PROGRESS EXCHANGE 2013
DISCOVER. DEVELOP. DELIVER.

Zero-Footprint Data Monitor

Dustin Grau, Software Architect

BRAVEPOINT

5000 Peachtree Ind. Blvd.
Suite 100
Norcross, GA 30071

About the Presenter

BRAVEPOINT

- Senior developer and consultant at BravePoint, Inc.
  - Founded in 1987 with currently ~125 employees
  - Consulting, training, and placement services
- WebSpeed application developer since 1999
  - Implementing JS/AJAX frameworks since 2010
  - Lead architect for modernization framework “Application Evolution”

Session Overview

BRAVEPOINT

- Gain insight into a running environment using existing ABL tools
- Quick primer on integrating ABL with Node.js and Socket.io
- You should leave here able to do all of this on your own!

- You will be able to download all software shown here today!

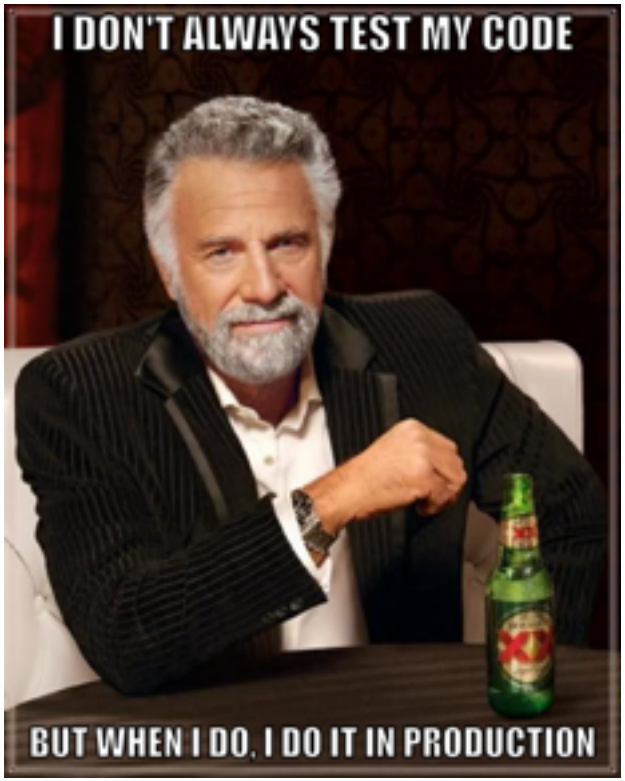
Agenda

BRAVEPOINT

- The Problem
- A Solution
- Demonstration
- Code How-to
- Future Enhancements
- Summary / Q&A

The Problem

- Client is having issues on a running production environment
- You need to get live statistics where the problem is occurring
- Typically difficult to access information about production systems
- Developers are not given access to production for some reason...



A Solution

BRAVEPOINT

- Should ideally...
  - Not require additional software install on production
  - Keep a low profile (cpu/memory), with minimal setup
  - Be able to monitor continuously (active or proactive)
  - Handle multiple connections from interested parties
  - Data should update automatically, in near real-time
  - Maintain a level of security among data and subscribers

A Solution

BRAVEPOINT

- Our solution...
  - Is meant for developers, DBA's, or anyone needing information
  - Uses only ABL code on the observed environment
  - Works on any operating system where OpenEdge can be installed
  - Utilizes a separate, central server for all client connections
  - Is accessible by any web browser on any device
  - Uses WebSockets to push data to subscribed clients

WebSockets?

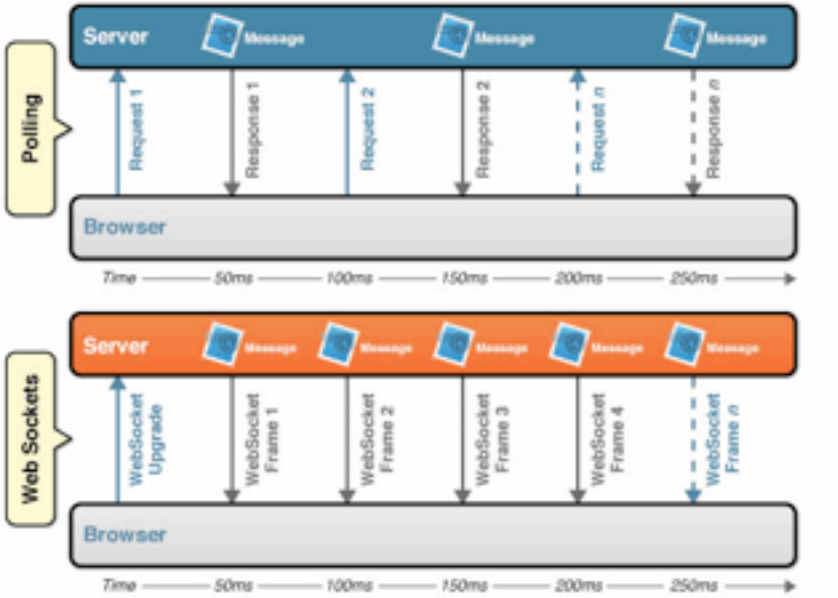
BRAVEPOINT

- RFC 6455 - The WebSocket Protocol
- Provides 2-way communication to supported browsers
- Begins as a standard HTTP GET request from client
- Requests an “upgrade” to WebSocket protocol
- Server performs handshake and initiates connection
- Client remains connected to server
- Can be provided by Node.js and Socket.io
- This is where the magic happens...

Performance Considerations

BRAVEPOINT

- AJAX vs. WebSockets
- AJAX is expensive (headers, overhead, etc.)
- Polling causes network traffic for useless data
- WebSockets have a small handshake, occurs once
- Lower latency when using WebSockets vs. HTTP
- Means nothing if your browser doesn't support WS



Parts List

- Progress OpenEdge (11+)
- Node.js (http://nodejs.org)
- Socket.io (http://socket.io)
- jQuery (http://jquery.com)
- A server (AWS, Rackspace, etc.)
- A browser (Chrome, Firefox, etc.)
- A device (tablet, phone, or PC)

Why Node.js?

BRAVEPOINT

- It's extremely lightweight, both for installation and runtime
- It can handle many, many simultaneous connections
- Can provide multiple services over the same port
- Has a package manager to provide additional capabilities
- Install is as simple as "npm install socket.io"
- Allows for event-driven applications using "on" statements
- Socket.io provides a single solution for 2-way communication
- Can use AJAX long polling and other fallback mechanisms

Original Output

BRAVEPOINT

12:23:55 Top Table I/O Information Reset: off

Table	Reads	Upd	Cre	Del	Tot Rds	Rows
>UserProfile	8	0	0	0	7,134	220
pzSiteRegistry	4	0	0	0	4,150	13
pzRegistration	2	0	0	0	1,715	10
GLTransaction	0	0	0	0	900,927	122,012
ItemGLTransJournal	0	0	0	0	841,076	429,411
ARGLTransJournal	0	0	0	0	538,289	229,286
GLTransJournalDeta	0	0	0	0	316,675	3,435
GLPeriod	0	0	0	0	296,297	256
GLTransactionTotal	0	0	0	0	191,259	112,350
GLAccount	0	0	0	0	122,061	2,924
UserData	0	0	0	0	9,719	11,851
JobTemplateCat	0	0	0	0	8,441	650
GLAcctRptCol	0	0	0	0	6,706	238
AppCode	0	0	0	0	5,464	5,147
JobCategory	0	0	0	0	5,157	287

Top User I/O Info

Name	Pid	Req	Reads	Hit%
>root	6938	330	0	100.0
root	6934	50	0	100.0
root	1629	0	0	?
root	6427	0	0	?
root	7064	0	0	?
root	7119	0	0	?
root	7235	0	0	?
root	7255	0	0	?
root	7657	0	0	?
root	7766	0	0	?
root	7768	0	0	?
root	6423	0	0	?
root	1633	0	0	?
root	1619	0	0	?
root	1644	0	0	?

File-Name	Index	Reads	Cre	Del	Split	Blk Del
>pzField	idxPrime	148	0	0	0	0
UserProfile	UserProfile	10	0	0	0	0
pzSiteRegistry	idxPrime	4	0	0	0	0
pzTable	idxPrime	4	0	0	0	0
pzRegistration	idxKey	2	0	0	0	0
RptAPPaymentTran	idxPrim	0	0	0	0	0
ItemPrice	pukPrice	0	0	0	0	0
DispatchSchedule	pukDispatchSched	0	0	0	0	0
ItemPrice	idxListPrice	0	0	0	0	0
RptAPVoucherDeta	idxPrim	0	0	0	0	0
ItemPrice	idxObsolete	0	0	0	0	0
RptAPVoucherDist	idxPrim	0	0	0	0	0
ItemRecUpDetail	pukRecUpNo	0	0	0	0	0
RptAPVoucherHead	idxPrim	0	0	0	0	0
ItemRecUpHeader	pukRecUpNo	0	0	0	0	0

DB: live /opt/ProfitZoom/devel/bin/databas
Requests: 15992986 Reads: 16161
Curr: 99.90% Full: 99.90%
load average: 0.07, 0.02, 0.02
RX: 23,698 TX: 19,270

1		0.000
2		0.000
3		0.000
4		0.000
5		0.000
6		0.000
7		0.000
8		0.000
9		0.000
10		0.000
11		0.000

Enhanced Output

BRAVEPOINT

Subscribed to db3b25a674d1819ce21162fb1d3ffad225467603 *Updated at 12:08:27* Reset Stats

Table Activity | Index Activity | User Activity

Username	Tenant	PID	DB Access	DB Reads	New Access	New Reads
demo@rei_com	REI	5348	9244	53	444	8
Dustin@windowsid	Default	5096	8003	1	404	0
demo@rei_com	REI	7560	9348	48	102	3
Anonymous	Default	568	11536	21	3	0
SYSTEM	NA	5928	3942	285	0	0
SYSTEM	NA	5800	4	0	0	0
SYSTEM	NA	4504	4	0	0	0
DB_Agent	NA	5268	4565993	138	0	0
SYSTEM	NA	872	4	0	0	0
SYSTEM	NA	4588	4	0	0	0
SYSTEM	NA	5416	13	0	0	0
SYSTEM	NA	5780	13	0	0	0
Anonymous	Default	3468	12178	136	0	0

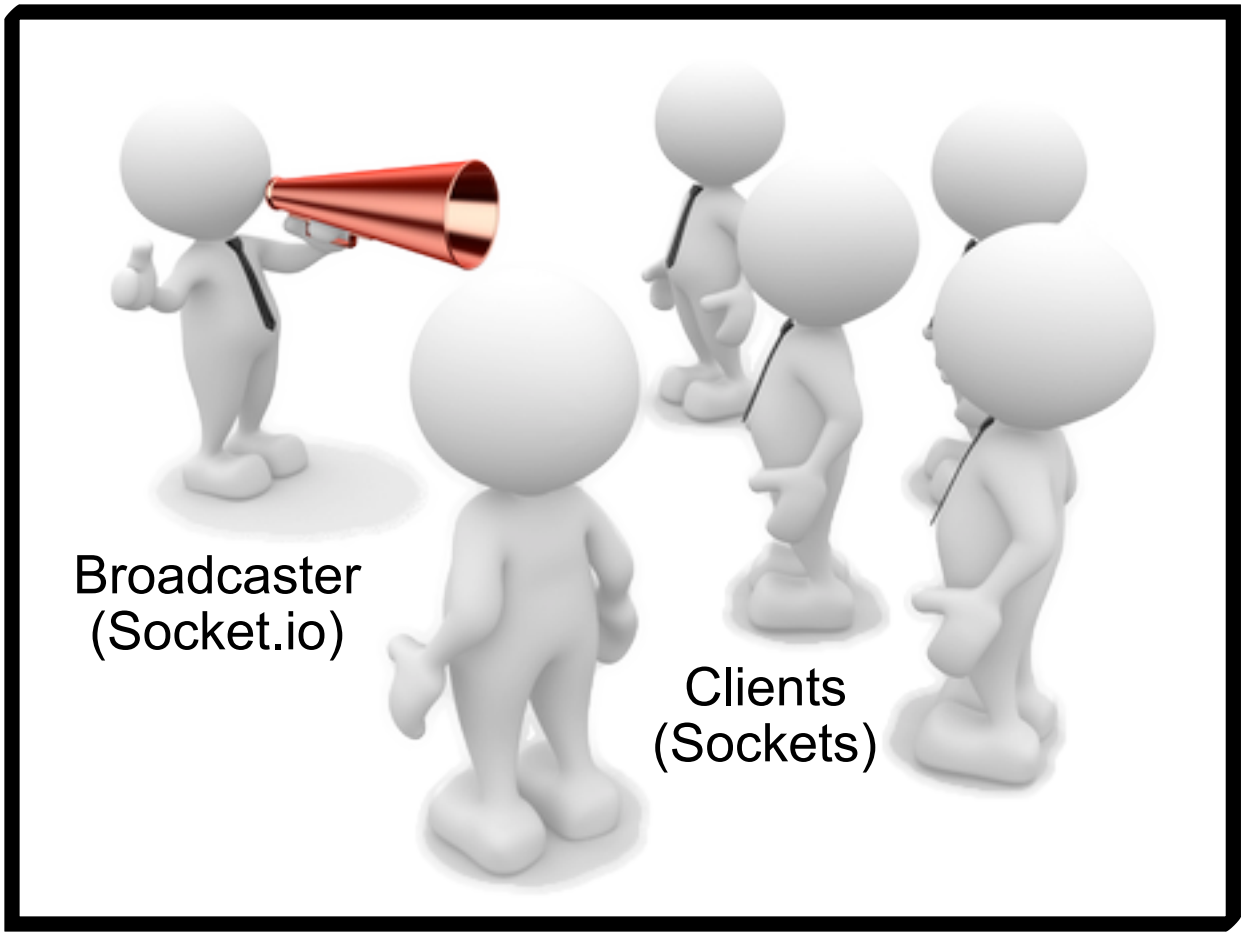
Action Overview

Harvester
(ABL)

UUID

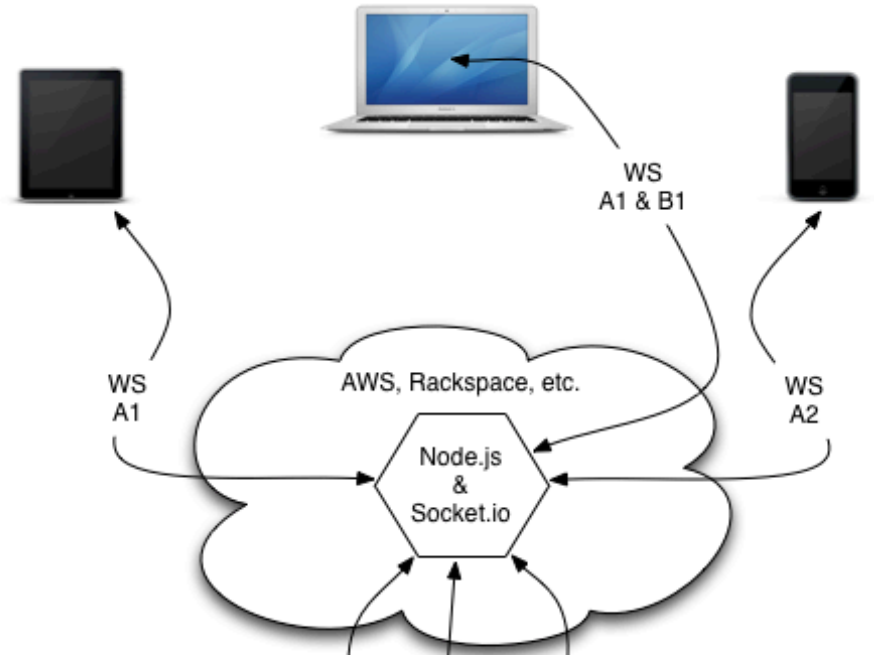
Data (POST)

Room (Node.js)



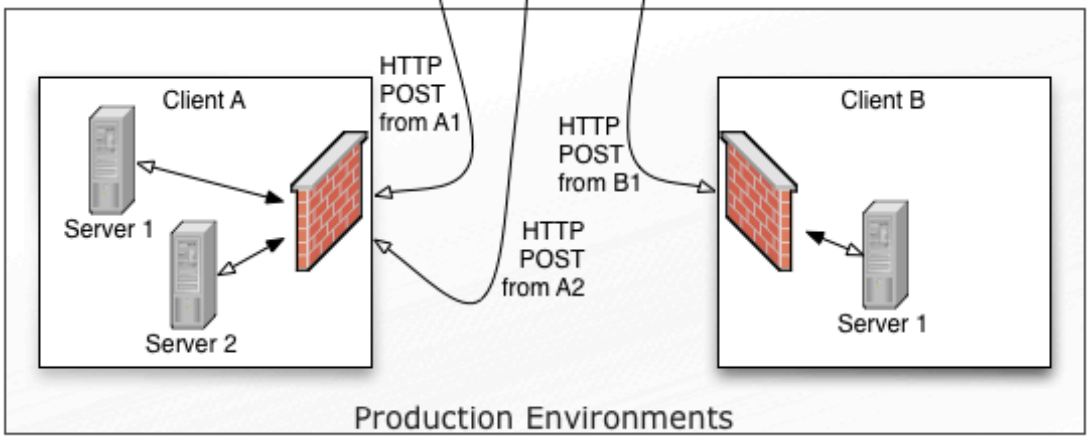
Architecture Overview

BRAVEPOINT



A1: `db3b25a674d1819ce21162fb1d3ffad225467603`

B1: `ec4c36b785e292adf322710c2e510be336578714`



EXCHANGE 2013

Code How-to

BRAVEPOINT

- Progress (Harvester)
  - monitor.p 
    - SysLoad.cls
    - WebSocket.cls
- Node.js (Broadcaster)
  - index.js
    - Socket.io (Server)
  - client.html
    - Socket.io (Client)

Startup (ABL)

BRAVEPOINT

start.sh

```
#!/bin/bash
export PROPATH=./;$1
_progres -b -p monitor.p -pf startup.pf > logs/monitor.log
```

startup.pf

```
# Database to Monitor

-db Sports2kMT
-H localhost
-N TCP
-S 8650

-rereadnolock
-T ./temp
-TB 31
-TM 32
-rand 2
-mmax 8000
```

monitor.p

BRAVEPOINT

```
&GLOBAL-DEFINE THROW ON ERROR UNDO, THROW
&GLOBAL-DEFINE SERVER_IP "127.0.0.1"
&GLOBAL-DEFINE SERVER_PORT 1337
&GLOBAL-DEFINE UUID "db3b25a674d1819ce21162fb1d3ffad225467603"
&GLOBAL-DEFINE PROCESS_WAIT 10
&GLOBAL-DEFINE MAX_ROWS 30

ROUTINE-LEVEL ON ERROR UNDO, THROW.
USING com.bravepoint.*.

DEFINE VARIABLE oSysLoad AS SysLoad NO-UNDO.
ASSIGN oSysLoad = NEW SysLoad( {&SERVER_IP},
                               {&SERVER_PORT},
                               {&UUID},
                               {&PROCESS_WAIT},
                               {&MAX_ROWS} ).

MESSAGE SUBSTITUTE("[&1 &2] Starting monitor...",
                   STRING(TODAY, "99/99/9999"), STRING(TIME, "HH:MM:SS")).
oSysLoad:startMonitor().

FINALLY:
    DELETE OBJECT oSysLoad NO-ERROR.
END FINALLY.
```

SysLoad.cls

BRAVEPOINT

```
 METHOD PUBLIC VOID startMonitor ():
...
MAINBLK:
REPEAT:
    waitFor().
    buildSample().

    IF DATASET activityData:WRITE-JSON("LONGCHAR", cRequest, FALSE,
                                       "UTF-8", FALSE, TRUE) THEN DO:
        jsonRequest:Add(INPUT "activityData", INPUT cRequest).
        jsonRequest:Write(INPUT-OUTPUT cRequest, INPUT TRUE, INPUT "UTF-8").
        jsonRequest:Remove(INPUT "activityData").
        oWebSocket:sendData( INPUT  SUBSTITUTE("http://&1:&2/&3",
                                               cServerAddr, iServerPort, cMonitorUUID),
                             INPUT  cRequest,
                             OUTPUT iStatus,
                             OUTPUT cMimeType,
                             OUTPUT cResponse ).
        ...
    END. /* WRITE-JSON */

    ...
END.
END METHOD. /* startMonitor */
```

Code How-to

BRAVEPOINT

- Progress (Harvester)
  - monitor.p
    - SysLoad.cls
    - WebSocket.cls
- Node.js (Broadcaster)
  - index.js 
    - Socket.io (Server)
  - client.html
    - Socket.io (Client)

index.js

BRAVEPOINT

```
/**
 * Provide service at a default port.
 */
var listen_port = 1337;

/**
 * Create an HTTP server to handle regular web requests.
 */
var http = require("http"); // HTTP service
var url = require("url"); // URL parser
var fs = require("fs"); // FileSystem
var httpServer = http.createServer(onHttpRequest).listen(listen_port);

/**
 * Handle socket.io connections.
 */
var io = require("socket.io").listen(httpServer);
io.sockets.on("connection", onSocketConnect);
```

index.js (onHttpRequest)

BRAVEPOINT

```
function onHttpRequest(request, response) {
    var parsedUrl = url.parse(request.url);
    var pathName = parsedUrl.pathname || "";
    var pathArray = pathName.split("/"); // Convert to array.
    var pathVal = (pathArray.length > 1) ? pathArray[1] : "";

    switch(request.method){
        case "GET":
            doGet(pathVal, request, response);
            break;
        case "POST":
            doPost(pathVal, request, response);
            break;
        default:
            // Unsupported HTTP method.
            response.writeHeader(405, "Method Not Allowed",
                                 {"Content-Type": "text/plain"});
            response.end();
    }
}
```

index.js (doGet)

BRAVEPOINT

```
function doGet(pathVal, request, response){
    var filename = null;

    switch(pathVal){
        case "":
        case "client.html":
            // Serve the client HTML file on GET.
            filename = "/client.html";
            break;
        default:
            // File not found for serving.
            response.writeHeader(404, "Not Found",
                                 {"Content-Type": "text/plain"});
            response.end();
    }

    if (filename) {
        fs.readFile(__dirname + filename, "utf8", function(error, content) {
            response.writeHeader(200, "OK", {"Content-Type": "text/html"});
            response.end(content);
        });
    }
}
```

index.js (doPost)

BRAVEPOINT

```
function doPost(pathVal, request, response){
    var uuid = pathVal; // In this case the path value is a UUID.
    var postData = ""; // Store the body data on POST.
    var count = 1;

    request.on("data", function(chunk){
        postData += chunk;
        if (postData.length > 1e6) {
            postData = ""; // Abort if data appears to be a [malicious] flood.
            response.writeHeader(413, {"Content-Type": "text/plain"}).end();
            request.connection.destroy();
        }
        count++;
    }); // on data

    ...
```

index.js (doPost)

BRAVEPOINT

```
request.on("end", function(){
    var responseBody = {response: "Broadcast Sent: " + uuid};
    if (postData != "") {
        var jsonObj = null;
        try {
            jsonObj = JSON.parse(postData);
        } catch(parseErr) {
            responseBody = {response: "JSON Error: " + parseErr.message};
        }

        // Add the UUID for socket broadcast.
        if (harvesters.indexOf(uuid) < 0) {
            harvesters.push(uuid);
        }

        // Broadcast to clients on socket server, based on UUID.
        if (jsonObj && io.sockets.clients(uuid).length > 0) {
            var room = io.sockets.in(uuid);
            room.emit("broadcast-data", jsonObj.activityData);
        }
    } // postData

    ...
```

index.js (doPost)

BRAVEPOINT

```
        // Prepare response body with optional commands.
        if (commands[uuid]) {
            responseBody.commands = commands[uuid];
            delete commands[uuid];
        }

        // End the response with a message.
        var responseJSON = JSON.stringify(responseBody);
        var responseHeaders = {
            "Content-Type": "application/json",
            "Access-Control-Allow-Origin": "*",
            "Content-Length": Buffer.byteLength(responseJSON)
        };
        response.writeHeader(200, "OK", responseHeaders);
        response.write(responseJSON);
        response.end();
    }); // on end
}
```

index.js (onSocketConnect)

BRAVEPOINT

```
function onSocketConnect(socket){
    /**
     * Handle requests to listen for broadcast consoles
     */
    socket.on("listen", function(uuid, callback) {
        // If the uuid looks legit, subscribe.
        if (uuid.length == 40) {
            socket.uuid = uuid; // Save UUID on socket.
            socket.join(uuid); // Join a "room" for UUID.

            // Callback to the listener with a successful flag.
            callback(true);
        } else {
            // If the uuid is not available, reject the request to listen.
            callback(false);
        }
    }); // on listen

    ...
```

index.js (onSocketConnect)

BRAVEPOINT

```
    ...

    /**
     * Handle requests for new commands back to harvester
     */
    socket.on("send-command", function(uuid, command) {
        // If the uuid is being broadcast, add to queue.
        if (harvesters.indexOf(uuid) >= 0) {
            if (commands[uuid] && commands[uuid] instanceof Array) {
                // Queued commands pending, add to existing list.
                if (commands[uuid].indexOf(command) == -1) {
                    // Prevent adding duplicate commands.
                    commands[uuid].push(command);
                }
            } else {
                // No queued commands, create new queue for uuid.
                commands[uuid] = [command];
            }
        }
    }); // on send-command
}
```

Startup (Node.js)

BRAVEPOINT

```
Windows:
C:\> node index.js


Linux:
# node index.js
```

Code How-to

BRAVEPOINT

- Progress (Harvester)
  - monitor.p
    - SysLoad.cls
    - WebSocket.cls
- Node.js (Broadcaster)
  - index.js
    - Socket.io (Server)
  - client.html 
    - Socket.io (Client)

client.html (HTML)

BRAVEPOINT

```
<body>
    <form id="sub-form">
        <label for="uuid-input">Enter a UUID:</label>
        <input id="uuid-input" maxlength="40" size="50" value="db3b25a674d1819ce21162fb1d3ffad225467603" />
        <input type="submit" id="sub-button" value="Subscribe" />
        <div id="uuid-error">The specified UUID is invalid.</div>
    </form>
    <div id="console">
        <div id="monitor-header">
            <div id="monitor-uuid"></div>
            <div id="monitor-time"></div>
            <div id="monitor-controls">
                <form id="command-form"><input type="button" id="reset-button" value="Reset Stats" /></form>
            </div>
        </div>
        <div id="monitor-data">
            <ul>
                <li><a href="#tab-1">Table Activity</a></li>
                <li><a href="#tab-2">Index Activity</a></li>
                <li><a href="#tab-3">User Activity</a></li>
            </ul>
            <div id="tab-1">
                <table id="activity-table"></table>
            </div>
            <div id="tab-2">
                <table id="index-table"></table>
            </div>
            <div id="tab-3">
                <table id="user-table"></table>
            </div>
        </div>
    </div>
</body>
```

client.html (CSS/JS)

BRAVEPOINT

```
<link rel="stylesheet" src="http://normalize-css.googlecode.com/svn/trunk/normalize.css" />

<link rel="stylesheet" href="http://code.jquery.com/ui/1.10.3/themes/smoothness/jquery-ui.css" />


<script src="http://code.jquery.com/jquery-1.10.2.js"></script>

<script src="http://code.jquery.com/ui/1.10.3/jquery-ui.js"></script>

<script src="/socket.io/socket.io.js"></script>
```

client.html (JS)

BRAVEPOINT

```
<script>
    $(document).ready(function() {
        // Give focus to the UUID input when the page loads
        $("#uuid-input").focus();

        // Create a connection to the server
        var socket = io.connect(document.URL);

        // Prepare tab interface.
        $("#monitor-data").tabs();

        // Handler for updating screen data.
        $.updateConsole = function(data){ ... }

        // Handler for subscription event.
        $.doSubscribe = function(){ ... }

        // Handle submission of the form, try to subscribe to the UUID.
        $("#sub-form").submit(function(ev){
            // Prevent the browser from submitting the form via HTTP
            ev.preventDefault();

            // Attempt to subscribe to the given UUID.
            $.doSubscribe();
        });
    });
</script>
```

client.html (doSubscribe)

BRAVEPOINT

```
$.doSubscribe = function(){
    var uuid = $("#uuid-input").val();
    if (uuid) {
        socket.emit("listen", uuid, function(successful) {
            if (successful) {
                // Hide the subscription form and show the main console.
                $("#sub-form").hide();
                $("#console").show();
                $("<span>").addClass("system-message").text("Subscribed to "
                                                    + uuid).appendTo("#monitor-uuid");

                // Handle incoming broadcasts (sends "data" as a parameter).
                socket.on("broadcast-data", $.updateConsole);

                // Handle sending of commands.
                $("#reset-button").click(function(ev) {
                    // Queue command for the next broadcast from harvester.
                    socket.emit("send-command", uuid, "reset");
                    alert("Statistics will be reset after the next update.");
                });
            } else {
                // If the request to subscribe was rejected, show an error message.
                $("#uuid-error").show();
            }
        });
    }
}
```

client.html (updateConsole)

BRAVEPOINT

```
$.updateConsole = function(data){
// Split data into tables.
var monitorData = null;
if (typeof(data) == "string") {
    try {
        // Always parse in a try/catch block!
        monitorData = JSON.parse(data);
    } catch(parseErr) {
        // Fail silently.
    }
} else if (typeof(data) == "object") {
    monitorData = data;
}
monitorData = monitorData || {};
var activity = monitorData.returnAct || [];
var indexAct = monitorData.returnIdxAct || [];
var userAct = monitorData.returnUsrAct || [];

$("#activity-table").empty();
$("#index-table").empty();
$("#user-table").empty();
...
}
```

Event Overview

BRAVEPOINT

Harvester	Broadcaster	Client
`oSysLoad:startMonitor()`	`httpServer = http.createServer` `io.listen(httpServer)`	`socket = io.connect(document.URL)`
	`socket.on("listen", ...)`	`socket.emit("listen", ...)`
`oWebSocket:sendData(...)`	`room = io.sockets.in(uuid)` `room.emit("broadcast-data", ...)`	`socket.on("broadcast-data", ...)`
[POST Response]	`socket.on("send-command", ...)`	`socket.emit("send-command", ...)`

Demo (Revisited)

BRAVEPOINT

Subscribed to db3b25a674d1819ce21162fb1d3ffad225467603 *Updated at 12:08:27* Reset Stats

Table Activity | Index Activity | User Activity

Username	Tenant	PID	DB Access	DB Reads	New Access	New Reads
demo@rei_com	REI	5348	9244	53	444	8
Dustin@windowsid	Default	5096	8003	1	404	0
demo@rei_com	REI	7560	9348	48	102	3
Anonymous	Default	568	11536	21	3	0
SYSTEM	NA	5928	3942	285	0	0
SYSTEM	NA	5800	4	0	0	0
SYSTEM	NA	4504	4	0	0	0
DB_Agent	NA	5268	4565993	138	0	0
SYSTEM	NA	872	4	0	0	0
SYSTEM	NA	4588	4	0	0	0
SYSTEM	NA	5416	13	0	0	0
SYSTEM	NA	5780	13	0	0	0
Anonymous	Default	3468	12178	136	0	0

Future Enhancements

BRAVEPOINT

- Integrate with a database for historic logging/retrieval of data
- Improve security of the application endpoints beyond a UUID
- Improve portability and deployment of the application
- Implement an ABL WebSocket for connecting to a socket.io server
- Extend beyond just server monitoring (think “push notifications”)

Summary

BRAVEPOINT

- Fully use what is available to you in the ABL environment
- No third-party installations necessary for the server-side
- Use of a separate server for client connections bridges the gap
- Organize connections into manageable notification groups
- Node.js is lightweight and serves all connections (HTTP & WS)
- Socket.io provides realtime data and handles fallback gracefully
- Certainly not limited to just the scenario presented here
- Code is available for you to use immediately!
- So, any questions?

Thank You!

BRAVEPOINT

Dustin Grau

dgrau@bravepoint.com

http://goo.gl/Xoikn9