MASHING-UP MULTIPLE DATA STREAMS - LESSONS LEARNED USING THINGWORX AND KEPWARE

Yale Evans
Lead Application Analyst

“

How to deploy successful mashups while avoiding the pitfalls of improper setup and configuration of Kepware and Thingworx. The do’s and don’ts to make your life easier.

”

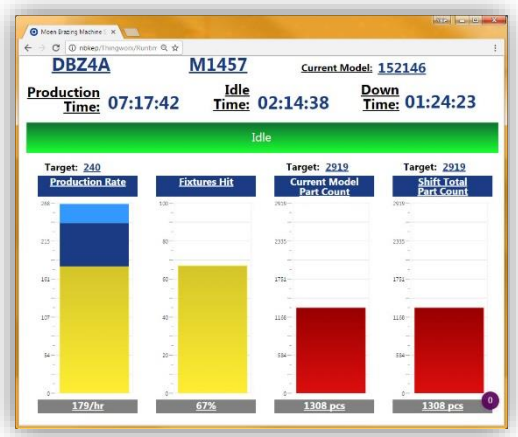
DATA FLOW

PLC

0010101001010101101010010101010101101010110101010101010101011...

KEPServerEX

thingworx



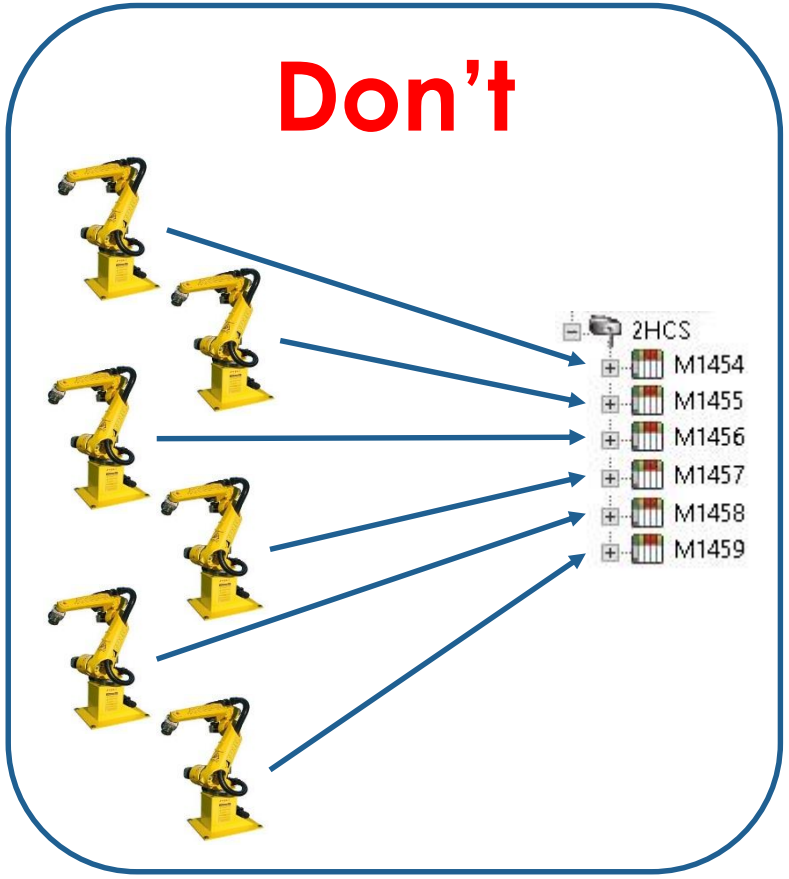
MASHUP

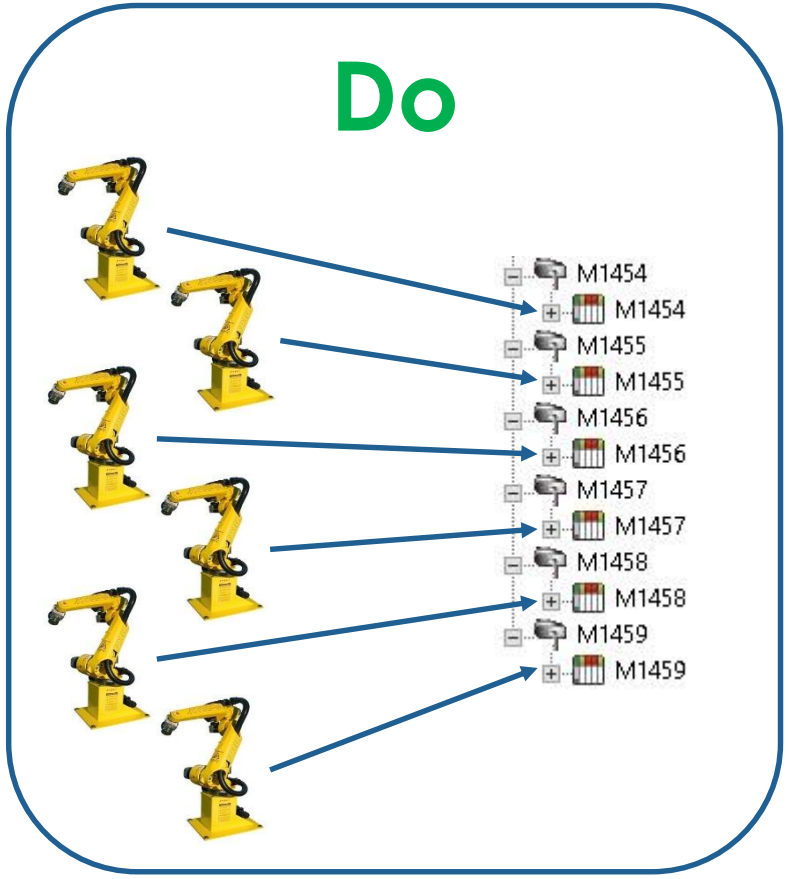
1. KEPWARE SETUP

INSTALLING AND CONFIGURING KEPSERVER EX

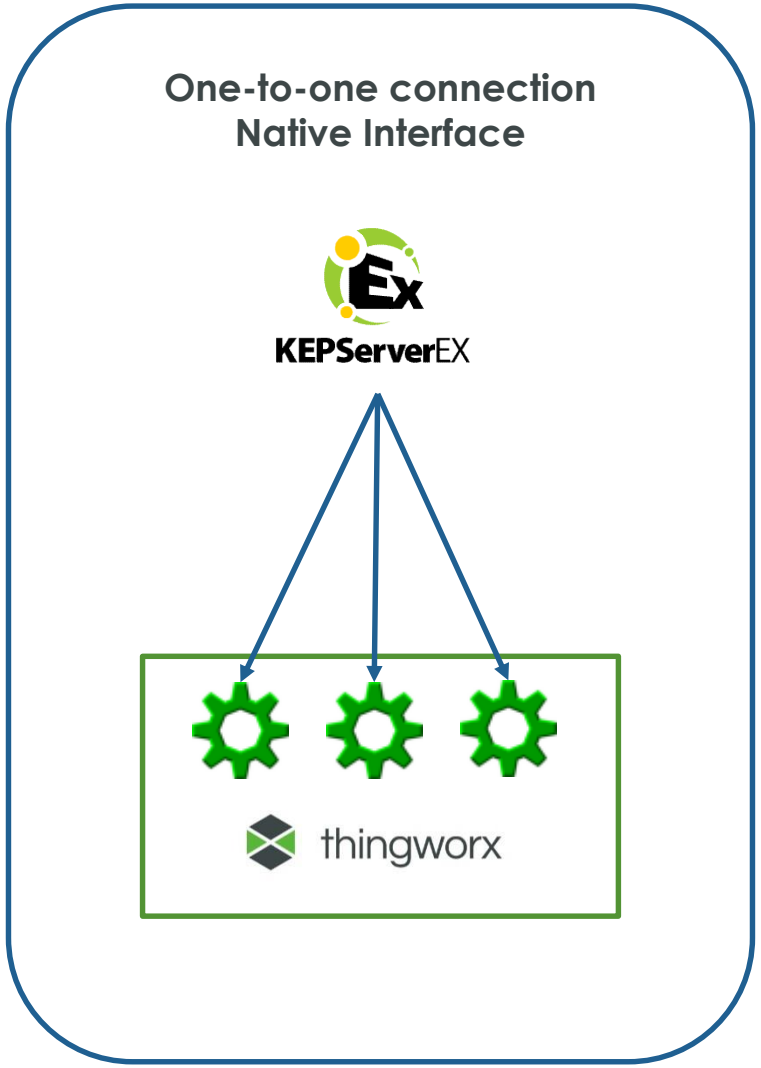
CHANNELS AND DEVICES

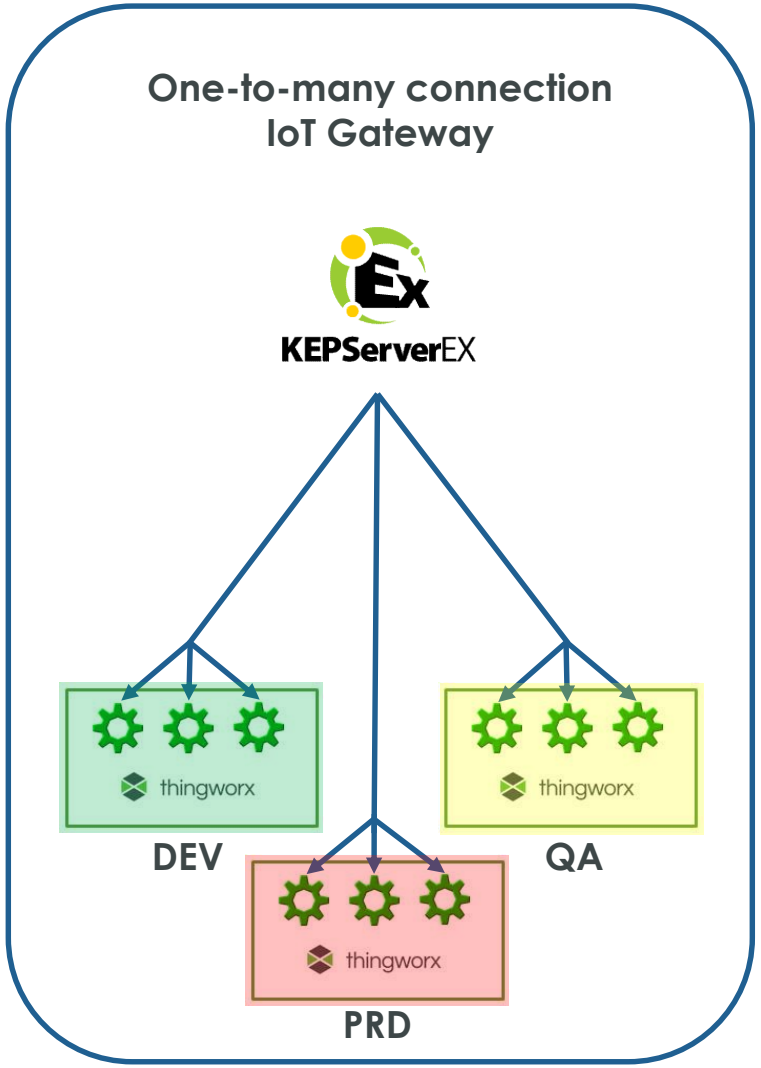
Set up a channel for EVERY DEVICE to enable parallel data streaming (* for fast networks)





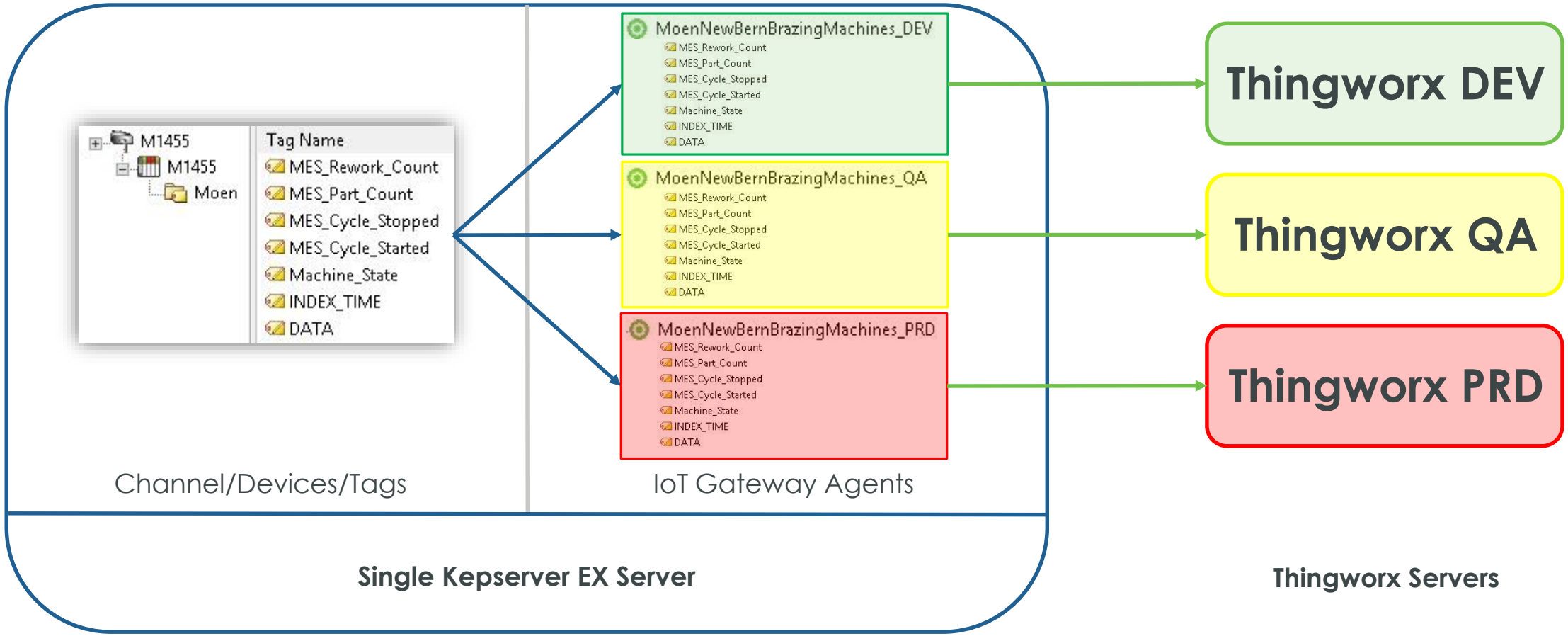
CONNECTING KEPSERVER AND THINGWORX





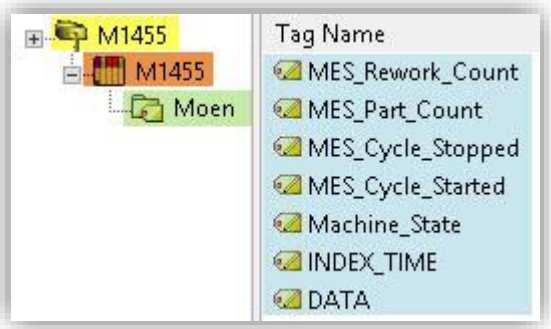
IoT GATEWAY CONFIGURATION

- IoT Gateway functionality allows same data to multiple Thingworx servers
- Beneficial for development and testing

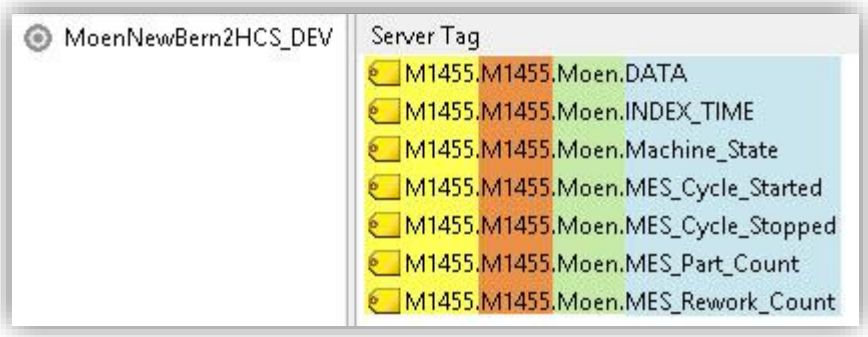


TAG NAMING IN KEPSERVER EX

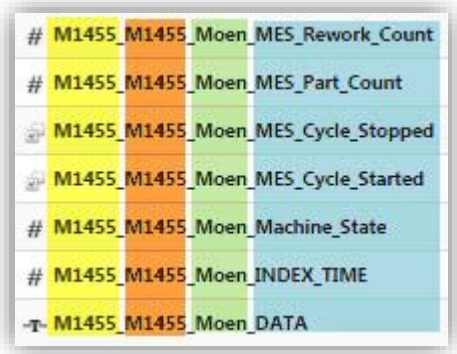
- Thingworx uses "channel_device_tag-group_tag-name" for property names
- Keep all names short
- Group tags logically



Kepware Channel Config



Kepware IoT Gateway Config



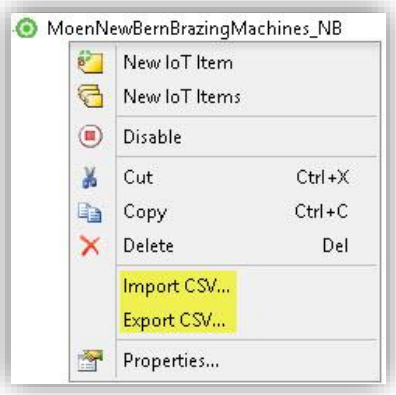
Thingworx Properties

Legend:

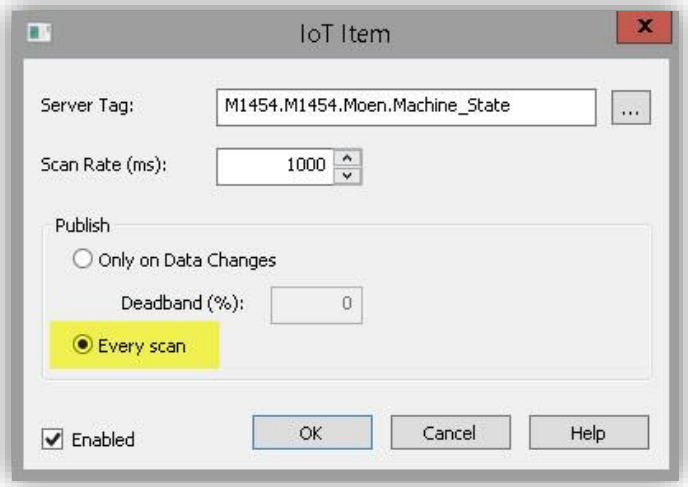
Channel
Device
Tag Group
Tag Name

OTHER TAG MAINTENANCE ADVICE

- Use export/import to CSV for bulk changes



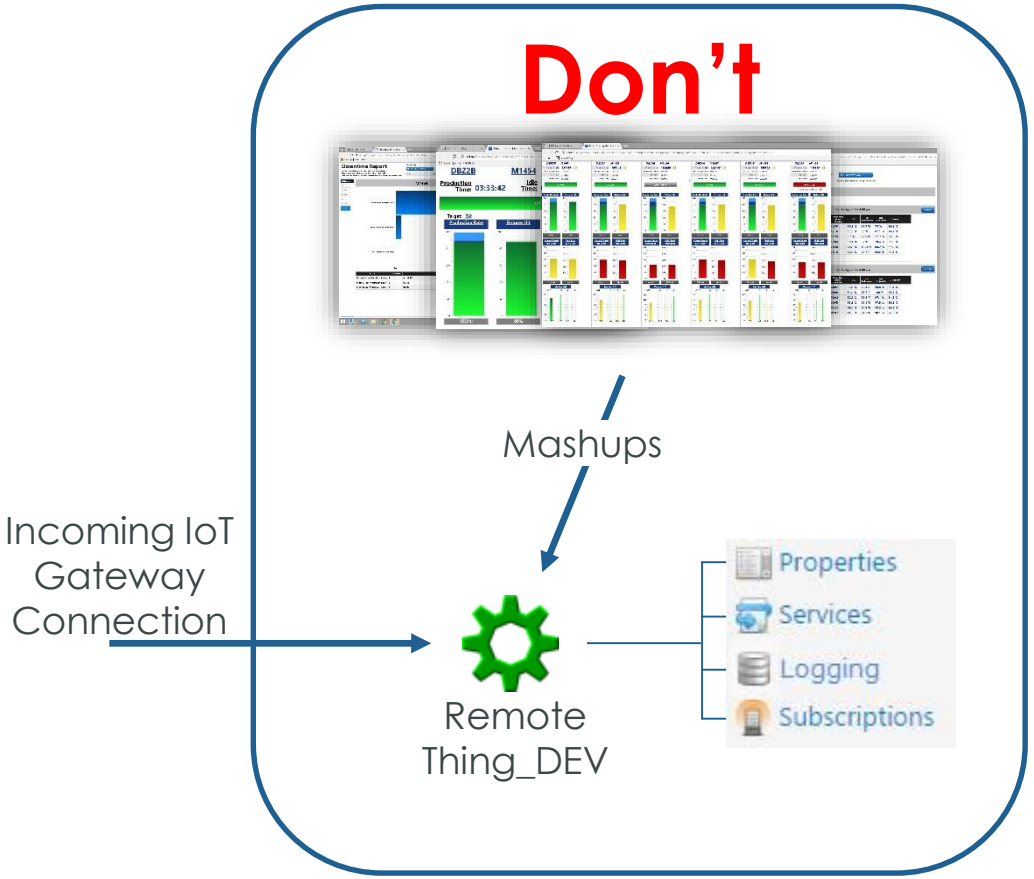
- FIRST time, set publish to "Every Scan" then "Only on Data Changes"

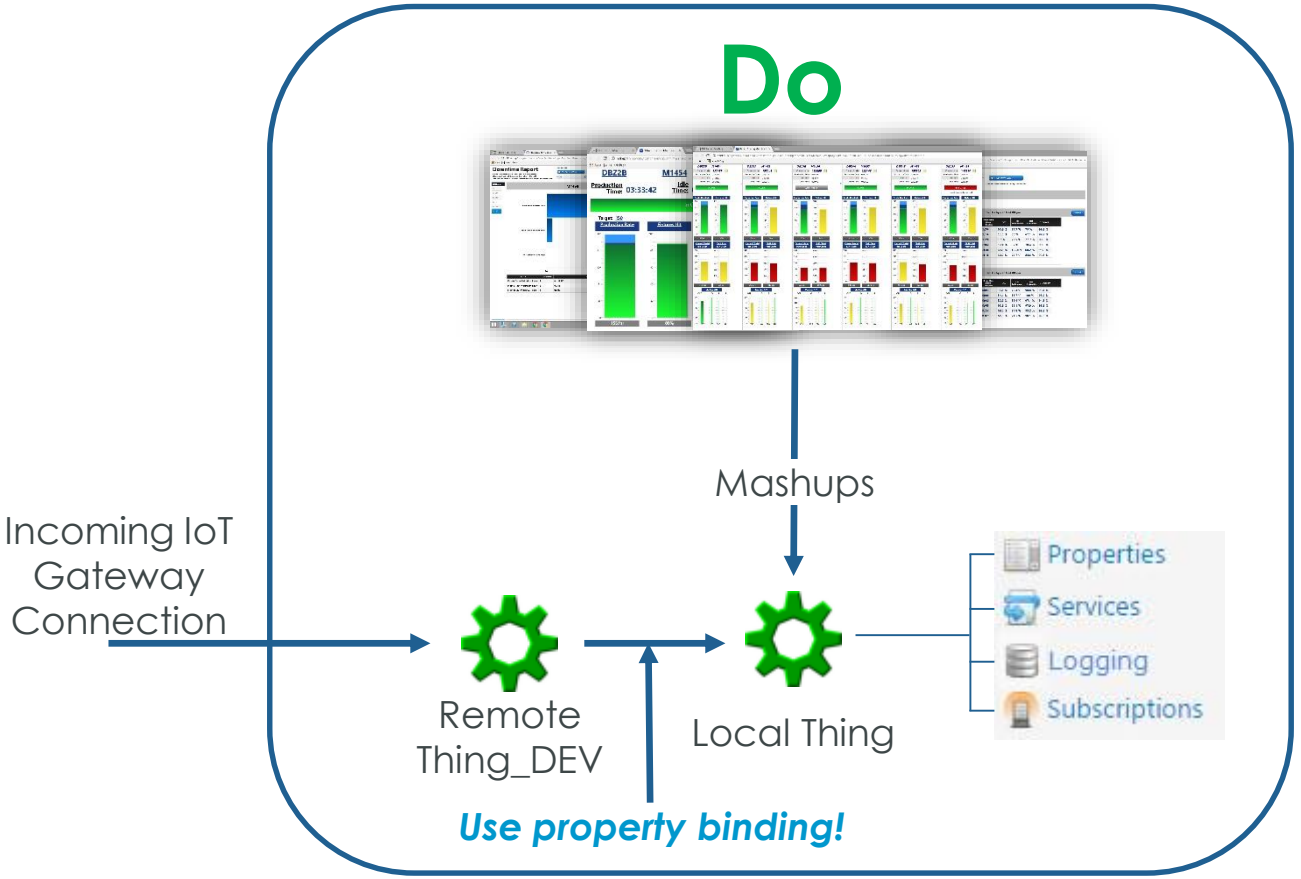


2. THINGWORX CONFIGURATION

IOT GATEWAY/THINGWORX MAPPING

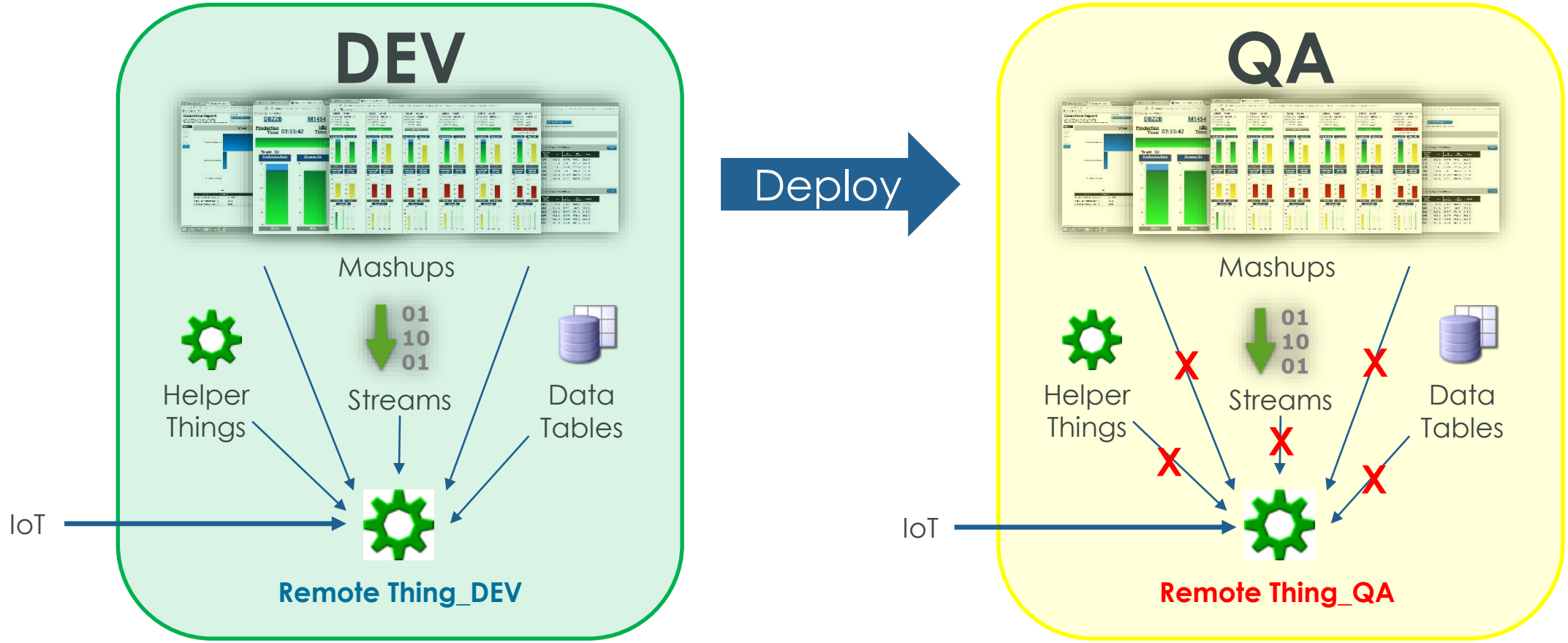
- Do **NOT** use *remote* 'Thing' mapped to Kepware as a reference
- **DO** use *local* 'Thing' bound to remote Thing





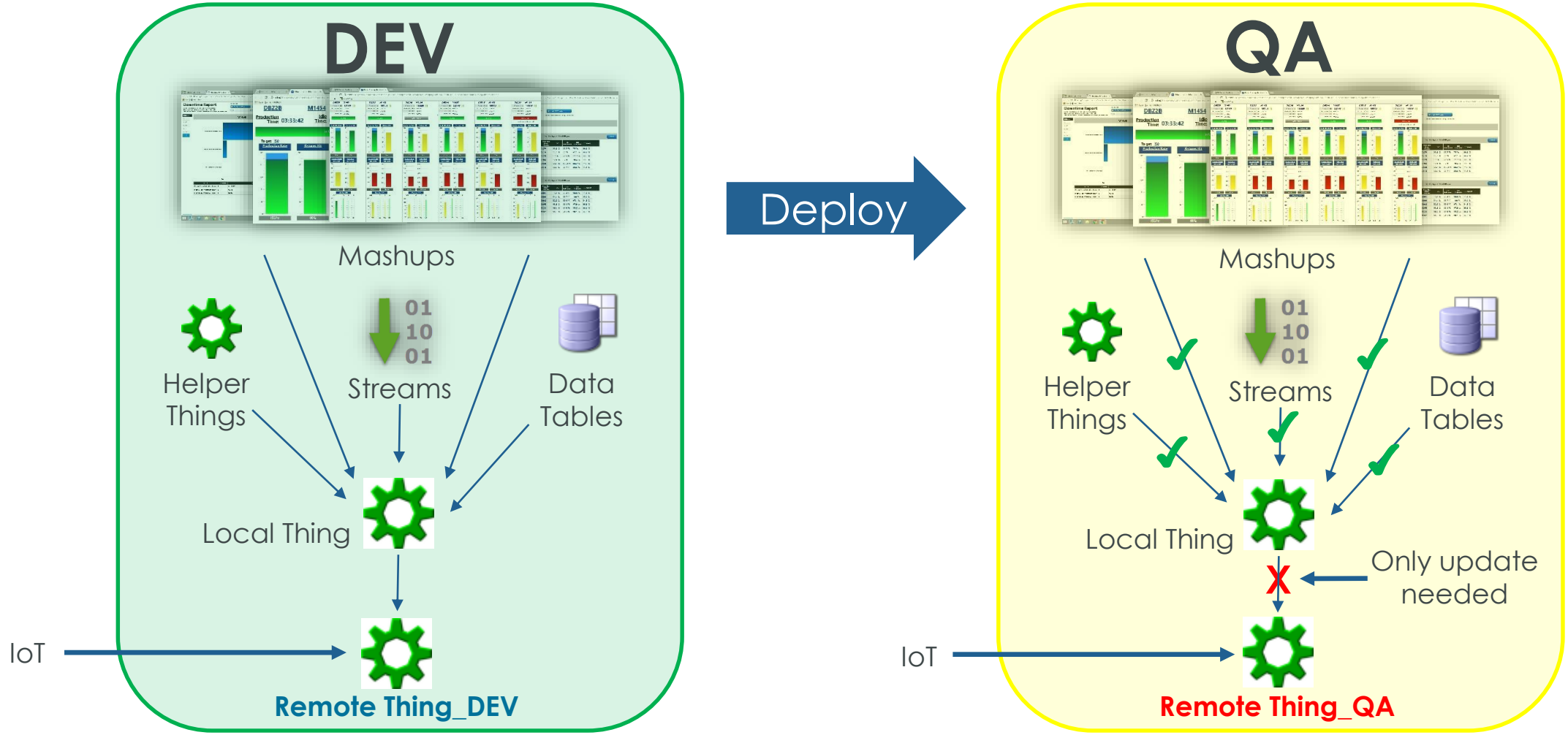
WHY IS FIRST METHOD BAD?

- Remote thing will be named differently on each server
- Mismatches will occur when deploying



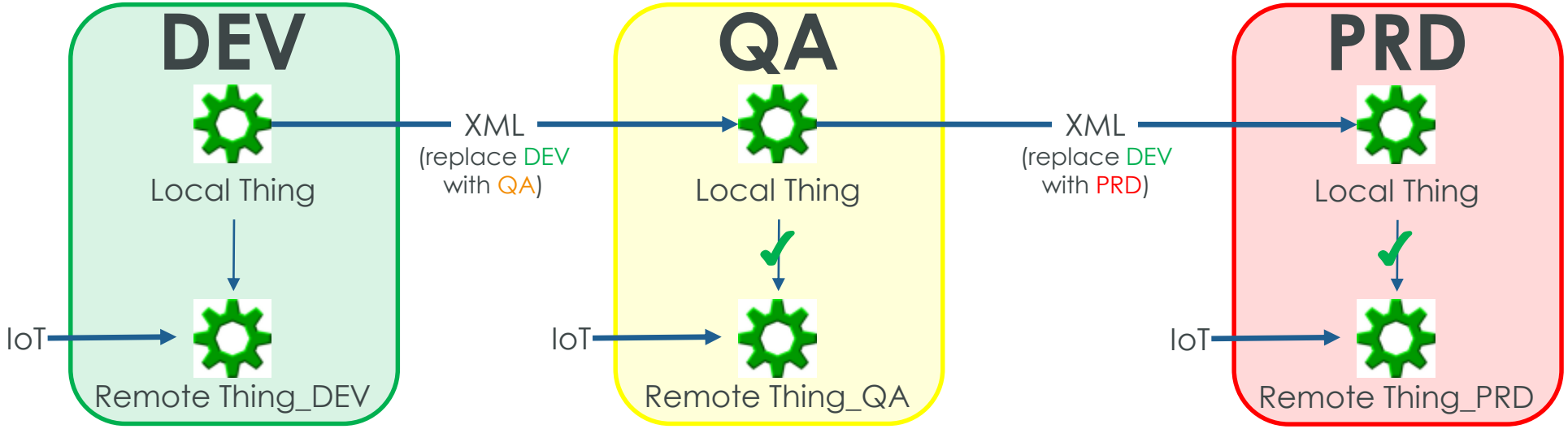
WHY IS SECOND METHOD BETTER?

- All links point to same entity name



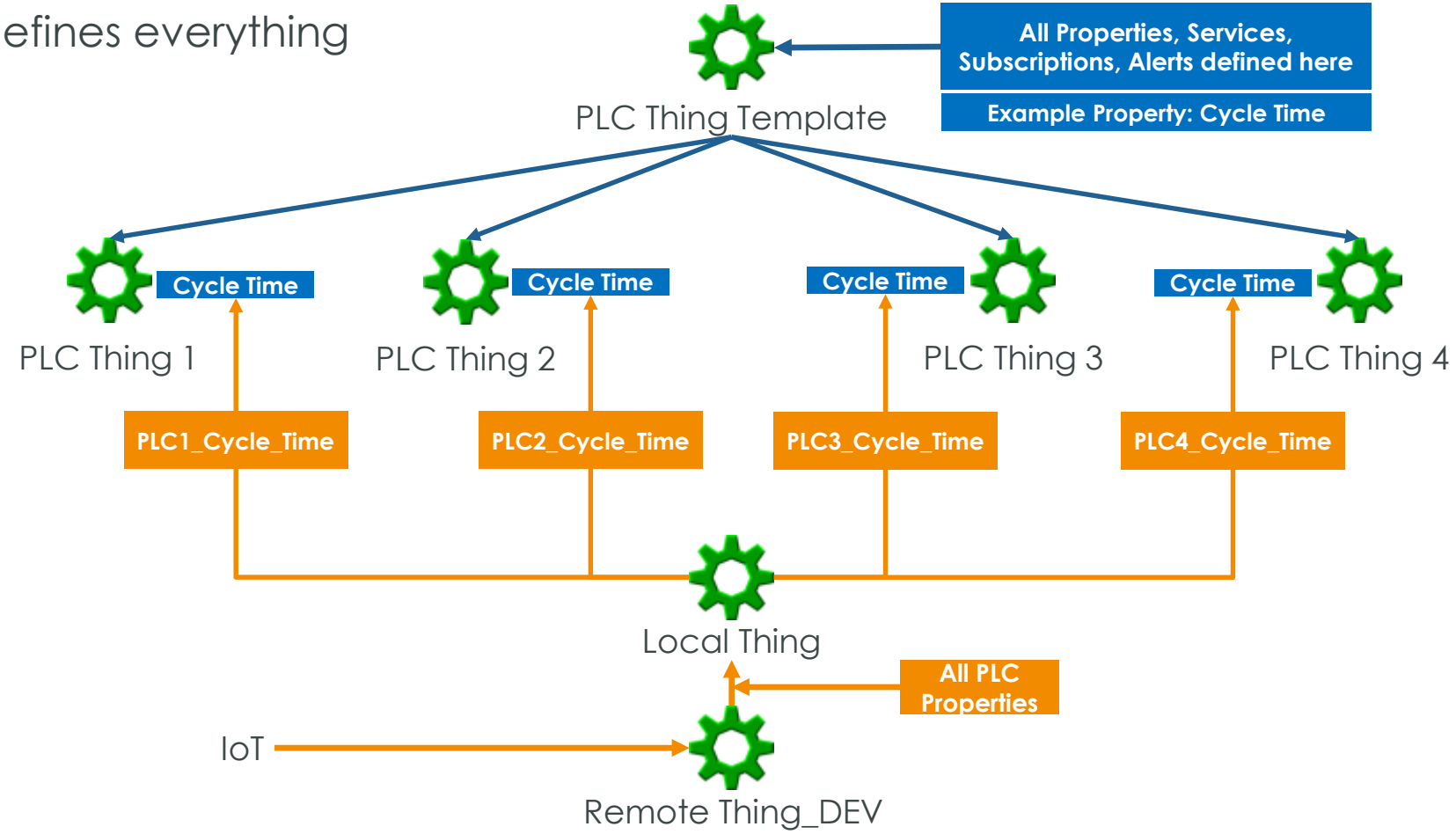
REDUCING DEPLOYMENT COMPLEXITY

- Using a local, commonly-named Thing means only have to update one XML file during deployment
- Edit XML file before import and replace remote thing name



ENTITY FOR EACH DEVICE

- Data for all devices in single Thingworx entity
- Create entity for each device using property binding
- Thing Template defines everything



PERFORMING CALCULATIONS

- Raw data often needs to be fed through algorithms to get meaningful output

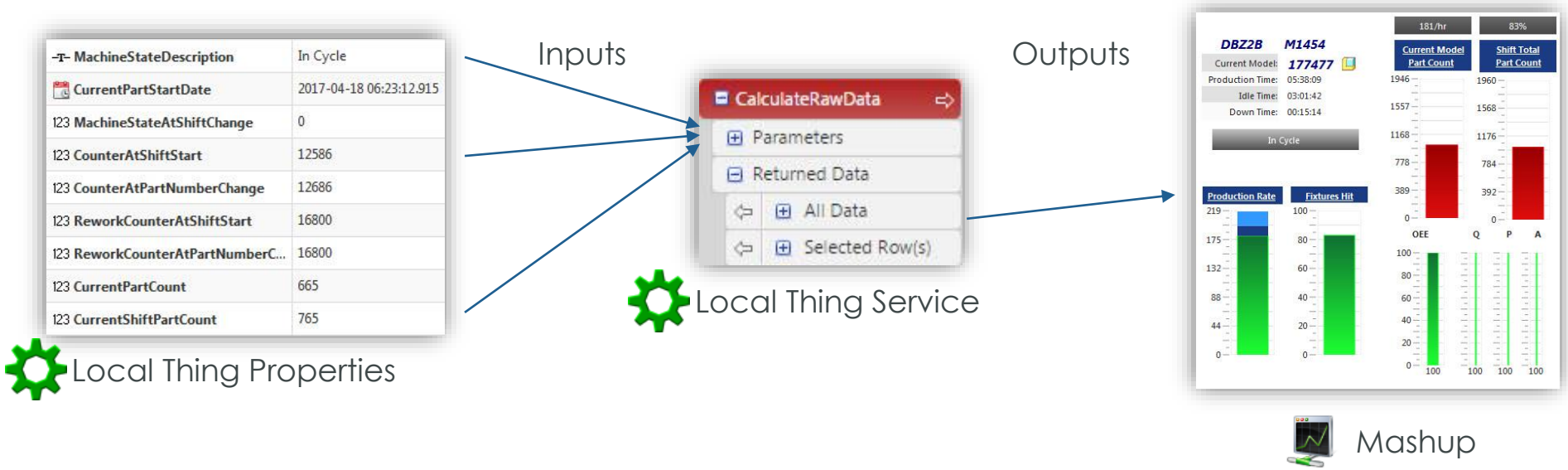
- Start Time
- Part Count
- Cycle Time
- Good Parts
- Bad Parts
- Fault Code

$$f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L}\right)$$

- Actual Rate
- Target Rate
- Run Time
- Down Time
- Faults/Hour
- Total Faults

ANOTHER MISTAKE – RUN CALCULATIONS AT MASHUP TIME

- Initial attempt:
  - Create a service to perform calculations
  - Service uses input from raw data
  - Bind service output to mashup widgets



RUNNING CALCULATIONS AT MASHUP TIME IS BAD

- OK when one user is viewing
- Every additional user runs same service
- Multiplied by auto-refresh rate
- What if more than one service?

1 user x 15-second refresh rate x 1 service = 4 services run/minute

5 users x 15-second refresh rate x 2 services = 40 services run/minute

10 users x 15-second refresh rate x 3 services = 120 services run/minute

O(N)

BETTER: RUN CALCULATIONS USING A SUBSCRIPTION

- Use a scheduler thing to run the calculations 
- Subscription runs service on scheduled event 
- Store results in properties 

- Use GetProperties service to display values in mashups 

Now one service *only* runs once every scheduled period and all mashups use the same output data

SCHEDULED SUBSCRIPTION SEQUENCE

Local Thing Scheduler

Every 15 seconds

Local Thing Subscription

```
var result = me.CalculateRawData();
for (var prop in result) {
            me[prop] = result[prop];
}
```

Input

Output

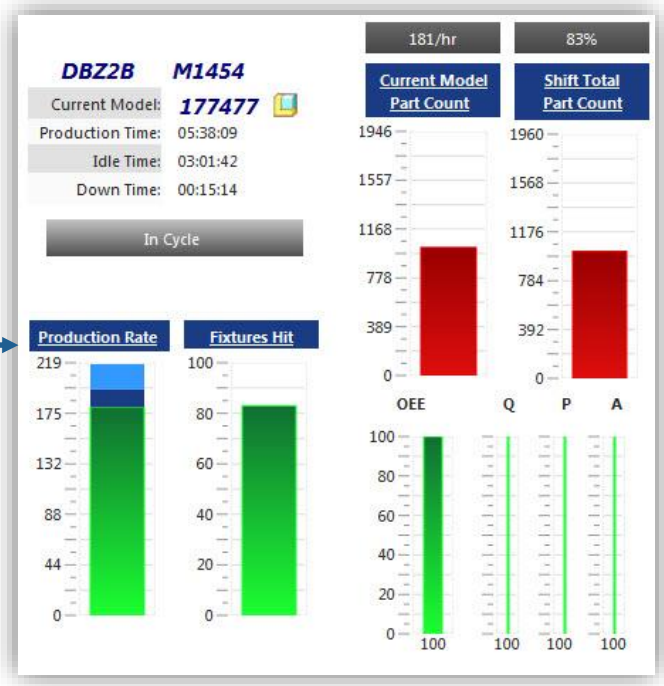
Local Thing Properties

MachineStateDescription	In Cycle
CurrentPartStartDate	2017-04-18 06:23:12.915
123 MachineStateAtShiftChange	0
123 CounterAtShiftStart	12586
123 CounterAtPartNumberChange	12686
123 ReworkCounterAtShiftStart	16800
123 ReworkCounterAtPartNumberC...	16800
123 CurrentPartCount	665
123 CurrentShiftPartCount	765
123 ProductionRate	181
123 TargetPartRate	196
123 TheoreticalPartRate	22
123 FixtureHitRate	83
123 CurrentModelPartCount	1020
123 TheoreticalPartCount	1946
123 ShiftTotalPartCount	1020

Raw Data

Calculated Data

MASHUPS NOW JUST READ CALCULATED PROPERTIES



SUBSCRIPTION STRATEGY

- Service called by subscription returns JSON
- One-to-one mapping between JSON and Thing properties
- Super simple and reduces maintenance
- Subscription code never needs changing

Subscription

```
var result = me.CalculateRawData();
for (var prop in result) {
            me[prop] = result[prop];
}
```

Service

Service Name	Service Type	Inputs	Output
CalculateRawData	Local (JavaScript)		result

JSON Output

```
var result = {
   ProductionRate : 181,
   TargetPartRate : 196,
   TheoreticalPartRate : 22,
   FixtureHitRate : 83,
   CurrentModelPartCount : 1020,
   TheoreticalPartCount : 1946,
   ShiftTotalPartCount : 1020
}
```

One-to-one mapping

123 ProductionRate	181
123 TargetPartRate	196
123 TheoreticalPartRate	22
123 FixtureHitRate	83
123 CurrentModelPartCount	1020
123 TheoreticalPartCount	1946
123 ShiftTotalPartCount	1020

USE THING SHAPES

- Use thing shape for properties returned in JSON
- Especially if multiple 'Things' have same properties
- For new properties, add to service output and thing shape
- No need to maintain data shapes, info tables, or subscription code

JSON Output

```
var result = {
    ProductionRate : 181,
    TargetPartRate : 196,
    TheoreticalPartRate : 22,
    FixtureHitRate : 83,
    CurrentModelPartCount : 1020,
    TheoreticalPartCount : 1946,
    ShiftTotalPartCount : 1020
    RunningAverage : 123.5
}
```

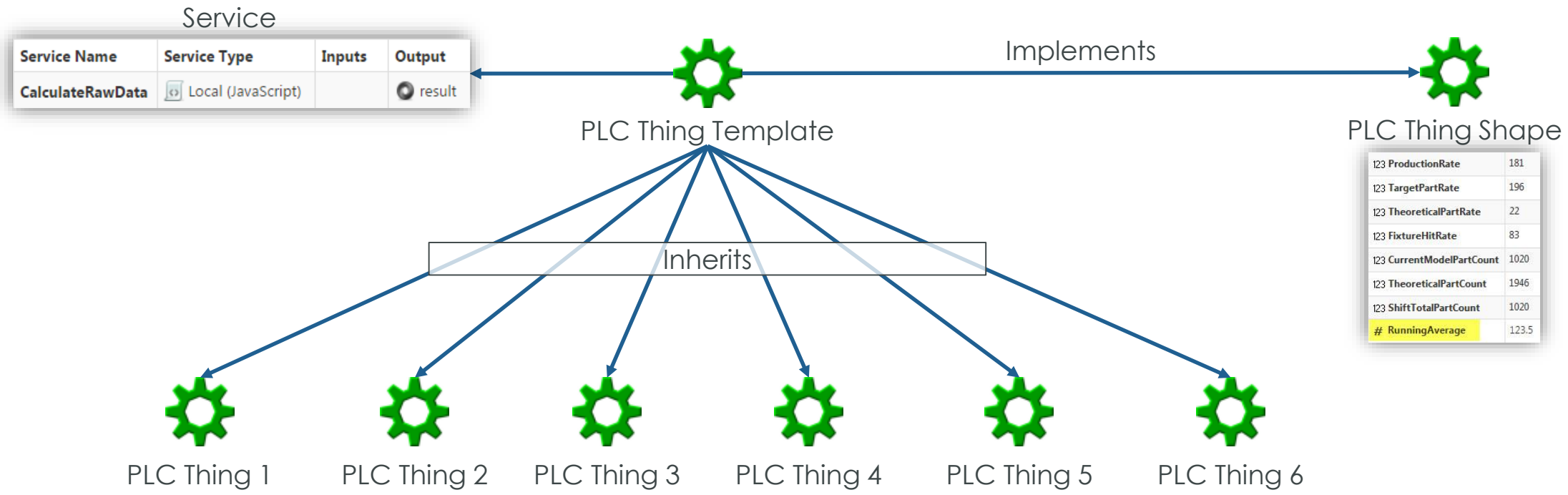
1. **Add here**
2. **Add here**
3. **Done!**

Thing Shape Properties

123 ProductionRate	181
123 TargetPartRate	196
123 TheoreticalPartRate	22
123 FixtureHitRate	83
123 CurrentModelPartCount	1020
123 TheoreticalPartCount	1946
123 ShiftTotalPartCount	1020
# RunningAverage	123.5

THING SHAPE IMPLEMENTATION

- Only need to update one Thing Template & one Thing Shape
- All inherited Things updated automatically



3. MASHUPS

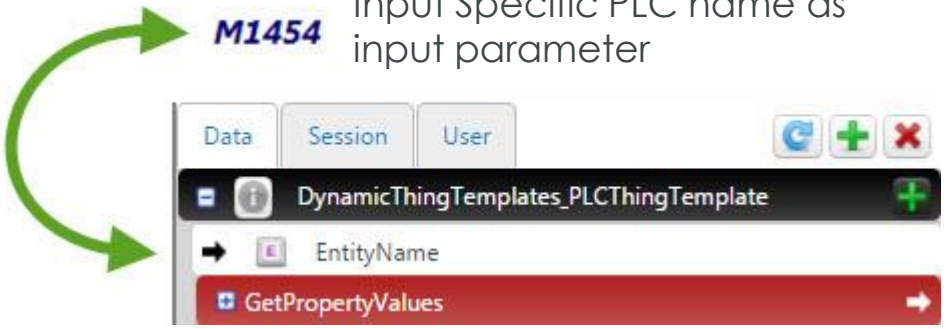
DESIGNING COMMON MASHUPS

Problem:

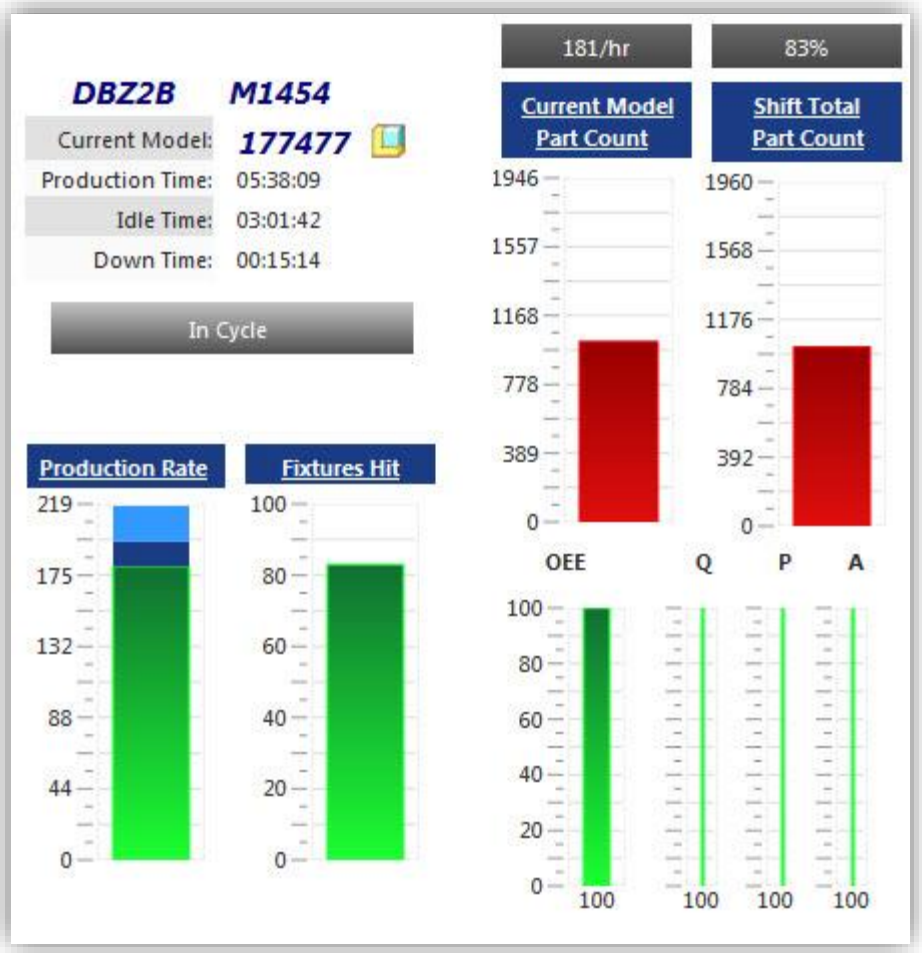
One mashup, several machines

Solution:

Use dynamic thing templates as data sources



Mashup: **CurrentJobRun**

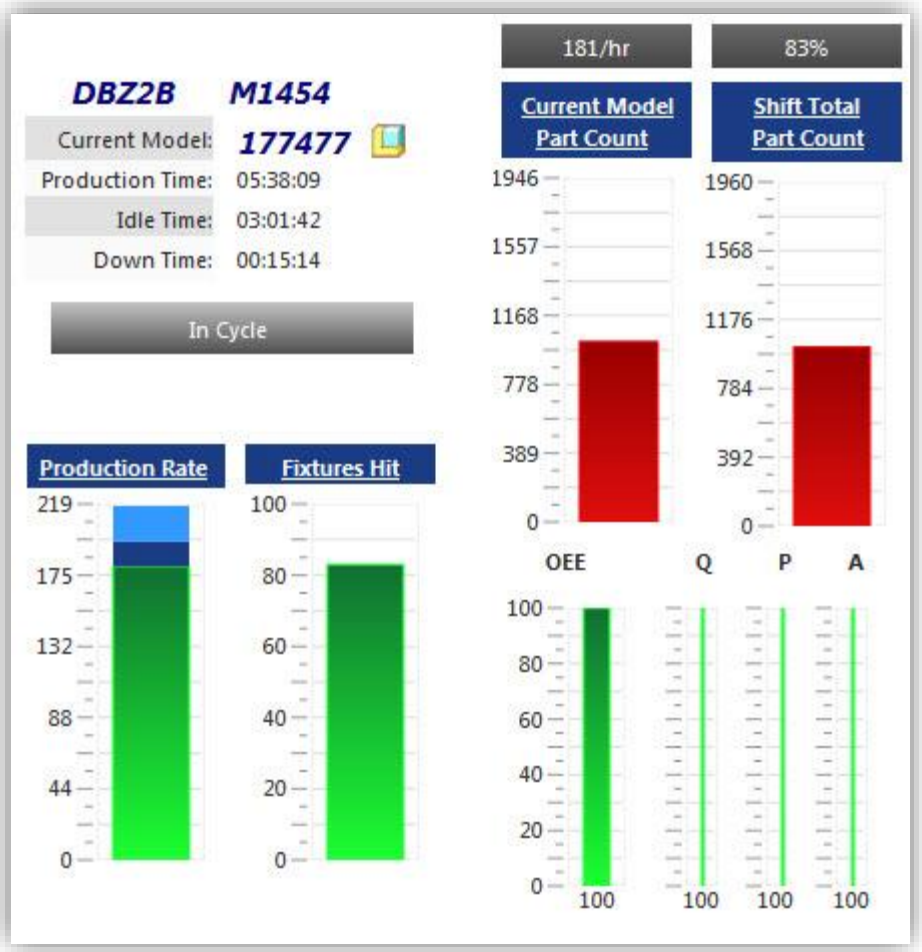


DESIGNING COMMON MASHUPS

Problem:

- Mashup shared by all roles
- Master exists for each role
- Can only configure one master

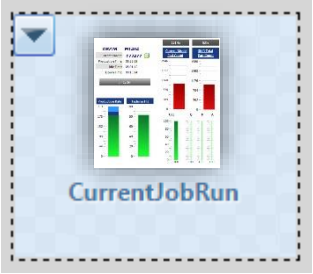
Mashup: **CurrentJobRun**



USE NESTED MASHUPS FOR DIFFERENT ROLES

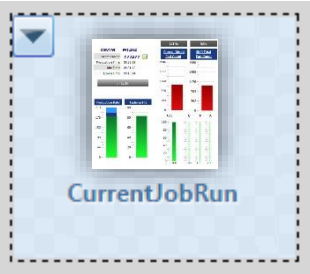
- Define **outer** mashups for different roles
- Each outer mashup has ***same* inner** mashup
- Outer mashups points to appropriate **master**

Mashup:
CurrentJobRun*Operator*



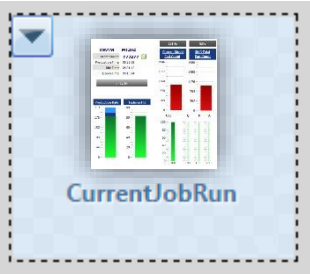
Master:
***Operator* Master**

Mashup:
CurrentJobRun*Maintenance*



Master:
***Maintenance* Master**

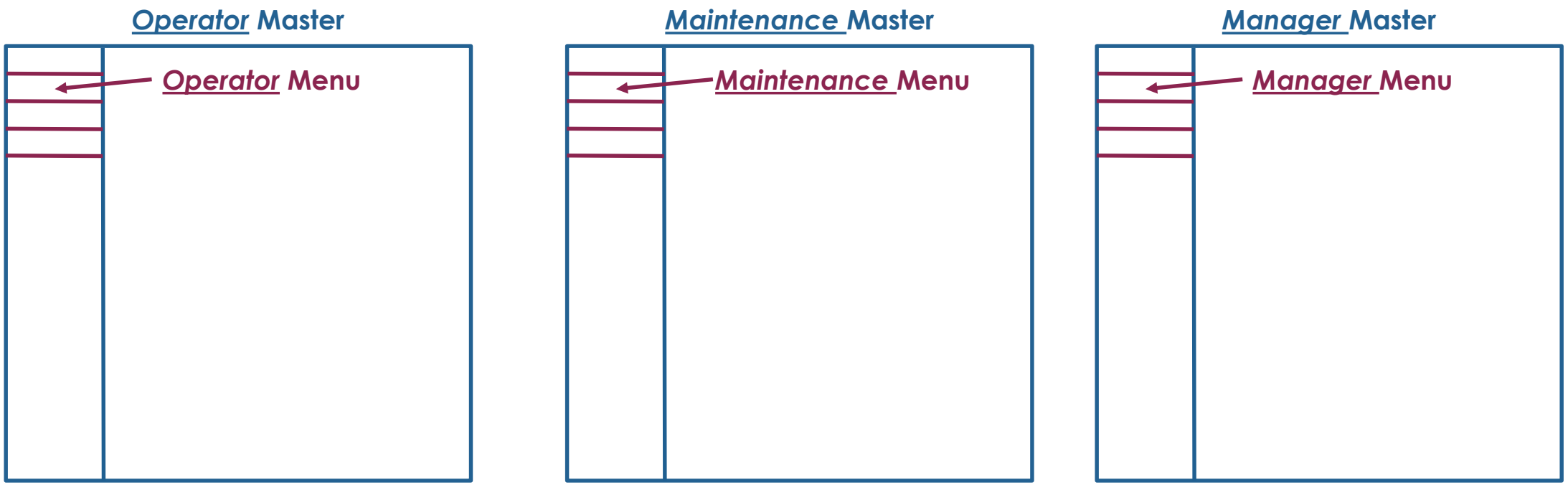
Mashup:
CurrentJobRun*Manager*



Master:
***Manager* Master**

MASTER MENUS POINT TO OUTER MASHUPS

- Each role has a **master**
- Each role has a **menu**



MENUS POINT TO OUTER MASHUPS

***Operator* Menu**

Title	Link
Current Job Run	CurrentJobRunOperator

***Maintenance* Menu**

Title	Link
Current Job Run	CurrentRunMaintenance

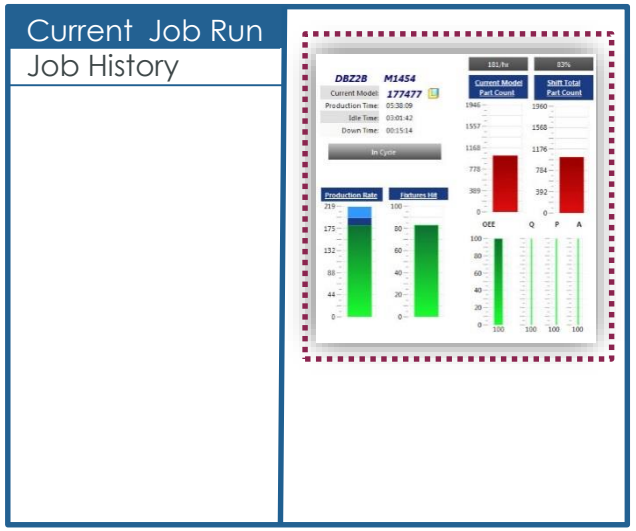
***Manager* Menu**

Title	Link
Current Job Run	CurrentRunManager

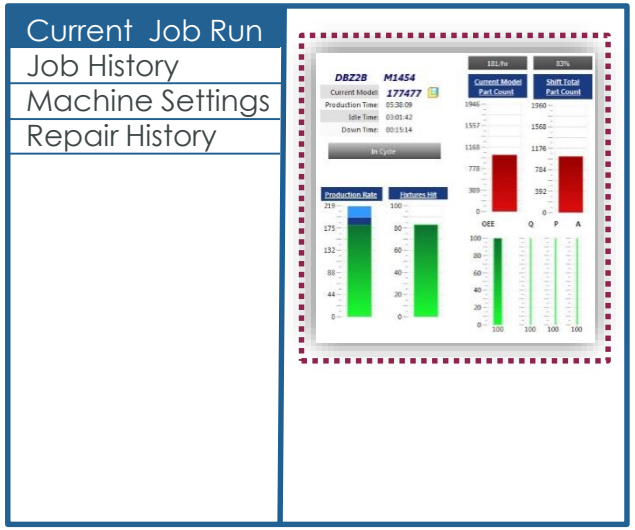
PUTTING IT ALL TOGETHER

- All roles see the same mashup
- Menus stay role-specific

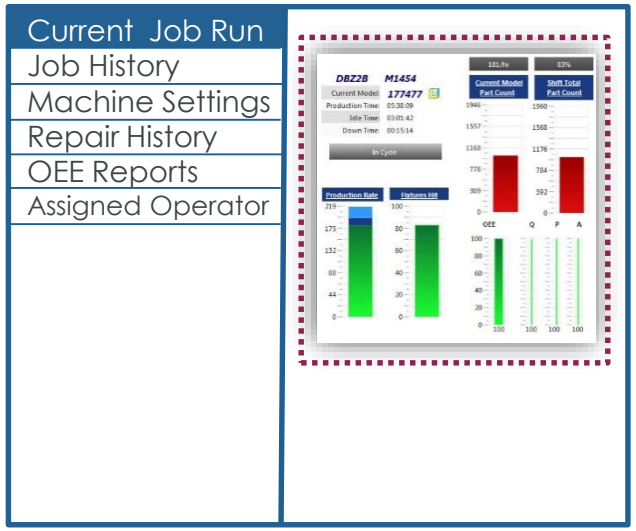
Operator will see:



Maintenance will see:



Manager will see:




Outer Mashup


Inner Mashup

4. APACHE TOMCAT

USE TOMCAT REWRITE VALVE

- Add RewriteValve to Apache Tomcat
- Add URL aliases to simplify access
- Reduce appkey management

Which URL is easier to manage and distribute?

http://thingworx.moen.com/Thingworx/Mashups/Moen+Brazing+Machine+Single+Machine+Full?appKey=3f91aab7-a88b-4e50-b9bc-33e353b61d8e&x-thingworx-session=true&MoenBrazingMachinePLCName=MoenNewbernBrazingMachinePLC_M1454

Or

http://thingworx.moen.com/m1454

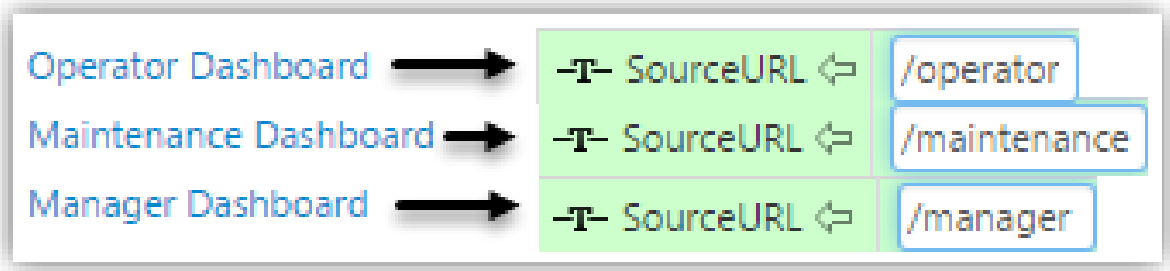
TOMCAT REWRITE VALVE

- Helpful in external launch pages
- Can be used in internal mashups

Apache Rewrite File

```
<Apache install directory>/conf/Catalina/localhost/rewrite.conf:
RewriteRule     /operator          /Thingworx/Mashups/MyApplicationOperator
RewriteRule     /maintenance       /Thingworx/Mashups/MyApplicationMaintenance
RewriteRule     /manager           /Thingworx/Mashups/MyApplicationManager
```

Internal Launch Mashup
(Link Widgets)



SUMMARY

WE WANT YOUR FEEDBACK

Please remember to complete your evaluation by selecting the session in your mobile app.

LIVEWORX 17™