostname, port, clientId);
  client.onMessageArrived = onMsgCallback;
  client.onConnectionLost = onConnLostCallback;
  client.connect({ onSuccess: onSuccessCallback });
}
```

Subscribe

```
function subscribe(topic) {
  client.subscribe(topic);
}
```

Publish

```
function publish(topic, data) {
  var msg = new Messaging.Message(data);
  msg.destinationName = topic;
  client.send(msg);
}
```

Receive

```
function onMsgCallback(msg) {
  var topic = msg.destinationName;
  var data = msg.payloadString;
  console.log(topic, data);
}
```

Agenda

- Getting started with MQ Telemetry Transport
- Building a collaborative application (Whiteboard)
- Connecting to an MQTT broker
  - Mosquitto
  - IBM MessageSight

Building a collaborative application

- In this module, you build a dynamic application (Whiteboard) with Worklight and MQ Telemetry Transport that lets users draw on a shared canvas in real time. You do this in two steps:
  - **Build the Worklight application**
    - First, implement “single-user” mode.
    - The application captures touch/click events and lets you paint on the canvas.
  - **Implement collaboration using an MQTT client and server**
    - You add a JavaScript MQTT client to the Whiteboard application that communicates with other clients through an MQTT server.
    - MQTT server options are covered in a subsequent slide.
    - **Note**: MQTT data does not use the IBM Worklight security and authentication mechanism.

Building a collaborative application

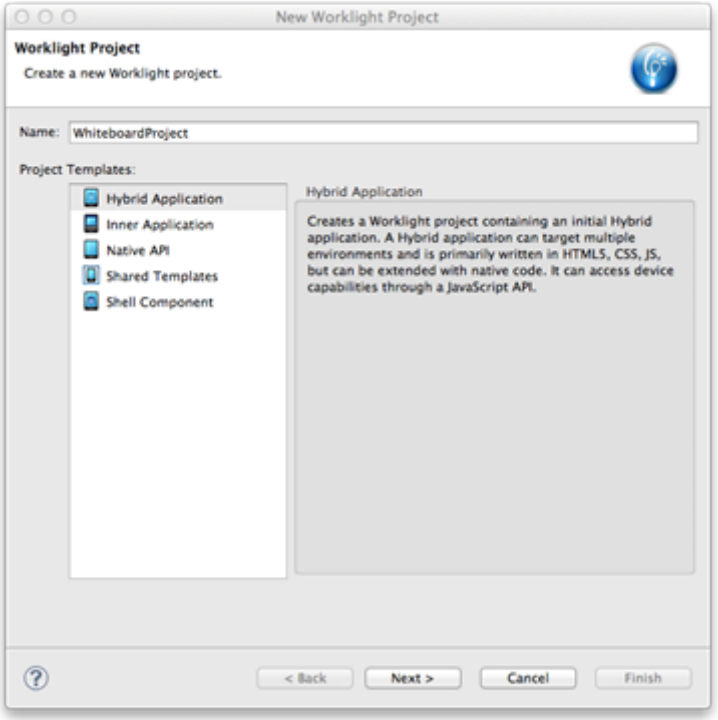
- The Whiteboard application uses MQTT messaging to provide a shared canvas for all users.
- Each Whiteboard will publish all drawing actions as MQTT messages on a topic unique to the application session.
- Each Whiteboard will also subscribe to the set of all drawing topics, and draw others' actions based on this data.
- With MQTT publish/subscribe messaging capabilities, the real-time collaborative experience can scale to many connected applications.

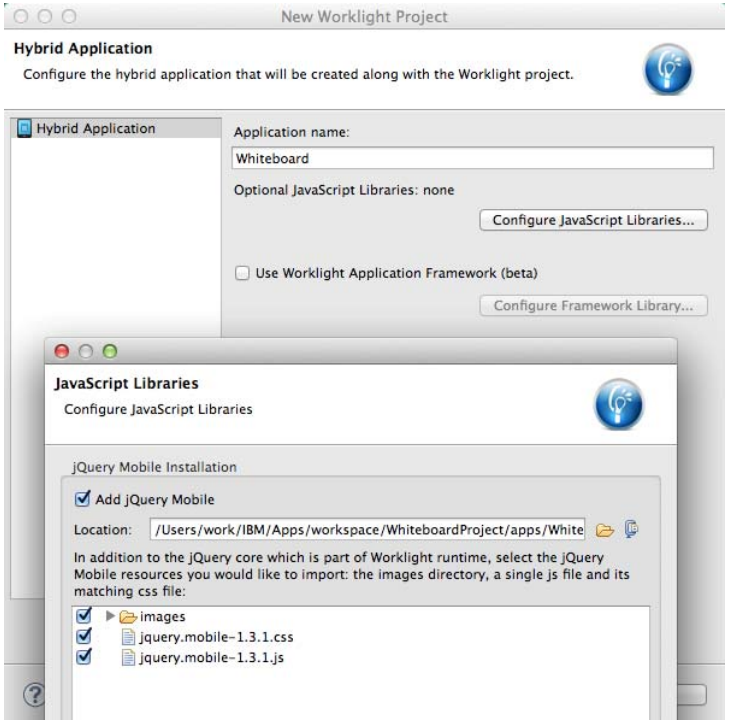
Building a collaborative application

- This scenario cannot efficiently be implemented by using traditional polling (HTTP) or mobile push notifications.
  - To provide a real-time experience, drawing actions should be reflected on other canvases within milliseconds.
  - HTTP polling is high-bandwidth (each request requires a new client connection) and results in poor latency.
  - Push notifications minimize bandwidth, but are inappropriate for small in-application updates.
  - MQTT publish/subscribe messaging minimizes bandwidth and latency: an MQTT connection is established once from client to server, and messages are pushed directly to the application with low latency.

Whiteboard – Create Project

- Create a **WhiteboardProject** project with a Hybrid Application named **Whiteboard**. Add the jQuery Mobile 1.3.1 library, obtained separately.
  - For more information, see the **Working with UI frameworks** module.





Whiteboard – Build UI

- Add a header, a canvas, and a button to clear the canvas.

index.html

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Whiteboard</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=1.0, user-scalable=0">
    <link rel="shortcut icon" href="images/favicon.png">
    <link rel="apple-touch-icon" href="images/apple-touch-icon.png">
    <link href="jqueryMobile/jquery.mobile-1.3.1.css"
rel="stylesheet">
    <link rel="stylesheet" href="css/main.css">
    <script>window.$ = window.jQuery = WLJQ;</script>
    <script src="jqueryMobile/jquery.mobile-1.3.1.js"></script>
  </head>
  <body style="display: none;">
    <div data-role="page" id="page">
      <div data-role="header" id="header" data-position="fixed">
        <h3>MQTT Whiteboard</h3>
        <a onclick="app.clear()" class="ui-btn-left" data-
role="button" data-icon="delete" data-theme="b">Clear</a>
      </div>
      <canvas id="whiteboard"></canvas>
    </div>
    <script src="js/initOptions.js"></script>
    <script src="js/main.js"></script>
    <script src="js/Whiteboard.js"></script>
    <script src="js/messages.js"></script>
  </body>
</html>
```

main.css

```
/* Reset CSS */
a, abbr, address, article, aside, ……
{
  margin: 0;
  padding: 0;
}

body {
  margin: 0 auto;
}

canvas {
  position: absolute;
}
```

Whiteboard – Capture Events

- Capture touch events, and draw on the canvas.

main.js

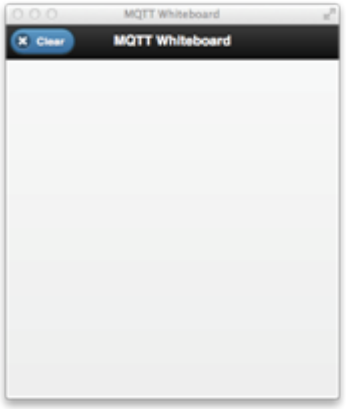
```
function wlCommonInit(){
  window.app = new WhiteboardApp();

  $("#whiteboard").on("vmousemove", function(event) {
    if (app.drawOn) {
      var x = event.pageX;
      var y = event.pageY - $("canvas").offset().top;
      app.draw(x, y, app.size, app.color);
    }
    event.preventDefault();
  });
  $("#whiteboard").on("vmousedown", function(event) {
    var x = event.pageX;
    var y = event.pageY - $("canvas").offset().top;
    app.draw(x, y, app.size, app.color);
    app.drawOn = true;
  });
  $("#whiteboard").on("vmouseup", function(event) {
    app.drawOn = false;
  });
  $("#whiteboard").bind("tap", function(event) {
    event.preventDefault();
  });

  $(window).resize(function() { resize(); });
  resize();
}
```

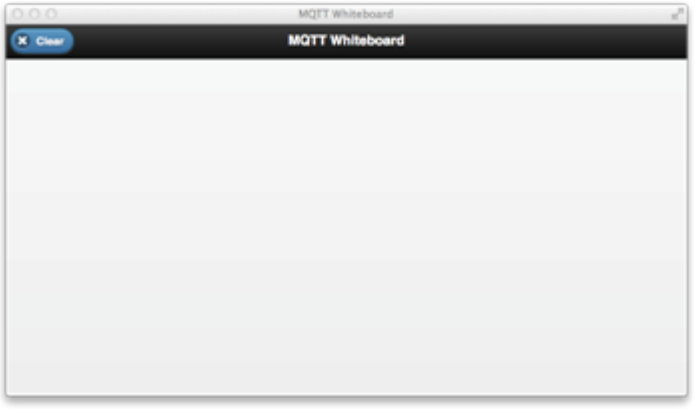
Whiteboard – Capture Events

- Adjust the canvas size to fill available space when the screen resizes.

main.js (cont.)

```
var resize = function() {
  var winSize = {
    width: window.innerWidth ||  document.body.clientWidth,
    height: window.innerHeight || document.body.clientHeight
  };
  // make canvas fill space under header
  $("canvas").css("top", $("#header").innerHeight() + "px");
  $("canvas")[0].width = winSize.width;
  $("canvas")[0].height = winSize.height - $("#header").innerHeight();
}
```





Whiteboard – Drawing

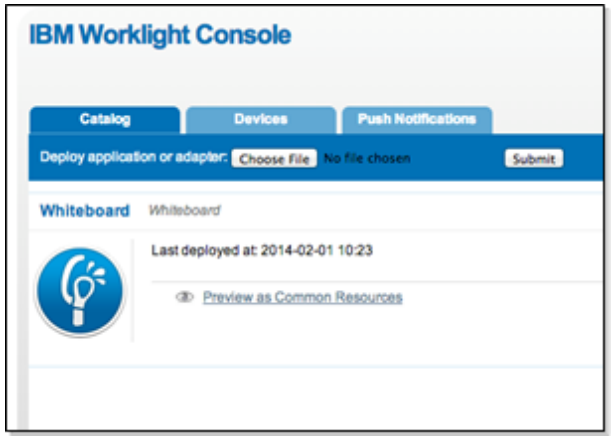
- Define a new object that performs canvas drawing actions.

Whiteboard.js

```
var Colors = [
  "rgb(255,0,0)", "rgb(0,170,0)", "rgb(0,0,255) »,
  "rgb(0,0,0), "rgb(0,255,255)", "rgb(127,255,212)",
  "rgb(139,69,19)"];

function WhiteboardApp() {
  this.size = 8;
  this.color = Colors[Math.floor(Math.random() * Colors.length)];
  this.drawOn = false;
  this.canvas = $("canvas")[0];
}

WhiteboardApp.prototype.draw = function(x, y, size, color) {
  var context = this.canvas.getContext("2d");
  for (var i = 1; i <= size; i+=2) {
    context.save();
    context.beginPath();
    var alpha = 1.0 - Math.pow(i/size, 2);
    context.globalAlpha = alpha;
    context.strokeStyle = color;
    context.arc(x, y, i, 0, 2*Math.PI);
    context.stroke();
    context.restore();
  }
}

WhiteboardApp.prototype.clear = function() {
  var context = this.canvas.getContext("2d");
  context.clearRect(0, 0, this.canvas.width, this.canvas.height);
}
```

Whiteboard – Run the Application

- Before you add collaboration through MQTT messaging, test the application by deploying it to your local Worklight Server.
  - Select **Run As > Run on Development Server**.
  - Open http://localhost:10080/WhiteboardProject/console.
  - Select **Preview as Common Resources**.
  - Click/tap to draw on the canvas.



Whiteboard – Adding collaboration

- You can now add collaborative features to the application with publish/subscribe MQTT messaging. You will add an MQTT client to the Whiteboard application and connect to an MQTT broker when the application loads.
- To create a shared canvas, each Whiteboard client publishes **draw** and **clear** actions on a topic unique to the client. The type of action, and options, are described in the MQTT message payload as JSON data.
- Example: **draw**
  - `Topic: whiteboard/<clientId>`
  - `Payload: {“type” : “draw”, “x” : “127”, “y” : “53”, “color”: “rgb(255,0,0)”}`
- Example: **clear**
  - `Topic: whiteboard/<clientId>`
  - `Payload: {“type” : “clear” }`
- When another Whiteboard client receives this message, `WhiteboardApp.draw()` or `WhiteboardApp.clear()` is called with the parameters from the JSON message.

Whiteboard – Add MQTT client

- Add the Eclipse Paho JavaScript MQTT client to the project:
  - `WhiteboardProject/apps/Whiteboard/common/js/mqttws31.js`
- Load the .js file in `index.html`:

index.html

```
<html>
  <head>
    ...
  </head>
  <body style="display: none;">
    ...
    <script src="js/initOptions.js"></script>
    <script src="js/main.js"></script>
    <script src="js/mqttws31.js"></script>
    ...
```

- On application load, invoke **connect** the client (defined next).

main.js

```
function wlCommonInit(){
  window.app = new WhiteboardApp();
  app.connect();
  ...
```

Whiteboard – Add MQTT client

- When you initialize and connect the MQTT client, you specify the following configuration parameters and connection options.

MQTT Client Configuration	
host	The DNS name or IP of the MQTT broker host
port	The port number for the host
clientId	A unique MQTT identifier for this client (1 to 23 characters)
onMessageArrived	A callback that is called when a message is delivered to the client
onConnectionLost	A callback that is called when the connection has been lost

Connection options	
keepAliveInterval	A period of inactivity (seconds) after which the client will disconnect
onSuccess	A callback that is called after a successful connection to the broker
onFailure	A callback that is called after an unsuccessful connection to the broker

Whiteboard – Add MQTT client

- Add the configuration parameters, callbacks, and connection options, and connect the MQTT client (you will define the callbacks shown in red next).
- If you do not have an MQTT broker available, see **Connecting to an MQTT broker** later in this module for details about how to find your own broker.

Whiteboard.js

```
function WhiteboardApp() {
  ...
  this.host = ”< My MQTT broker IP >";
  this.port = “< My MQTT broker port >”;
  // generate a 6-character alpha-num unique
ID
  this.uuid = Math.random().toString(36).
    slice(2).substring(0, 6);
  this.clientId = "whiteboard-"+this.uuid;
  this.client = new Messaging.Client(
    this.host,
    this.port,
    this.clientId);
}
```

```
WhiteboardApp.prototype.connect = function() {
  this.client.onMessageArrived = (function(self) {
    return function(msg) { self.onMsg(msg); }
  })(this);
  this.client.onConnectionLost = function() {
    alert("Connection lost!");
  };
  var connectOptions = new Object();
  // extend keep-alive to one hour
  connectOptions.keepAliveInterval = 3600;
  connectOptions.onSuccess = (function(self) {
    return function() { self.onConn; }
  })(this);
  connectOptions.onFailure = function() {
    alert("Failed to connect!");
  };
  this.client.connect(connectOptions);
}
```

Whiteboard – Add callbacks

- Next, implement the **onMsg** and **onConn** callbacks.
- When the MQTT connection succeeds, you will subscribe to the wildcard topic of all Whiteboard applications: `whiteboard/+`

Whiteboard.js

```
WhiteboardApp.prototype.onConn = function() {
  this.client.subscribe("whiteboard/+");
}

WhiteboardApp.prototype.onMsg = function(msg) {
  var topic = msg.destinationName;
  var payload = msg.payloadString;

  // make sure the topic matches whiteboard/*
  if (topic.indexOf("whiteboard/") == 0) {
    var sourceUUID = topic.split("/")[1];

    // do not process own actions
    if (sourceUUID == this.uuid) { return; }

    var data = JSON.parse(payload);
    if (data.type == "draw") {
      this.draw(
       data.x, data.y,
       this.size, data.color, true);
    } else if (data.type == "clear") {
      this.clear(true);
    }
  }
}
```

You must differentiate between **draw** and **clear** actions that come from other Whiteboards and those that our Whiteboard initiates.

For actions that you initiate, you will want to publish an MQTT message. If you receive the action *from* an MQTT message, you will not want to republish this action as a new message.

Whiteboard – Publish actions as MQTT messages

- Next, you create and publish MQTT messages for our **draw** and **clear** actions

Whiteboard.js

```
WhiteboardApp.prototype.draw =
  function(x, y, size, color, fromOutside) {
  ...
  if (!fromOutside) {
    this.publishDraw(x, y, color);
  }
}

WhiteboardApp.prototype.clear =
  function(fromOutside) {
  ...
  if (!fromOutside) {
    this.publishClear();
  }
}
```

MQ Telemetry Transport provides three qualities of service for delivering messages between clients and servers:

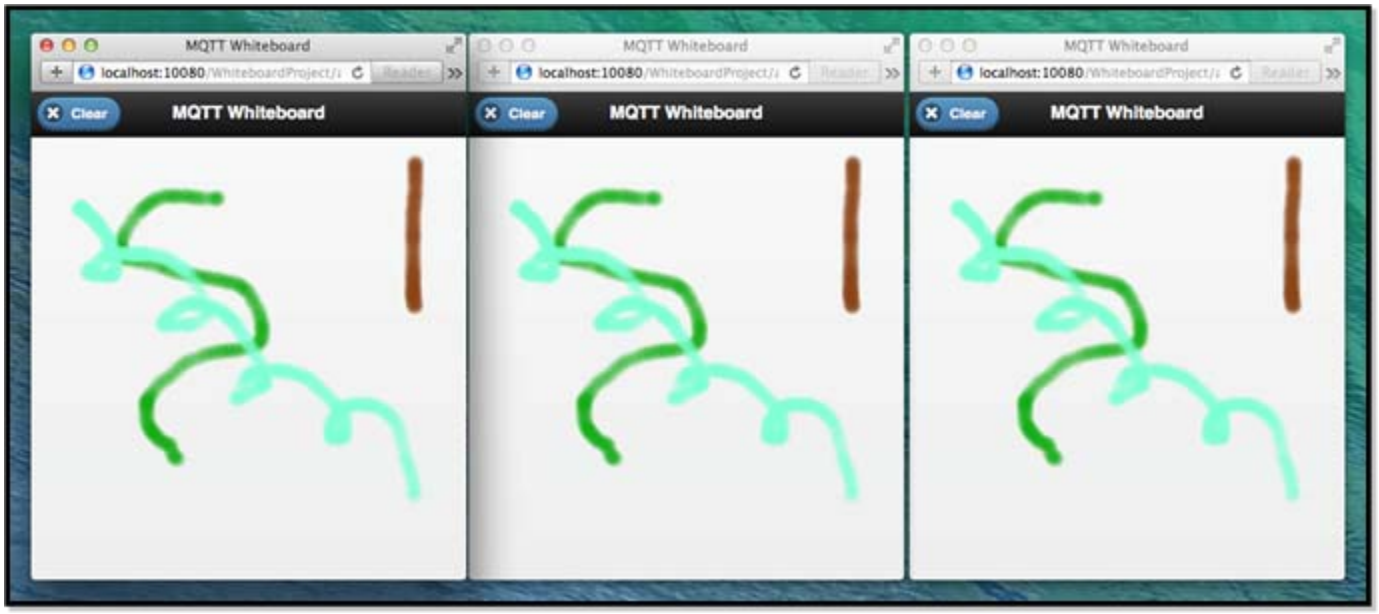
QoS 0 = “at most once”
QoS 1 = “at least once”
QoS 2 = “exactly once”

A **retained** message is retained on the MQTT server and transmitted to new subscribers to the message topic.

```
WhiteboardApp.prototype.publishDraw = function(x, y,
color) {

  var topic = "whiteboard/” + this.uuid;
  var data = JSON.stringify({
    type: "draw",
    x: x,
    y: y,
    color: color,
  });

  var msg = new Messaging.Message(data);
  msg.destinationName = topic;
  msg.qos = 0;
  msg.retained = false;

  this.client.send(msg);
}

WhiteboardApp.prototype.publishClear = function() {

  var topic = "whiteboard/” + this.uuid;
  var data = JSON.stringify({
    type: "clear"
  });

  var msg = new Messaging.Message(data);
  msg.destinationName = topic;
  msg.qos = 0;
  msg.retained = false;

  this.client.send(msg);
}
```

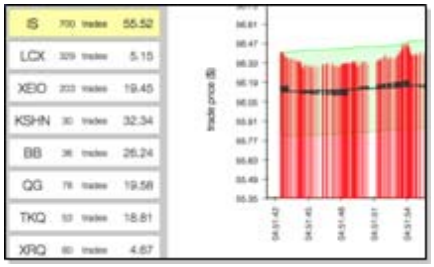
Whiteboard – Putting it all together

- Deploy to the Worklight development server again, and open the application in two (or more) windows.
- Draw in one Whiteboard, and MQTT messages are published to all other subscribing Whiteboard applications.



Extending the collaboration scenario

- This collaboration scenario can be modified and extended to build other types of real-time applications:
  - **Stock ticker**: clients subscribe to trade data for an exchange (for example, MarketData/NYSE/IBM) and update in real time.
  - **Chat application**: clients publish and subscribe on a topic for a particular chat room (for example, chat/room123/user1).
  - **Real-time emergency alerts**: clients subscribe to a topic for their particular geographic location, and emergency services publish alerts onto these topics (for example, emergency/<X>/<Y>/alert).



Agenda

- Getting started with MQ Telemetry Transport
- Building a collaborative application (Whiteboard)
- Connecting to an MQTT broker
  - Mosquitto
  - IBM MessageSight

Connecting to an MQTT broker

- An MQTT server is required to broker messages between MQTT clients. To support JavaScript MQTT clients (such as Eclipse Paho), the MQTT server must implement the WebSocket transport.
- Several MQTT servers are available, both open source and commercial.
  - Development MQTT servers:
    - Mosquitto (open source, multi-platform)
    - IBM MessageSight for Developers (virtual appliance image)
  - Enterprise MQTT servers:
    - IBM MessageSight (appliance)

Mosquitto

- Mosquitto is an open source, lightweight implementation of MQ Telemetry Transport V3.1, part of the Eclipse Paho messaging project.
- The Mosquitto broker is available for many different platforms (Windows, OS X, Linux distributions).
- More information:
  - http://mosquitto.org
  - http://projects.eclipse.org/projects/technology.mosquitto
- Download Mosquitto:
  - http://mosquitto.org/download/

IBM MessageSight for Developers

- IBM MessageSight for Developers is a virtual IBM MessageSight appliance image.
- MessageSight is a low-latency, reliable, and scalable messaging server with strong security and easy management.
- The developer image can be run with virtualization software such as VMWare, Oracle VirtualBox, and KVM.
- More information:
  - http://ibm.com/messagesight
  - https://www.ibmdw.net/messaging/messagesight/
  - http://www.youtube.com/watch?v=kcEDoRqhkhA

Notices

- Permission for the use of these publications is granted subject to these terms and conditions.
- 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.
- For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:
  - Intellectual Property Licensing
    Legal and Intellectual Property Law
    IBM Japan Ltd.
    1623-14, Shimotsuruma, Yamato-shi
    Kanagawa 242-8502 Japan
- **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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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.
- Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:
  - IBM Corporation
    Dept F6, Bldg 1
    294 Route 100
    Somers NY 10589-3216
    USA
- Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.
- The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.
- 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.

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.
- Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:
  - © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Privacy Policy Considerations

- IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.
- Depending upon the configurations deployed, this Software Offering may use session cookies that collect session information (generated by the application server). These cookies contain no personally identifiable information and are required for session management. Additionally, persistent cookies may be randomly generated to recognize and manage anonymous users. These cookies also contain no personally identifiable information and are required.
- If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent. For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the sections entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Support and comments

- For the entire IBM Worklight documentation set, training material and online forums where you can post questions, see the IBM website at:
  - http://www.ibm.com/mobile-docs
- **Support**
  - Software Subscription and Support (also referred to as Software Maintenance) is included with licenses purchased through Passport Advantage and Passport Advantage Express. For additional information about the International Passport Advantage Agreement and the IBM International Passport Advantage Express Agreement, visit the Passport Advantage website at:
    - http://www.ibm.com/software/passportadvantage
  - If you have a Software Subscription and Support in effect, IBM provides you assistance for your routine, short duration installation and usage (how-to) questions, and code-related questions. For additional details, consult your IBM Software Support Handbook at:
    - http://www.ibm.com/support/handbook
- **Comments**
  - We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this document. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.
  - For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.
  - When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state.
  - Thank you for your support.
  - Submit your comments in the IBM Worklight Developer Edition support community at:
    - https://www.ibm.com/developerworks/mobile/worklight/connect.html
  - If you would like a response from IBM, please provide the following information:
    - Name
    - Address
    - Company or Organization
    - Phone No.
    - Email address

Thank You