Mainframe and Mobile: Perfect Together 16036

Monday, March 2, 2015: 3:15PM-4:15PM

Ron Piracci, Jason Fournier

Verizon

SHARE in Seattle 2015

Why is the Mainframe & Mobile Perfect together?

- Backend systems already in place
- Capable of workload handling
  - IBM Mobile Workload Pricing
- Almost no new code to original programs
  - May need an adapter
- Tools already in place
  - z/OSMF REST APIs, CICS Web Services, z/OS Connect, etc.

What did we try to accomplish?

- Lower MTTR
  - Can see all systems together to determine if one failure may have caused collateral damage
- Meet SLA’s
  - Bring systems up quicker when less time is wasted checking systems
- Quicker troubleshooting/problem resolution
  - Allows techs to work quicker on bridges to determine problems by being able to isolate real issues quicker

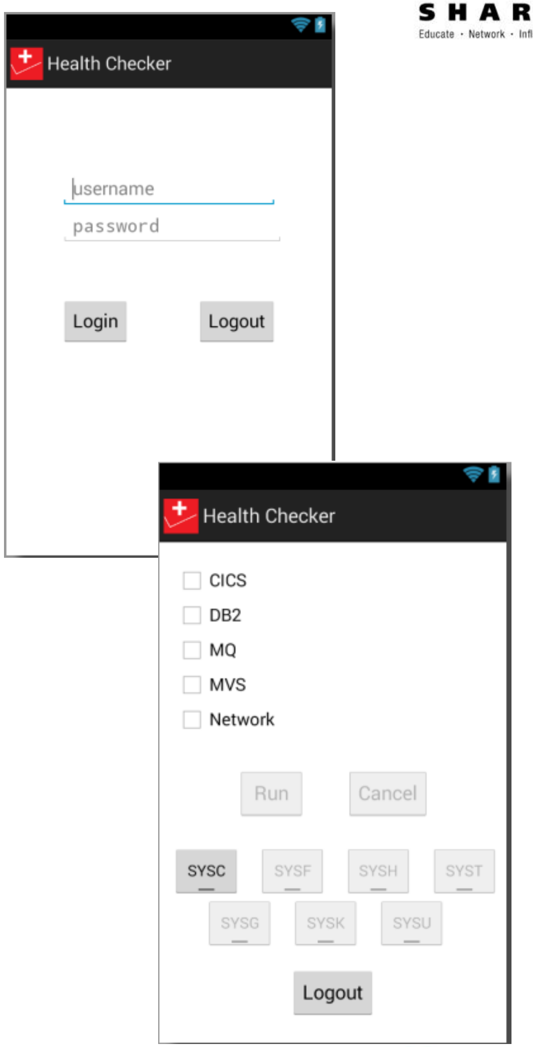
Why did we need a mobile app?

- Ease of use
  - Easy GUI and minimal interaction needed
- Availability (can access anywhere)
  - Whether on-call or just curious
- Speed - Current process lengthy
  - Laptops take too long
    - Boot/awake, VPN/reconnect, start session, run jobs, decipher results

How does the app work?

- Login Screen
  - Login to system
  - Input validation
  - HTTPS, certificate verification
  - Return LTPA2 token

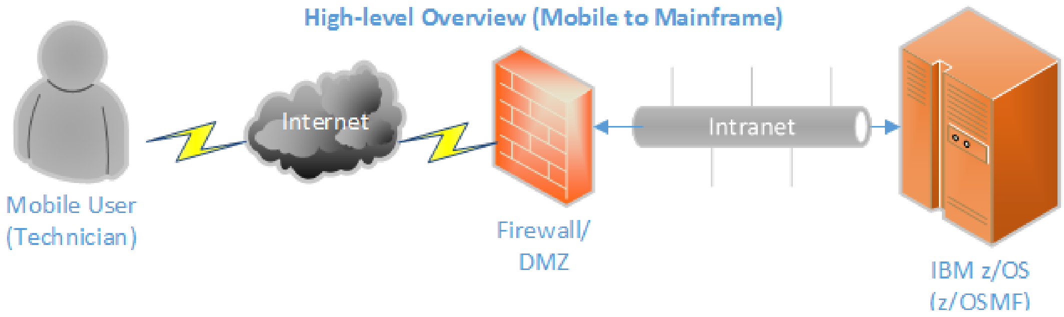
- Main Menu
  - Allows one or more health checks to run
  - Will run on one LPAR



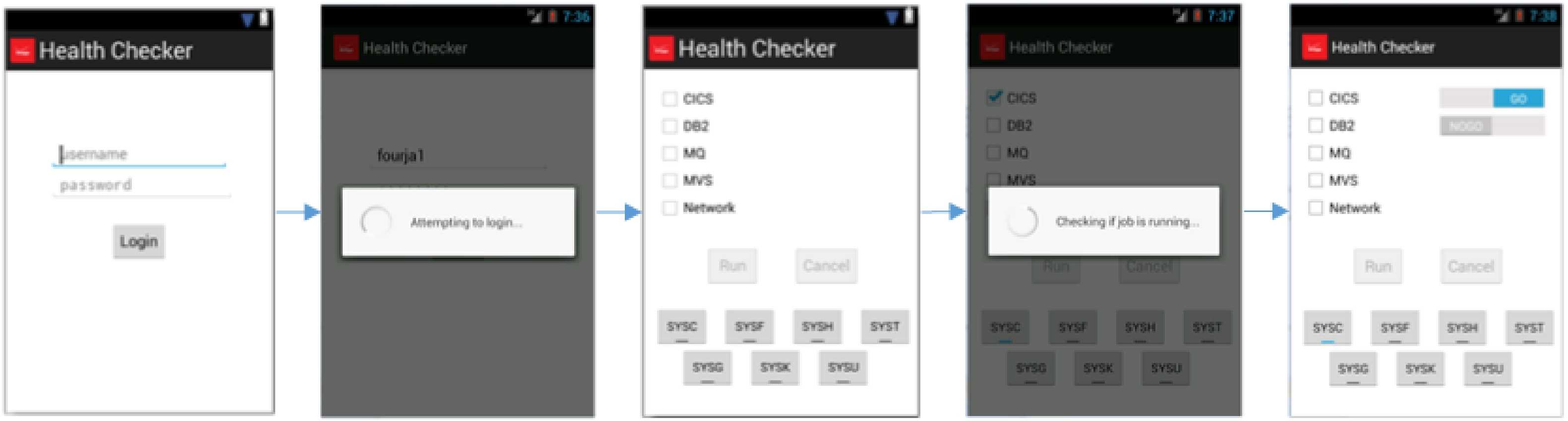
How does the app work? (Cont'd)

- Health Checks
  - Check if health check is running or has run recently
    - If it is, we will piggy back off the running job
    - If run recently, will use those results and not run again
    - Remember, lots of people wanting updates
  - If it is not, submit new job

- Results
  - Show a simple GO/NOGO
  - Either everything is fine, or something is wrong
  - Allows for quick status while waiting for on-call



Mobile Workflow



Attempting to login...

Checking if job is running...

Logon Screen

Logon Processing

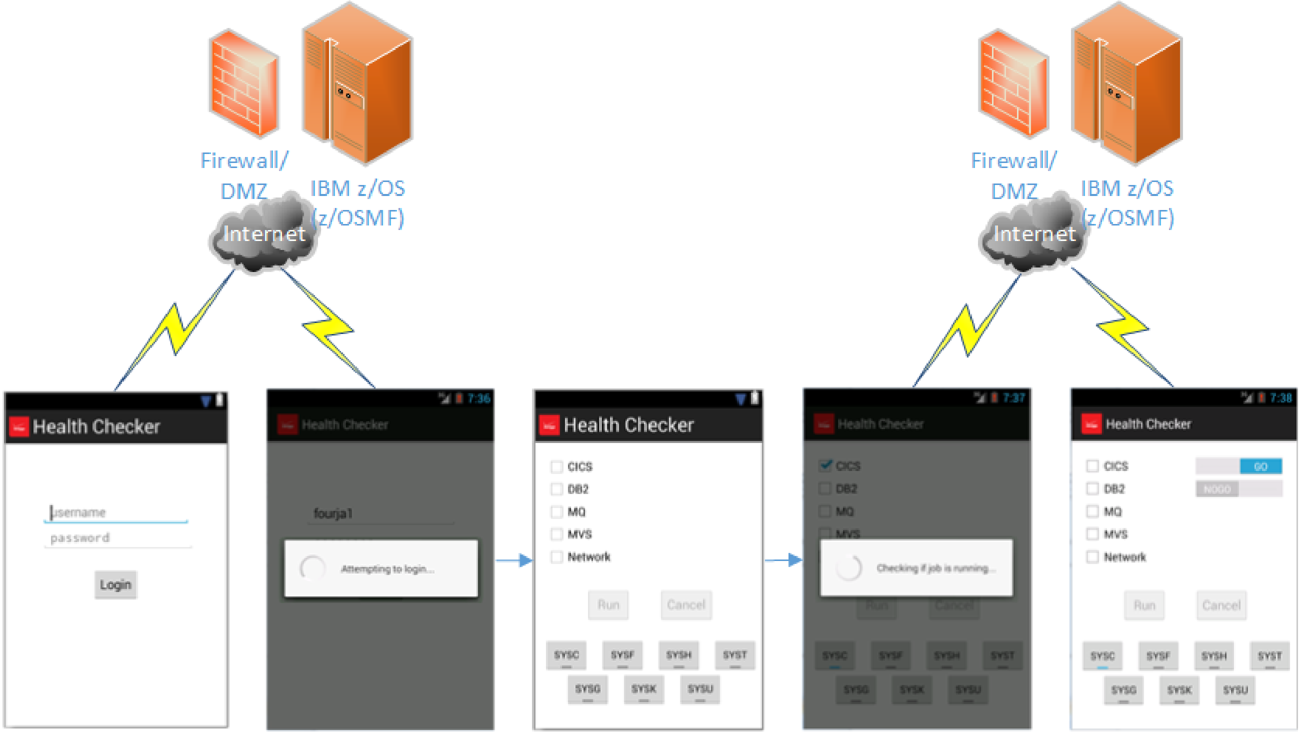
Main Menu
(Health Checks/Systems)

Job(s) Running

Results

General Flow

Shows high-level descriptions for each Activity.



Username and password sent to z/OSMF and authenticated with RACF

After login, z/OSMF sends back LTPA2 Token for use in later authentication

User selects the health check(s) he wants to run and selects a system(LPAR) and taps Run

App submits job details as JSON, checks if job has run recently/is still running, if it is, will use those results, if not will submit new job to MF, via z/OSMF

Results are received as JSON or Plain Text and parsed to see if the job passed or failed and displays results

Detailed Flow

Shows each Activity purpose and description.

Why did we use REST/JSON?

- SOAP is dying (don’t fool yourselves, it is!!)
- SOAP is complicated to use/setup/consume
- JSON is the new standard being used
- Widely accepted and easy to use
  - SDK’s are supporting it
    - Android is very well supported
    - iOS is just called something different, but supported
  - Use across OS’s and devices including web pages
    - JSON remains the same regardless
- Lightweight!!

What development approach did we take?

- Tools, SDK, etc.
  - Xcode
    - Need a Macintosh
    - Need developer account (Enterprise for in-house apps)
  - Android Developer Toolkit (Eclipse)
    - Use PC
    - No developer account

What were our limitations?

- Batch jobs only
- No true SDSF output
- This has changed
  - TSO/E address space services
  - z/OS data set and file REST interface

What problems did we have?

- Certificate handling
  - Self signed certificates CA not recognized by SDKs
- Internal proxy during development
  - Needed a way to check and auto login if found
- Integration with EMP Team
  - Our internal team with processes in place to allow red to green zone traffic securely
- Lack of testers
  - Needed more people with wider variety of devices and OS versions

Things to look out for

- Security
- MITM (Man-In-The-Middle) Attacks
- Version Control
- Source Code repository

Things to look for (Cont'd)

- Security
  - MITM Attacks
    - Use certificates!! Verify, verify, verify!

  - Don't "cheat" yourselves (or your company)
    - Do not bypass certificates
    - Check server names against certificates
    - Figure the code out, or leave yourselves open
    - HTTPS does not mean anything if you bypass certificates

Things to look for (Cont’d)

- Version Control
  - Makes sure app is current
    - Allows you to know what users have
    - Keeps help desk support in sync
  - Keeps security in check
    - When bugs are found, force the next upgrade
  - All users on same code base
    - Helps troubleshooting and communication between team members when using the exact same app version

Things to look for (Cont’d)

- Source Code repository
  - Github
    - Not controlled locally, usually
  - Xcode
    - Stored on local machine, susceptible to hard drive crash
  - Saving versions on hard drive
    - Archiving
    - Requires a lot more space

What else could we have done?

- Worklight (IBM MobileFirst)
  - Prevents double work (saves time = saves money)
    - Creates a version for iOS and Android
    - Uses same code base
  - Requires Linux or Solaris
  - There are advantages especially if your company does not have a process already in place (red to green zone)

What are our future plans?

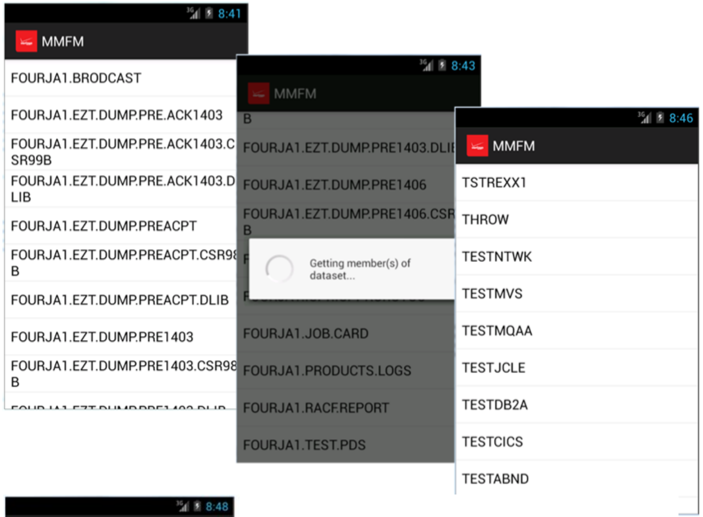
- z/OS Connect
  - Installed but not configured

- Further exploit of z/OSMF
  - Using other REST services

- Maybe Worklight
  - We have most functionality already in place
  - Would have to maintain 2 code bases without

What new apps are in our pipeline?

- Mobile Mainframe Filemanager
  - Display all DSNs
  - Allow editing of members (JCL)
  - Allow submitting of jobs



8:41 MMFM

FOURJA1.BRODCAST
FOURJA1.EZT.DUMP.PRE.ACK1403
FOURJA1.EZT.DUMP.PRE.ACK1403.CSR99B
FOURJA1.EZT.DUMP.PRE.ACK1403.DLIB
FOURJA1.EZT.DUMP.PREACPT
FOURJA1.EZT.DUMP.PREACPT.CSR98B
FOURJA1.EZT.DUMP.PREACPT.DLIB
FOURJA1.EZT.DUMP.PRE1403
FOURJA1.EZT.DUMP.PRE1403.CSR98B

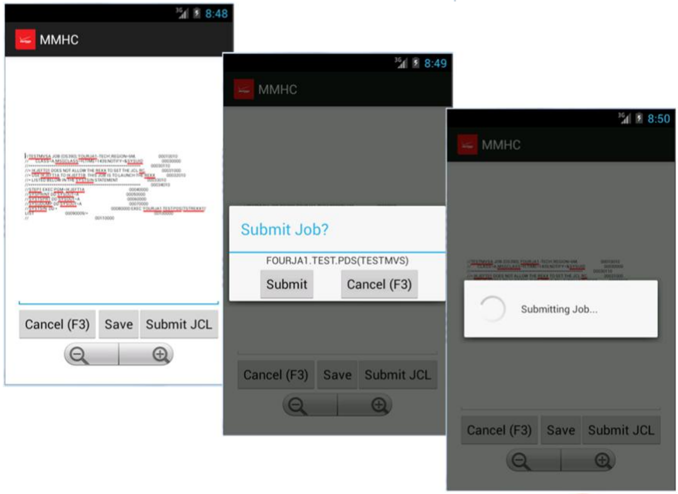
8:43 MMFM

FOURJA1.EZT.DUMP.PRE1403.DLIB
FOURJA1.EZT.DUMP.PRE1406
FOURJA1.EZT.DUMP.PRE1406.CSRB
Getting member(s) of dataset...
FOURJA1.JOB.CARD
FOURJA1.PRODUCTS.LOGS
FOURJA1.RACF.REPORT
FOURJA1.TEST.PDS

8:46 MMFM

TSTREXX1
THROW
TESTNTWK
TESTMVS
TESTMQAA
TESTJCLE
TESTDB2A
TESTCICS
TESTABND

- Mobile Mainframe Quick access
  - Run frequent used jobs quickly
  - Edit JCL
  - Submit jobs



8:48 MMHC

8:49 MMHC

Submit Job?
FOURJA1.TEST.PDS(TESTMVS)

8:50 MMHC

Submitting Job...

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

Going forward, things to think about…

- What OS to develop for?
  - iOS
  - Android
  - Windows Mobile
  - Blackberry

- Phone, tablet, or both?
  - What data is being displayed
  - Where is the user base using the data (wi-fi, cellular)
  - How many people have which kind of device
  - Become familiar with your GUI options
  - Think through your GUI layout (wireframe it!!!)

Questions???

Contact Information

- Ron Piracci

ron.piracci@verizonwireless.com

Work: 480-783-5157

- Jason Fournier

jason.fournier@verizonwireless.com

Work: 949-286-7896

And of course, some links…

- IBM MobileFirst Platform Foundation 6.3 requirements
  - http://www-969.ibm.com/software/reports/compatibility/clarity-reports/report/html/osForProduct?deliverableId=1404132386734
- http://www-01.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_ca/0/877/ENUSZP14-0280/index.html&lang=en&request_locale=en