OpenBMC + MRW

The Machine Readable Workbook – For A Data Driven OpenBMC

Matt Spinler
spinler@us.ibm.com
8/25/16

Agenda

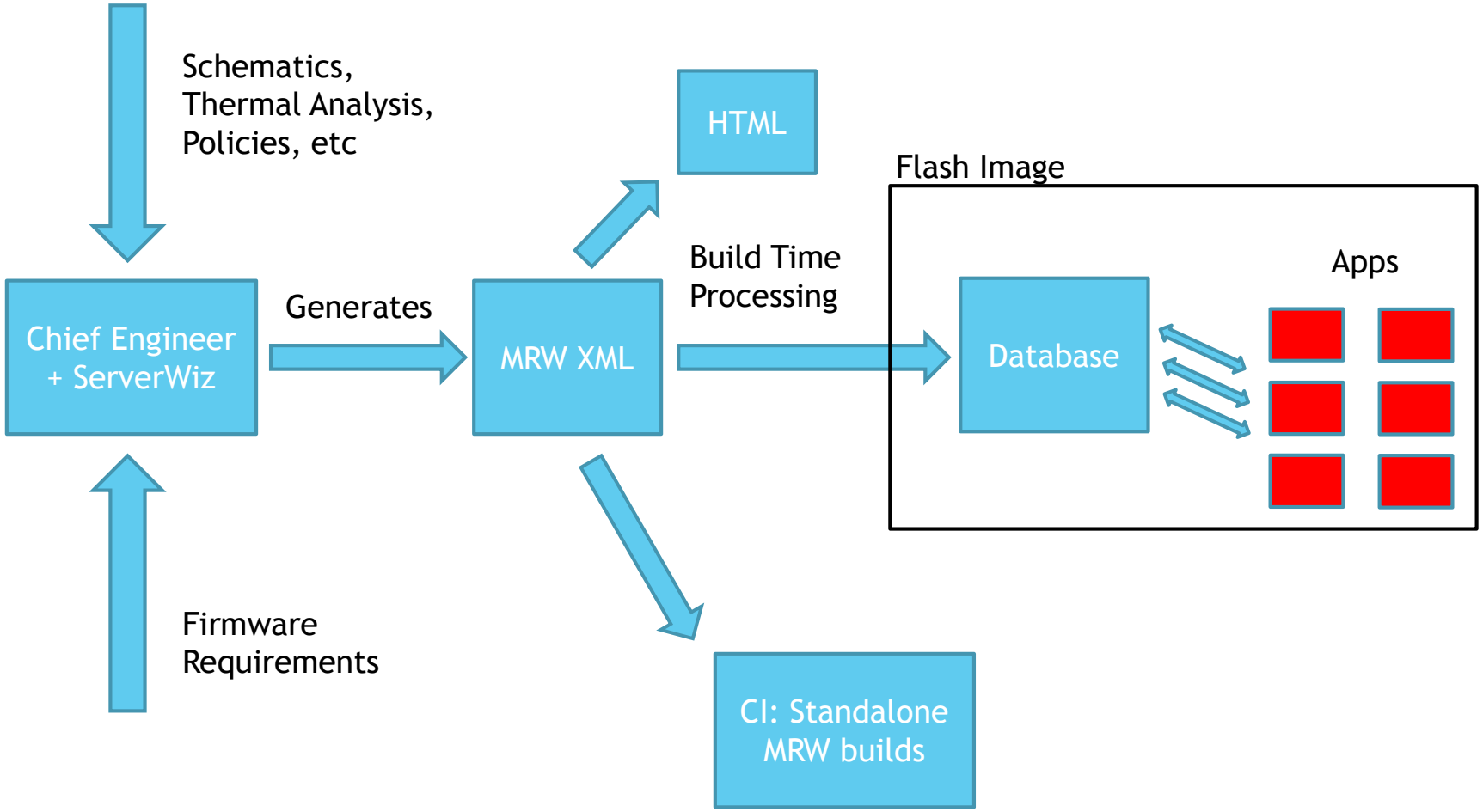
- History
- OpenBMC Design Goals
- Introducing ServerWiz2
- The API
- Customizing
- Patching
- Current Status
- Future Items

History

- IBM's enterprise service processors required a huge amount of data
- Each code component was on their own:
  - Dug up their own data for each system
  - Figure out how to store it and present it
  - Hopefully they'd hear if something changed in the hardware

- The solution? the **M**achine **R**eadable **W**orkbook
  - (as opposed to a PDF system workbook)
  - The chief engineer is responsible for the data
  - Uses the ServerWiz GUI for data entry, mostly
  - What is typed in can be read directly by code for use during build or runtime
  - Almost everything went into a relational DB in flash
  - Also HTML output for human consumption

Legacy Flow


Schematics,
Thermal Analysis,
Policies, etc
HTML
Flash Image
Build Time
Processing
Apps
Chief Engineer
+ ServerWiz
Generates
MRW XML
Database
Firmware
Requirements
CI: Standalone
MRW builds

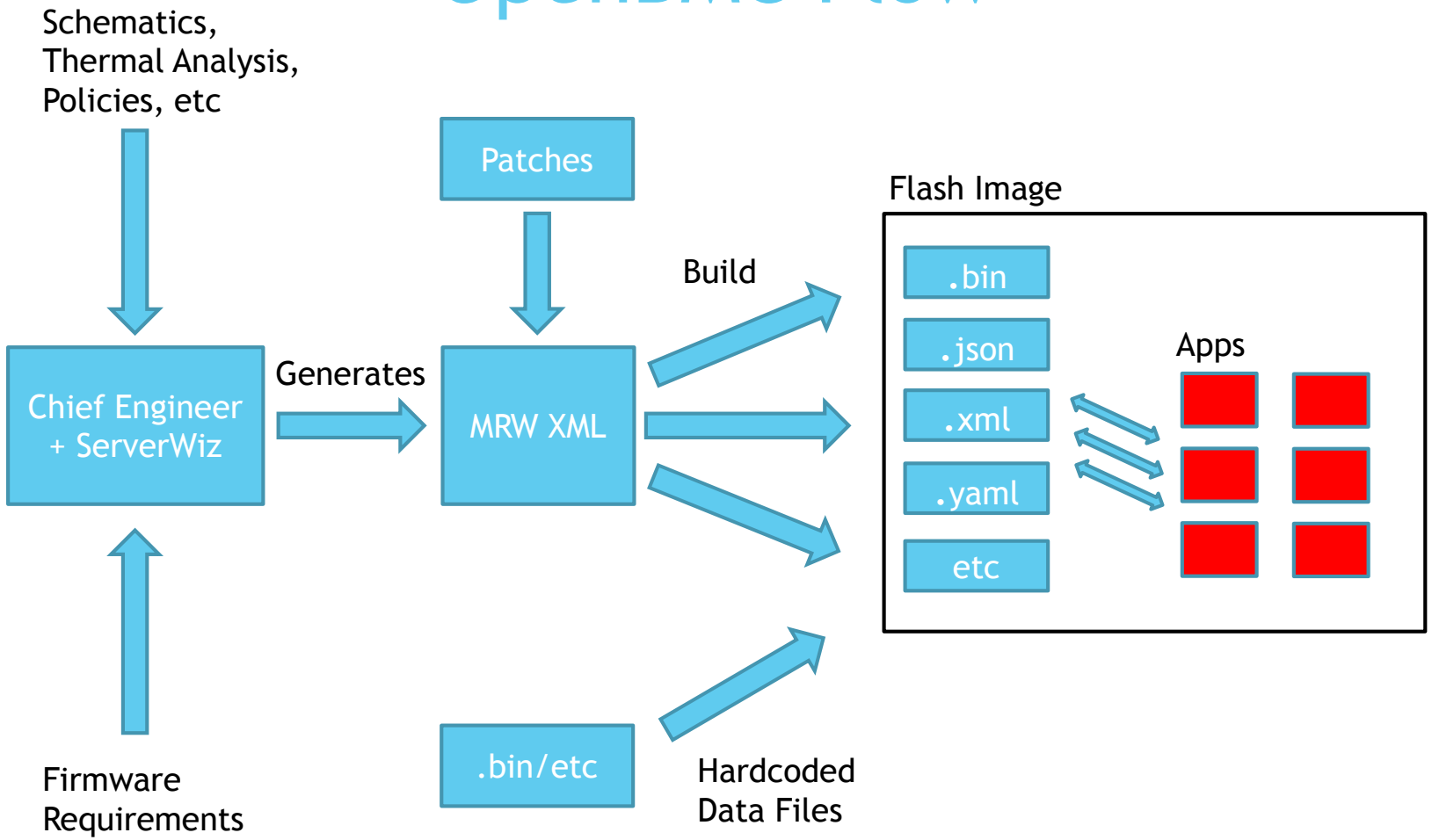
Ideal Goals for OpenBMC

- Use ServerWiz2!
- All system data comes from 1 repository
- The system owner is responsible for the data
- Components have no data related code changes for new (similar) systems
- Components own their own processing scripts and data formats
- There is an easy method to patch the system data for quick fixes

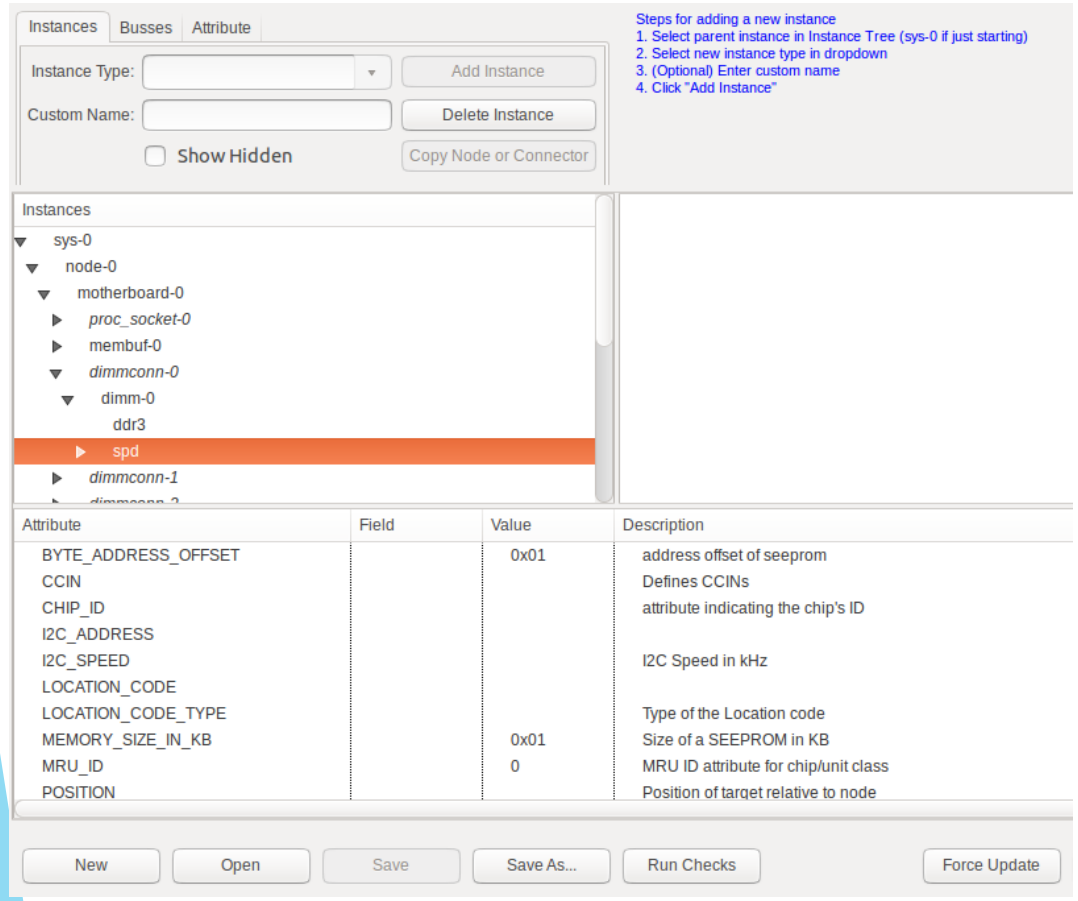
Lowest Bar Goals for OpenBMC

- Data is moved out of the code, provided by Serverwiz2 or hard coded or whatever
- Components have no data related code changes for new systems

OpenBMC Flow



Introducing ServerWiz2



- Hostboot uses today
- Platform Independent
- Open Source, and IBM supported
- Releases on Github
- Generates a single XML file
- Everything is a target
- Targets have attributes, connections
- XML metadata defines available target types and attributes
- Provides a Perl module to traverse the XML

https://github.com/open-power/serverwiz

https://github.com/open-power/serverwiz/blob/master/doc/Serverwiz2%20Overview.pdf

- Serverwiz2 is a hierarchically based XML editor that is targeted for representing a system topology.
- It has 3 primary concepts:
  - Instances
    - Node, card, connector, or chip
    - Chips can have units that specify subcomponents of that chip such as cores and bus interfaces
  - Busses/Connections
    - A connection between 2 units of Instances
    - Connections are made at the level in the hierarchy where they exist in the real system
  - Attributes
    - Instances and Connections both have attributes
    - Attributes are variables that Hostboot reads to control the behavior

The Targets.pm Perl API

- Parses <system>.xml
- Target, bus, and attribute based
- See Hostboot's src/usr/targeting/common/processMrw.pl

```
use Targets;
$targetObj->loadXML($serverwiz_file);
foreach my $target (sort keys %{ $targetObj->getAllTargets() }) {
    print "Target: $target\n";

    my $type = $targetObj->getType($target);
    my $fru_id = $targetObj->getAttribute($target,"FRU_ID");

    if ($type eq "BMC") {

        my $i2cs=$targetObj->findConnections($target,"I2C","PROC");

        foreach my $i2c (@{$i2cs->{CONN}}) {
            my $addr=$targetObj->getBusAttribute($i2c->{SOURCE},
                                                 $i2c->{BUS_NUM},"I2C_ADDRESS");
        }
    }
}


Target: /sys-0
Target: /sys-0/apss-0
Target: /sys-0/node-0
Target: /sys-0/node-0/motherboard-0
Target: /sys-0/node-0/motherboard-0/bmc-0
Target: /sys-0/node-0/motherboard-0/bmc-0/bmc_i2c_master
Target: /sys-0/node-0/motherboard-0/bmc-0/bmc_lpc_master
Target: /sys-0/node-0/motherboard-0/dimmconn-0
Target: /sys-0/node-0/motherboard-0/dimmconn-0/dimm-0
```

Customizing the XML

- To add an attribute field for serverwiz:
  1) Add the attribute definition to attribute_types_obmc.xml
  2) Specify which target type needs the attribute in target_types_mrw.xml
  3) Open ServerWiz, navigate to the instance, fill in the new value
  4) To use: $targetObj->getAttribute($bmcTarget, “BMC_MODEL”)

```
<attribute>
    <id>BMC_MODEL</id>
    <description>The BMC model</description>
    <simpleType>
        <string></string>
    </simpleType>
    <persistency>non-volatile</persistency>
    <readable/>
</attribute>
```

```
<targetType>
    <id>chip-sp-bmc</id>
    <parent>chip</parent>
    <parent_type>card-motherboard</parent_type>
    <parent_type>card-daughtercard</parent_type>
    <attribute>
        <id>BMC_MODEL</id>
        <default>NA</default>
    </attribute>
    ...
```

MRW XML Patching

- For quick MRW XML fixes, or prototyping
- Fixes are also in XML, will get applied during do_patch()
- …/meta-palmetto/recipes-phosphor/mrw/mrw-native/palmetto.xml.patch.xml

```
<patches>
    <targetFile>palmetto.xml</targetFile>

    <!-- Fix the PCIE card's type -->
    <attribute type="replace" xpath="targetInstance[id='pciecard_x8-0']/attribute[id='TYPE']">
        <id>TYPE</id>
        <default>PCIE_CARD</default>
    </attribute>

    <!-- Add its FRU name -->
    <attribute type="add" xpath="targetInstance[id='pciecard_x8-0']">
        <id>FRU_NAME</id>
        <default>PCIE_CARD</default>
    </attribute>
</patches>
```

Current Status

- Recipes out on gerrit:
  - Pull in MRW XML and Targets.pm from github.com/open-power
- Coming soon:
  - The Bitbake class to apply the XML patches
  - Generate system inventory from the XML

Future Items

- Device Tree
- Fan Control Parameters
- IPMI SDR data
- Hotplug Rules
- etc