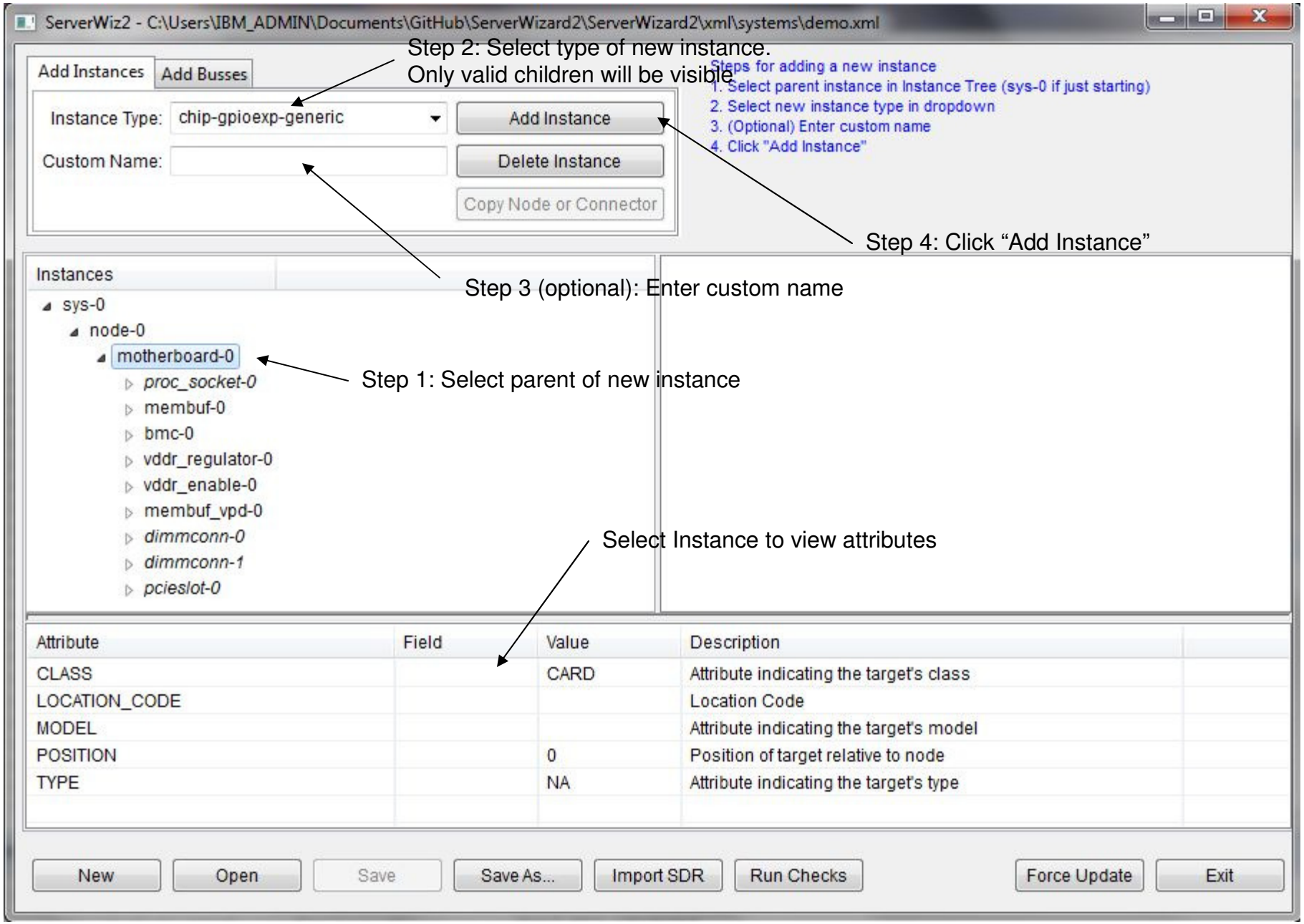
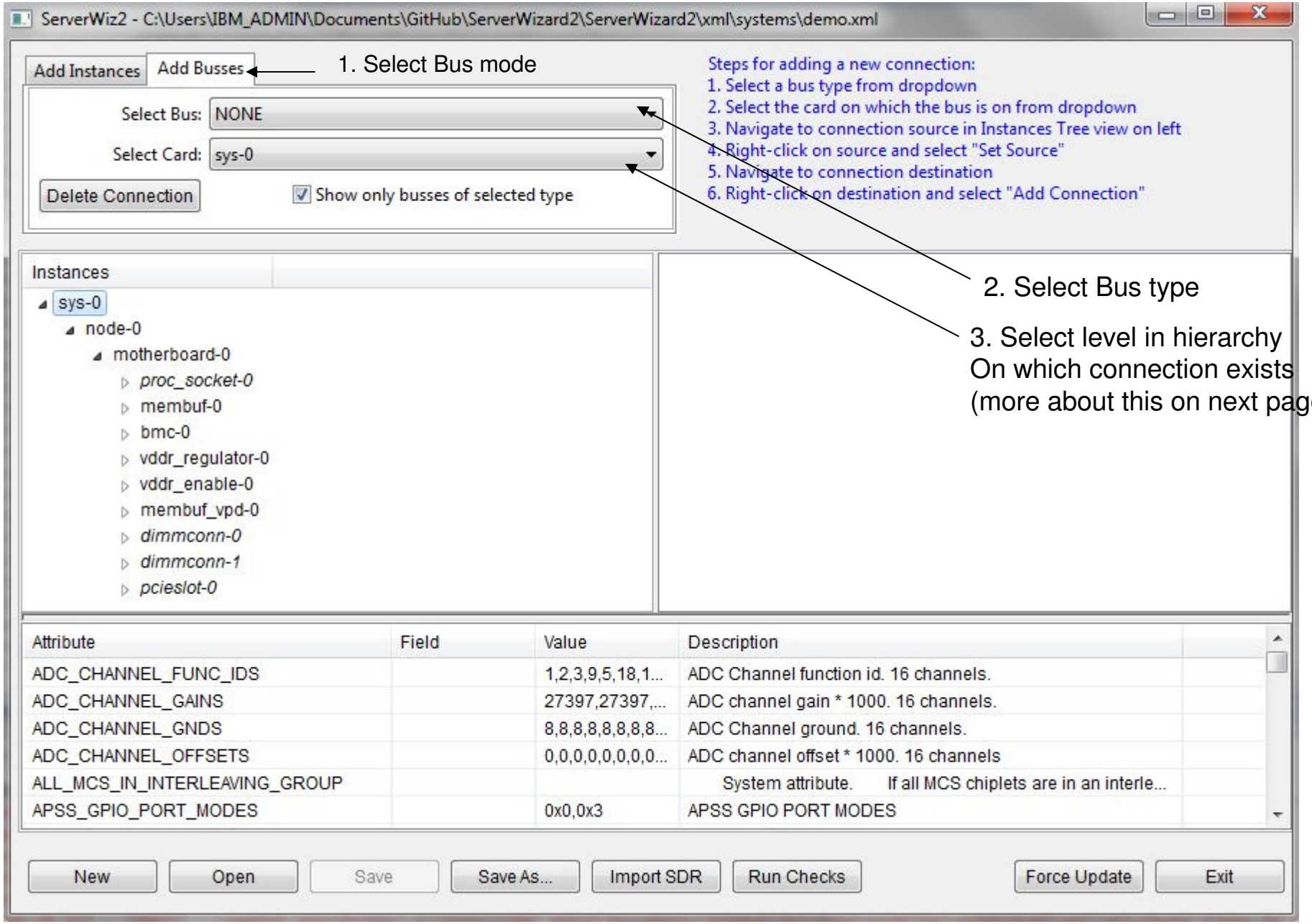
ServerWiz2 Overview

- 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

Instance Creation

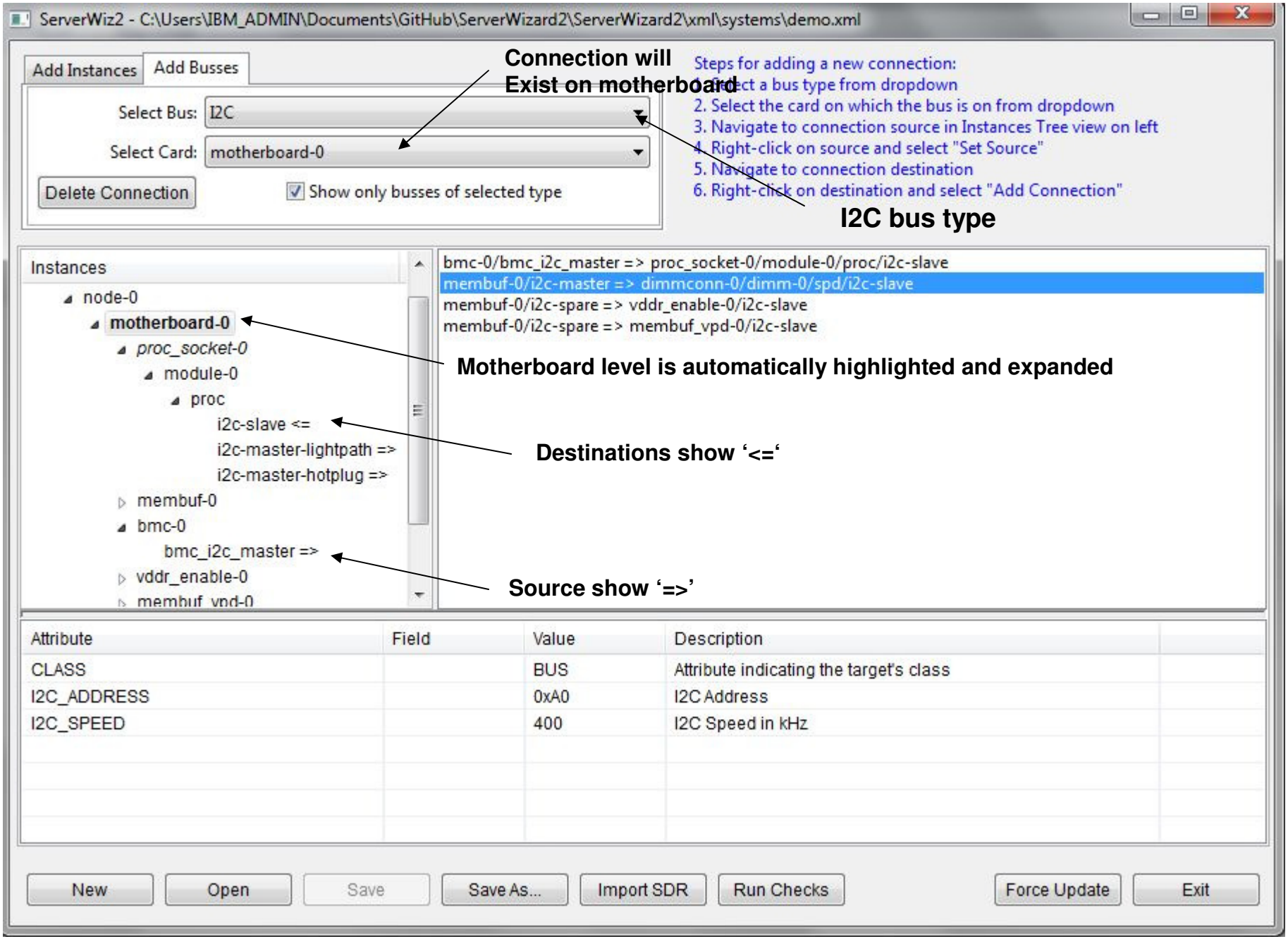


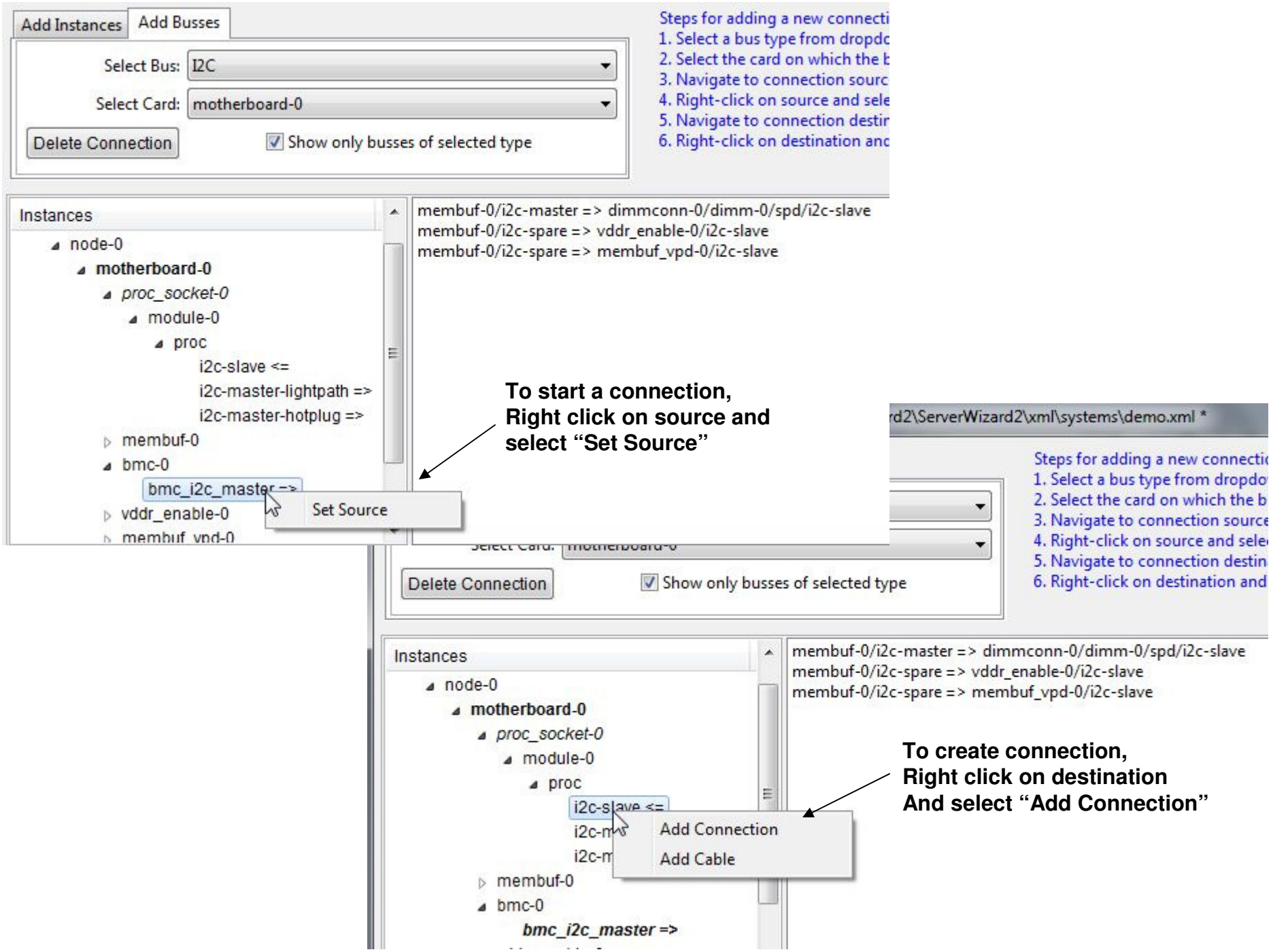
Connection Creation

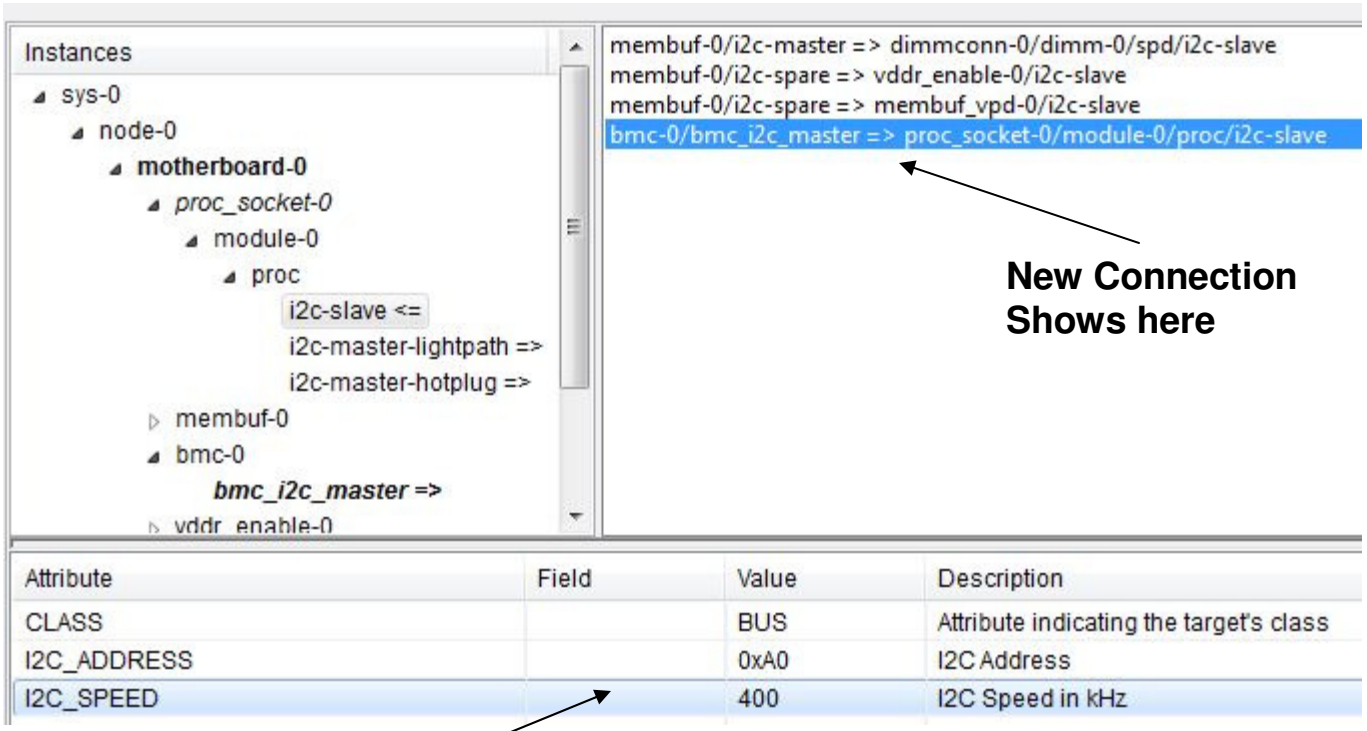


Connection Hierarchy

- Connections must be created at highest common point in hierarchy or physically where wire exists. Here are some examples:
  - Simple single motherboard system
    - All connections are created at motherboard level so Selected Card = motherboard.
    - The motherboard is also obviously where the physical wires exist
  - System with memory riser cards
    - The DMI bus spans the motherboard and riser card
    - The motherboard level is selected because that is the highest common level in the hierarchy
  - Multi-node system with cables connecting nodes
    - System level is selected because that is highest common level in hierarchy


ServerWiz2 - C:\Users\IBM_ADMIN\Documents\GitHub\ServerWizard2\ServerWizard2\xml\systems\demo.xml
Add Instances
Add Busses
Select Bus: I2C
Select Card: motherboard-0
Delete Connection
Show only busses of selected type
Steps for adding a new connection:
1. Select a bus type from dropdown
2. Select the card on which the bus is on from dropdown
3. Navigate to connection source in Instances Tree view on left
4. Right-click on source and select "Set Source"
5. Navigate to connection destination
6. Right-click on destination and select "Add Connection"
Connection will Exist on motherboard
I2C bus type
Instances
node-0
motherboard-0
proc_socket-0
module-0
proc
i2c-slave <=
i2c-master-lightpath =>
i2c-master-hotplug =>
membuf-0
bmc-0
bmc_i2c_master =>
vddr_enable-0
membuf_vpd-0
bmc-0/bmc_i2c_master => proc_socket-0/module-0/proc/i2c-slave
membuf-0/i2c-master => dimmconn-0/dimm-0/spd/i2c-slave
membuf-0/i2c-spare => vddr_enable-0/i2c-slave
membuf-0/i2c-spare => membuf_vpd-0/i2c-slave
Motherboard level is automatically highlighted and expanded
Destinations show '<='
Source show '=>'
Attribute
Field
Value
Description
CLASS
BUS
Attribute indicating the target's class
I2C_ADDRESS
0xA0
I2C Address
I2C_SPEED
400
I2C Speed in kHz
New
Open
Save
Save As...
Import SDR
Run Checks
Force Update
Exit


Add Instances
Add Busses
Select Bus: I2C
Select Card: motherboard-0
Delete Connection
Show only busses of selected type
Steps for adding a new connecti
1. Select a bus type from dropdc
2. Select the card on which the b
3. Navigate to connection sourc
4. Right-click on source and sele
5. Navigate to connection destir
6. Right-click on destination anc
Instances
node-0
motherboard-0
proc_socket-0
module-0
proc
i2c-slave <=
i2c-master-lightpath =>
i2c-master-hotplug =>
membuf-0
bmc-0
bmc_i2c_master =>
vddr_enable-0
membuf_vpd-0
Set Source
membuf-0/i2c-master => dimmconn-0/dimm-0/spd/i2c-slave
membuf-0/i2c-spare => vddr_enable-0/i2c-slave
membuf-0/i2c-spare => membuf_vpd-0/i2c-slave
To start a connection,
Right click on source and
select “Set Source”
rd2\ServerWizard2\xml\systems\demo.xml *
Steps for adding a new connecti
1. Select a bus type from dropdo
2. Select the card on which the b
3. Navigate to connection source
4. Right-click on source and sele
5. Navigate to connection destin
6. Right-click on destination and
Delete Connection
Show only busses of selected type
Instances
node-0
motherboard-0
proc_socket-0
module-0
proc
i2c-slave <=
i2c-m
i2c-m
membuf-0
bmc-0
bmc_i2c_master =>
Add Connection
Add Cable
membuf-0/i2c-master => dimmconn-0/dimm-0/spd/i2c-slave
membuf-0/i2c-spare => vddr_enable-0/i2c-slave
membuf-0/i2c-spare => membuf_vpd-0/i2c-slave
To create connection,
Right click on destination
And select “Add Connection”



New Connection Shows here

Select connection to View attributes. For example, An I2C bus has an address and speed.

Minimum System Requirements

System is always top level

Node is always second level

A System typically has a main board or motherboard

Must have at least 1 processor socket and processor chip

Must have at least 1 memory buffer

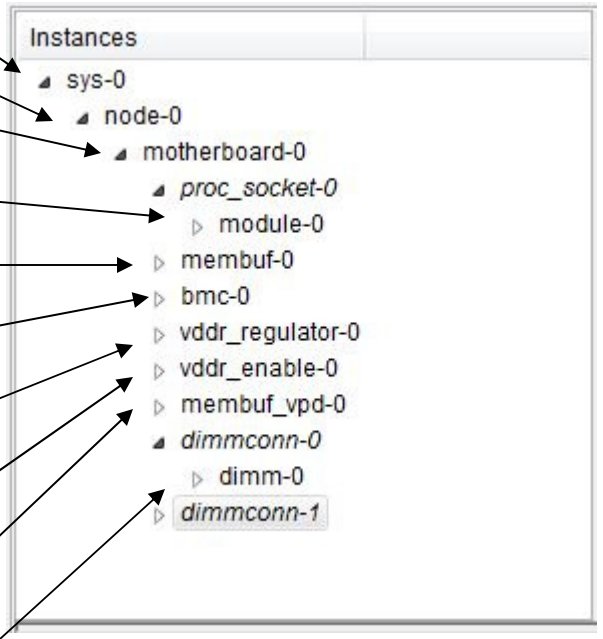
Must have 1 and only 1 BMC

Must have at least 1 VDDR regulator
(hostboot controls enable and must be aware of topology)

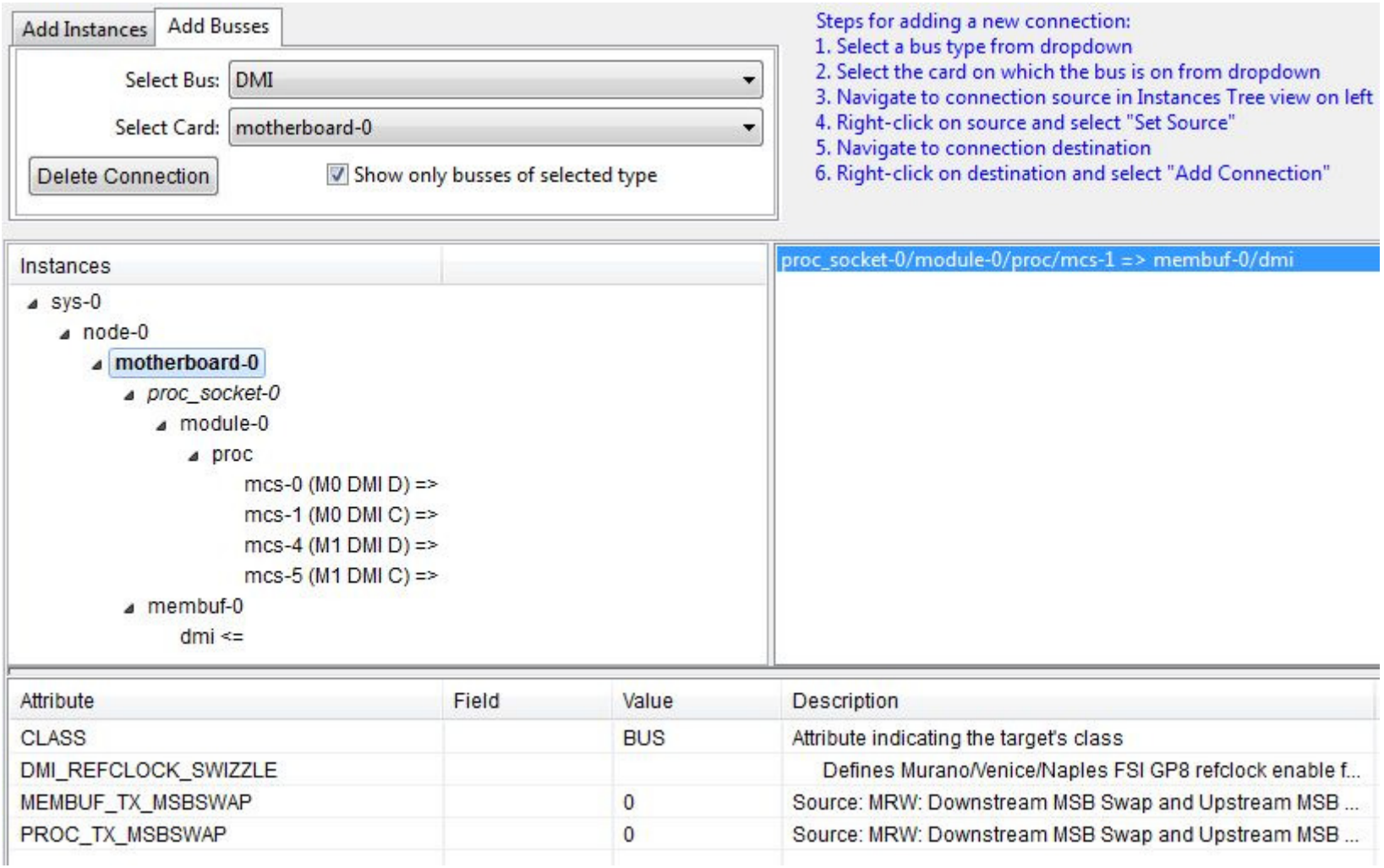
Must have at least 1 GPIO expander that controls VDDR regulator enable

Must have at least 1 VPD for storing memory wiring information

Must have at least 1 part of DIMM connectors and DIMMs

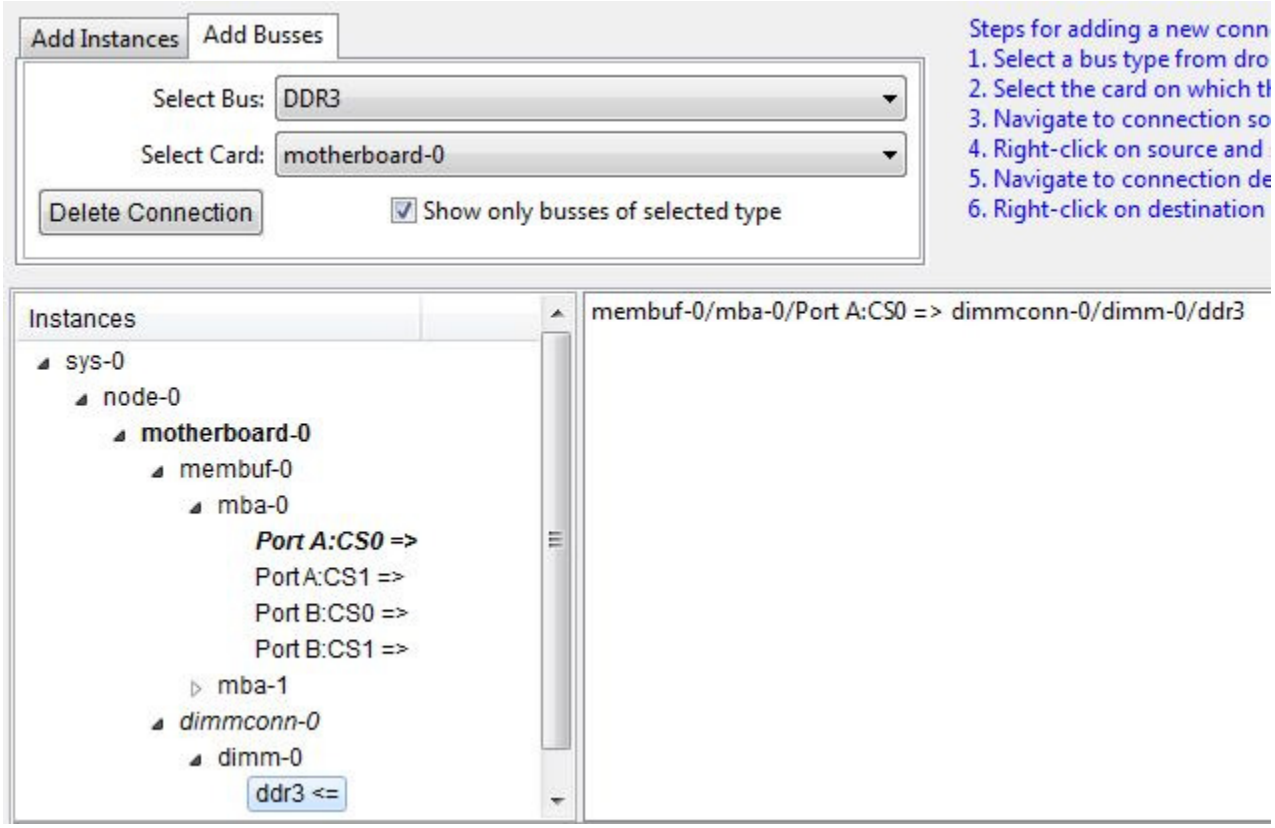


DMI



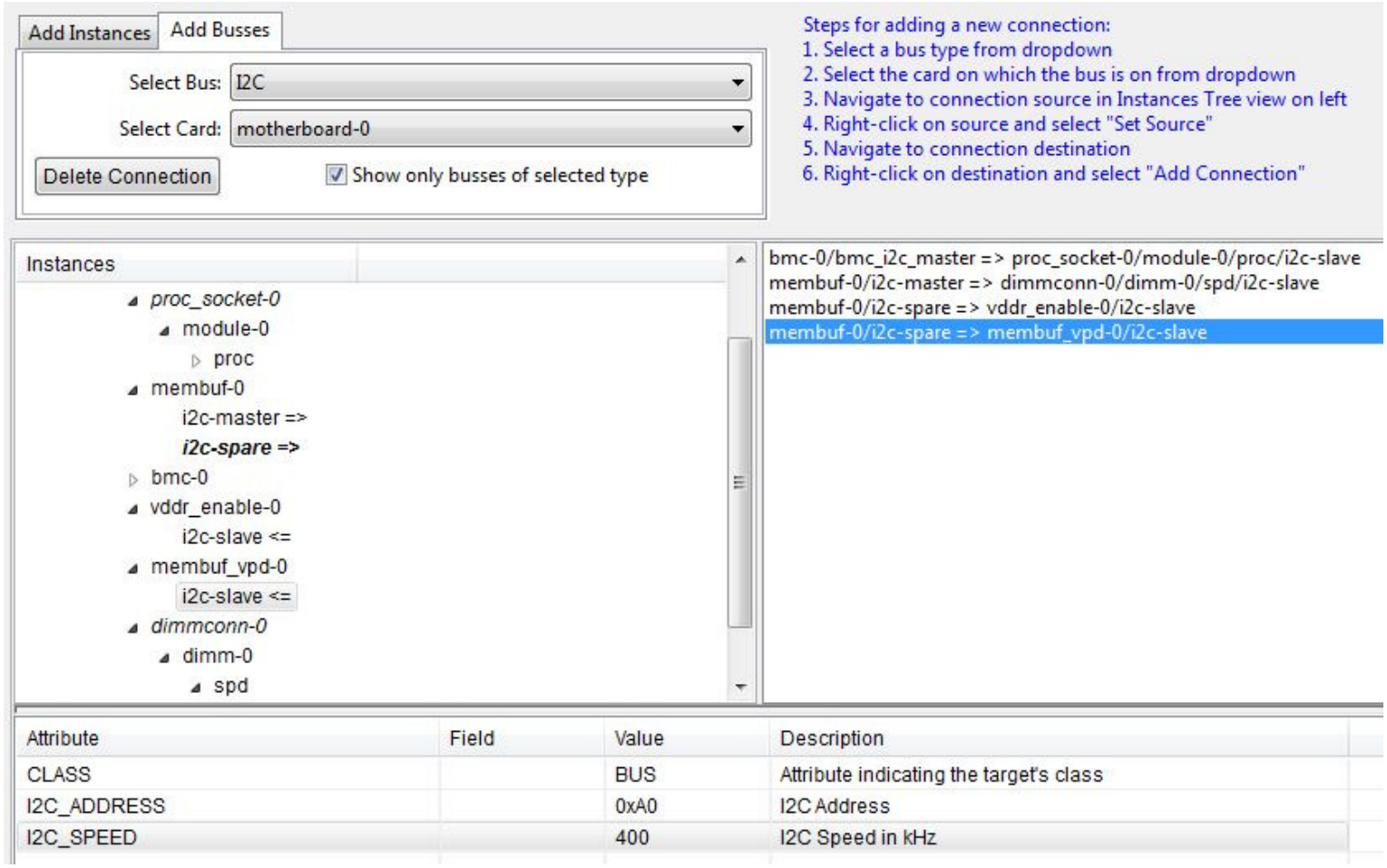
- Required Connections: All membuf's must have a DMI connection to a CPU
- The DMI name in the parenthesis match the schematic names
- If there is an lane reversal in the design, change the MSBSWAP attributes below to "1".

DDR



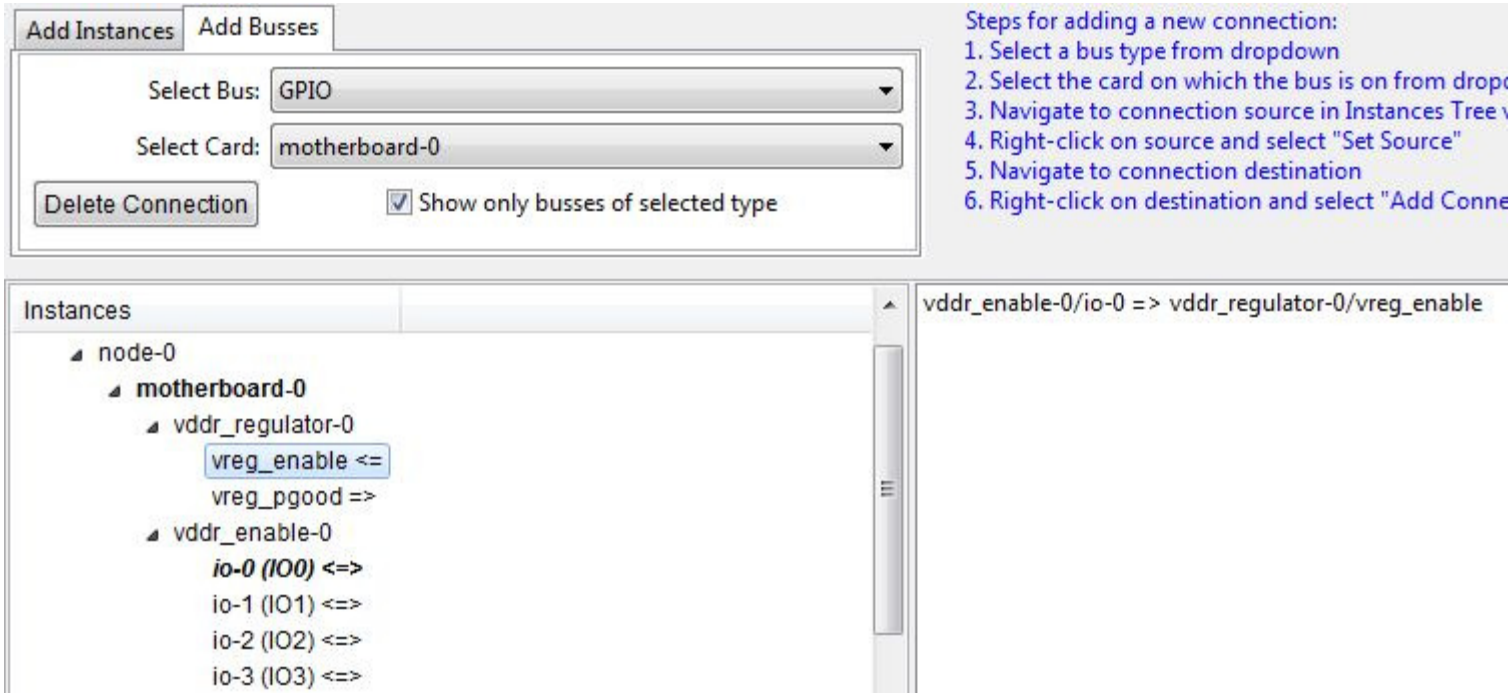
- Required Connections: All DIMMs must be connected to a membuf
- The names for the DDR ports match schematic names. Make sure DIMM naming convention and connections match schematic. It will make the I2C connections more straightforward.

I2C



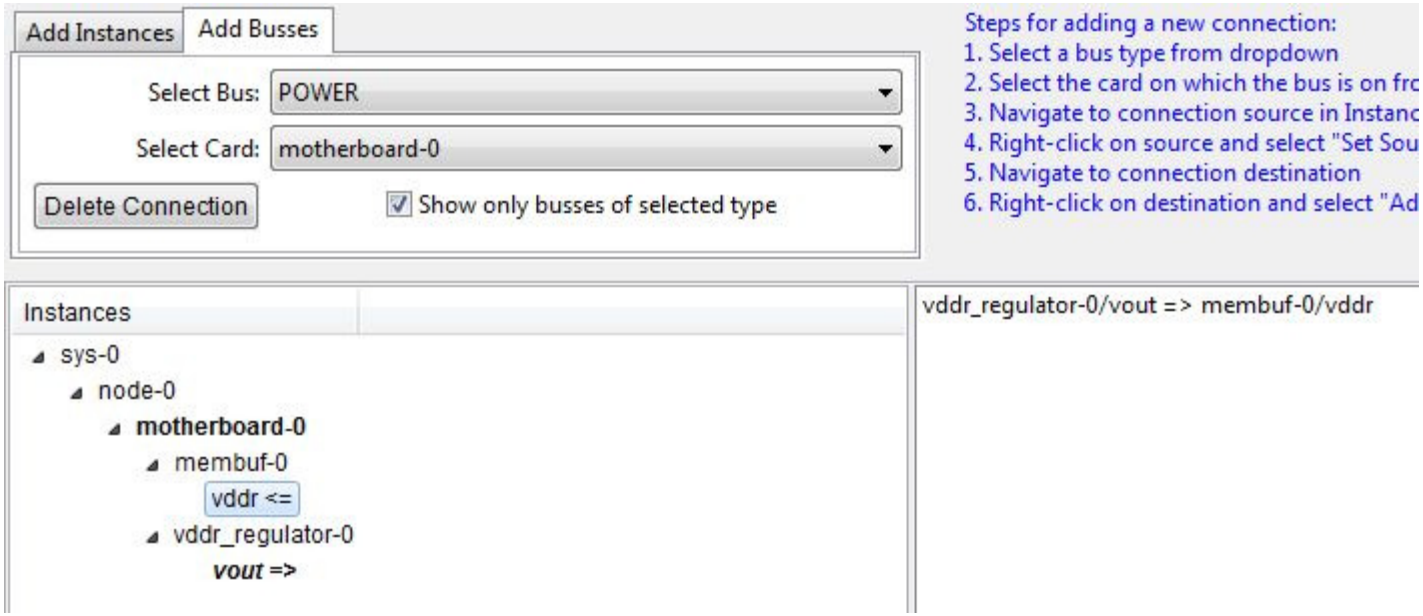
- Required connections:
  - BMC to CPU I2C slave for OCC communication
  - Membuf I2C connections to DIMMs SPD
  - Membuf I2C connection to GPIO expander to VDDR enable
  - Membuf or CPU I2C connection to VPD
- For I2C busses, make sure I2C_ADDRESS and I2C_SPEED attributes match the design

GPIO



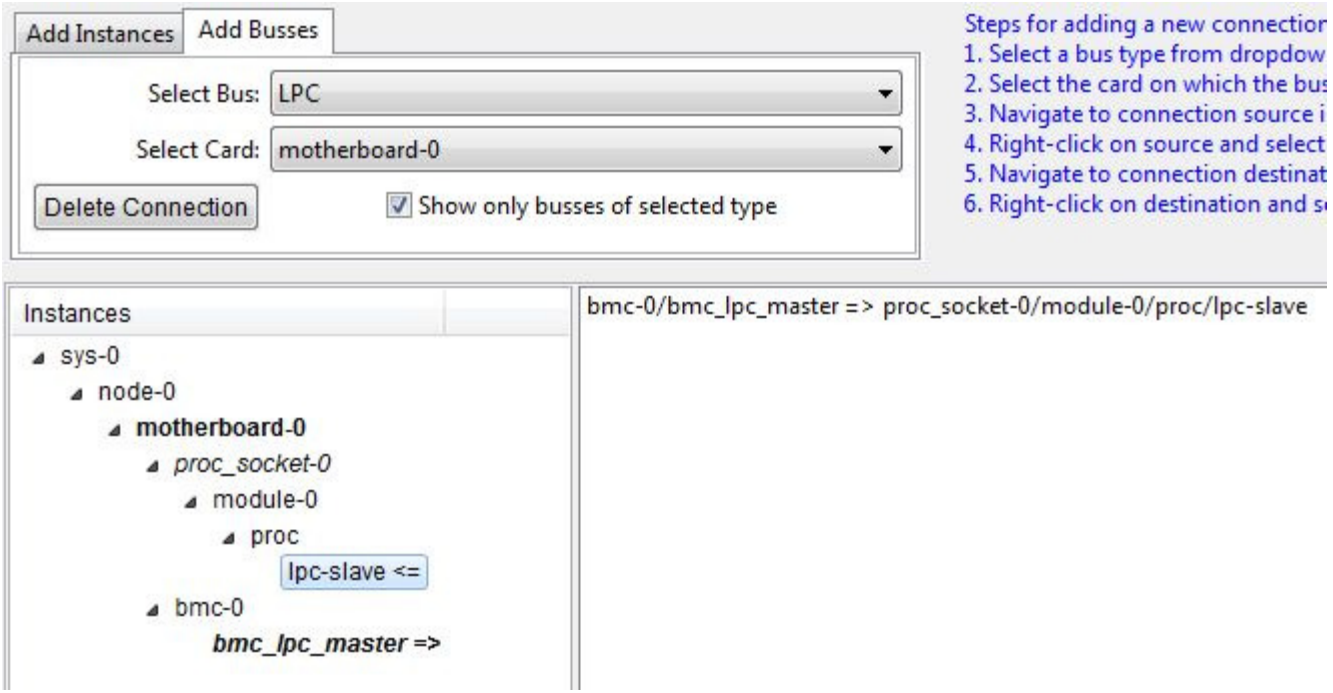
- Required Connection: GPIO expander that controls the VDDR regulator enable. The GPIO port # from GPIO expander must match design.

Power



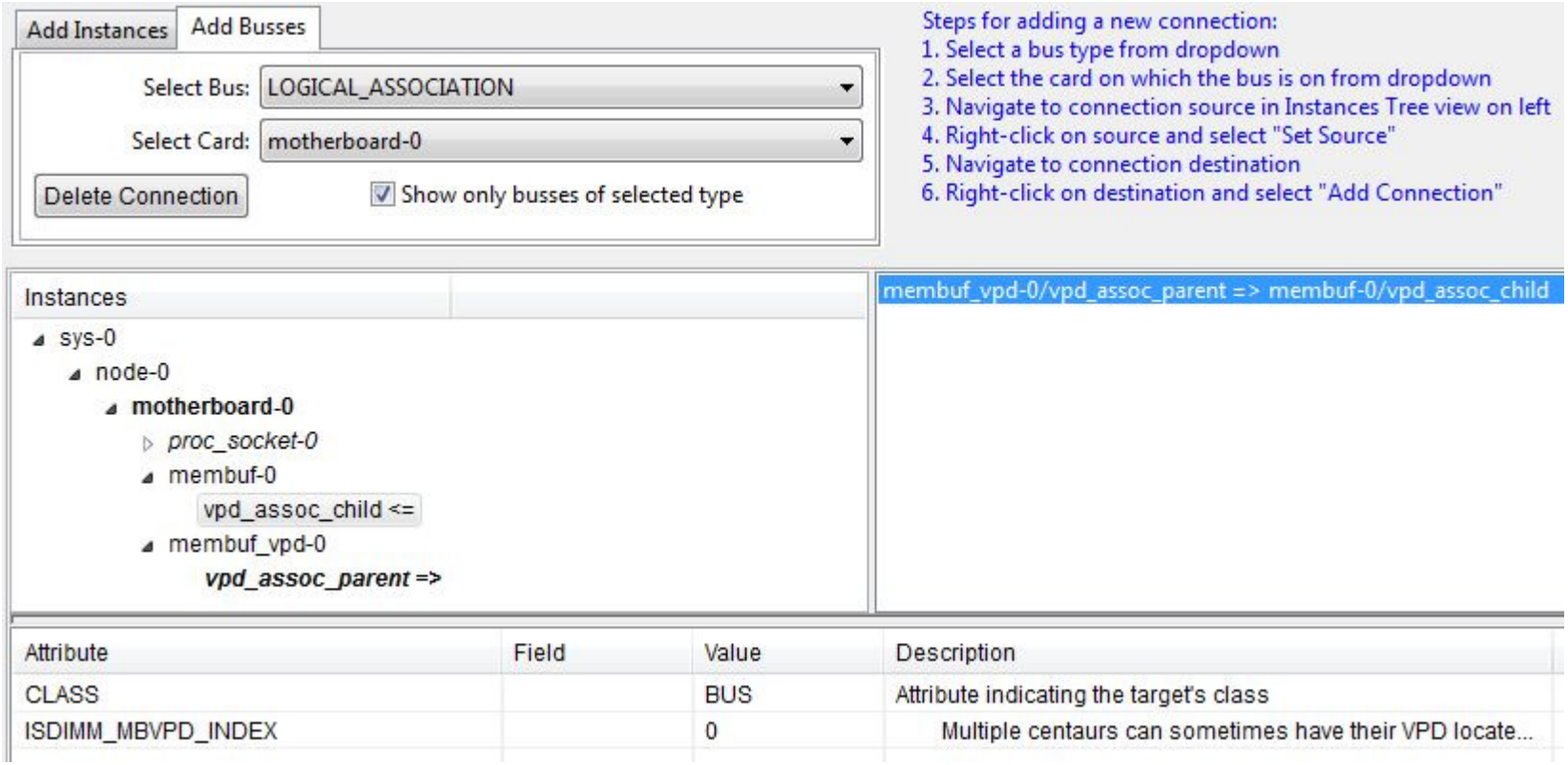
- Required Connections: VDDR regulator connection to membuf

LPC Bus



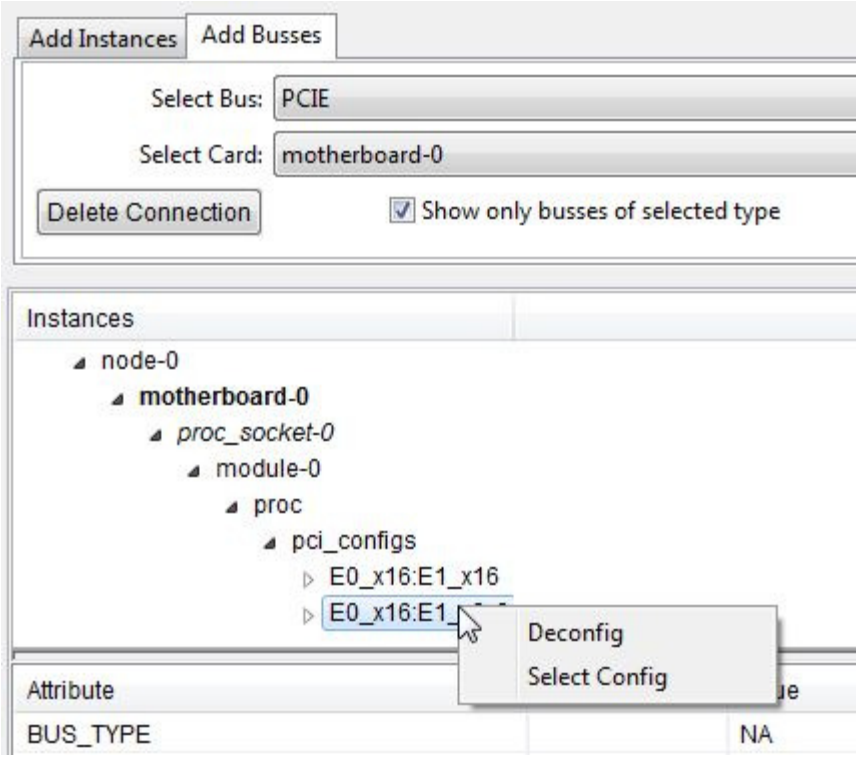
- Required connection: The LPC bus connection between the BMC and one of the CPU's tells Hostboot which CPU is the master.

Logical Association



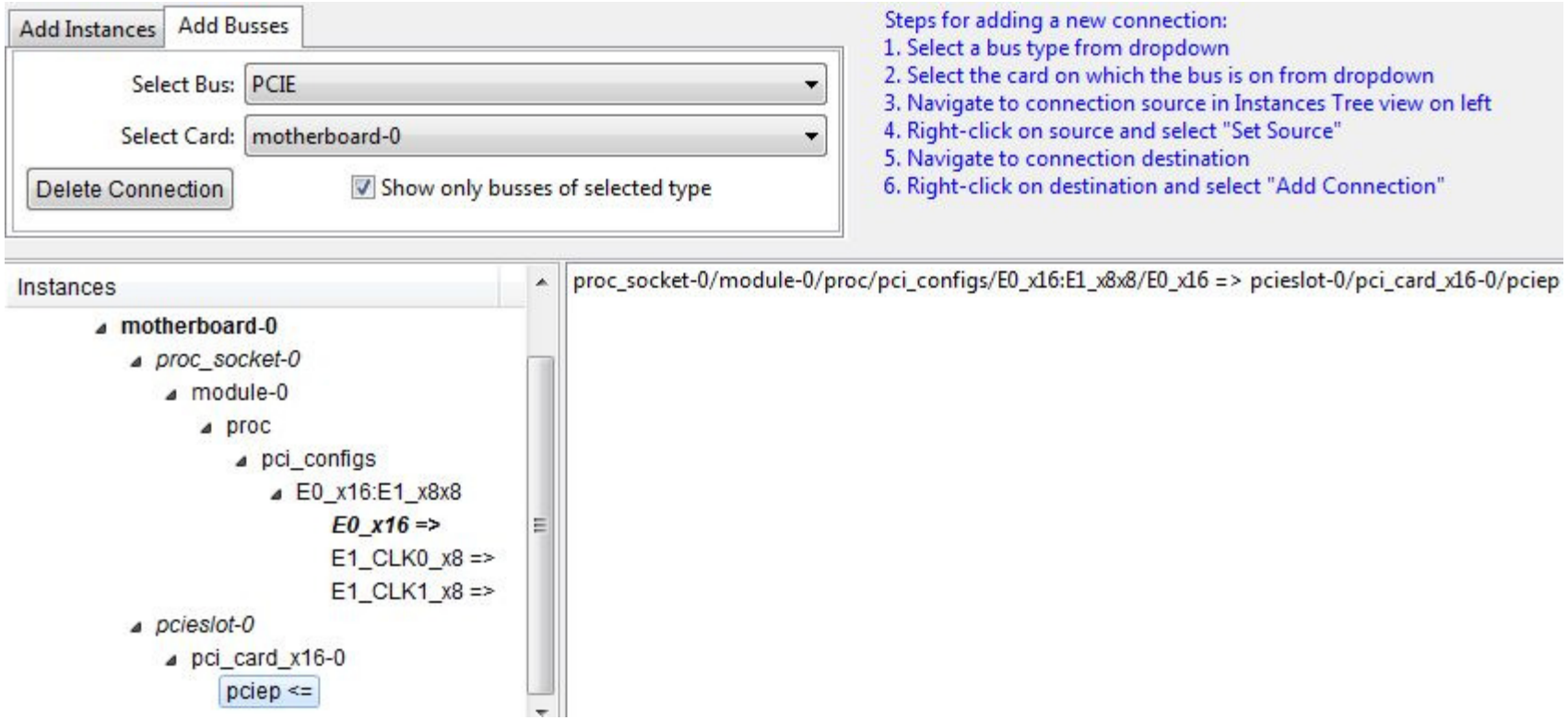
- Required connections: VPD that contains membuf wiring information connection to membuf
- This is a virtual connection that tells hostboot where the VPD exists for each membuf. The VPD can alternatively be attached to the CPU I2C master
- This logical association concept could be extended to FRU LED associations

PCIe



- The PCIe bus is unique in that it can be configured in several different ways. Under the “pci_configs” parent, you will see the various configurations. To choose one, right-click and select “Select Config”. The other configurations will be hidden. To make all configurations visible, right-click and select “Deconfig”.

PCIe



- Required Connections: None
- Here is an example where E0 is configured as a x16 and E1 is configured as 2 x8's.

Serverwiz Library Files

- Every 3 days, Serverwiz will check the XML files at:
  - https://github.com/open-power/serverwiz/tree/master/xml
  - If the files have change, it will prompt the user if they wish to download new files
- Description of Files:
  - attribute_types.xml, attribute_types_hb.xml, attribute_types_mrw.xml
    - Listing of all possible attributes including data type, default value, and description
    - Attribute_type.xml and attribute_types_hb.xml are same as hostboot
  - target_types_mrw.xml
    - Describes all the possible instances, the type of the instance, and the associated attributes
    - Systems, chips, cards, etc
    - Also describes the valid parent instance types
  - target_instances_v3.xml
    - Specific instances of a target types including attribute values and children

Making Changes

- Additions of attributes might occur because of new Hostboot requirements or features
- The change must first be thought of from a end user point of view (system designer/user of Serverwiz)
  1. If the attribute is a simple value that the user can enter directly then the developer must simply add the attribute into Serverwiz's attribute_types.xml and target_types_mrw.xml under appropriate target. The attribute will then show up in the Serverwiz and the user can enter desired value.
  2. If the attribute is a computed value based on several factors, then the handling of the attribute must be in processMrw.pl
     - If the computed valued depends on static values that are also new attributes, then follow Step 1.

IPMI Sensor Overview

- The POWER BIOS (Hostboot, OCC, and OPAL) updates virtual BMC sensors via IPMI set sensor commands. The BIOS has to be informed of the IPMI sensor ID's. It does so through IPMI_SENSORS attribute.
- Serverwiz can import XML (which could be generated by BMC development environment) that contains the sensor information using the "Import SDR" button and populate the IPMI_SENSORS attribute
- There is a strict mapping between the instance type and the IPMI entity ID. The mapping is maintained in the ENTITY_ID_LOOKUP attribute which is read-only (see Table on next page)
- The Serverwiz IPMI_INSTANCE attribute maps to the IPMI entity instance in the SDR import XML. The user must make sure the IPMI_INSTANCE attribute is unique per entity ID.

Entity ID Mapping

Sensor Name	Entity ID	Sensor Type Code	Target Association
CPU Temperature	0x03	0x01	proc
CPU Core Temperature	0xD0	0x01	core
CPU Functional Status	0x03	0x07	proc
CPU Core Functional Status	0xD0	0x07	core
CPU Core Frequency	0xD0	0xC1	core
Memory Buffer Temperature	0xD1	0x01	membuf
DIMM Temperatur	0x20	0x01	dimm
Memory Buffer Functional	0xD1	0x0C	membuf
DIMM Functional Status	0x20	0x0C	dimm
DIMM Frequency	0x08	0xC1	system
Backplane Fault	0x07	0xC7	node
System Event	0x01	0x12	system
Host Status	0x23	0x22	system
Firmware Boot Progress	0x22	0x0F	system
PCIe Link Present	0x23	0xC4	system
OS Boot Status	0x23	0x1F	system
OCC Active	0xD2	0x07	occ
Power Cap	0x17	0xC2	system
Boot Count	0x22	0xC3	system
System Power Limit (User set)	0x15	0xC2	system
Activate Power Limit	0x15	0xC6	system
APSS_Channel	0xD7	0xC2	system

* Entity Instance from SDR maps to IPMI_INSTANCE attribute

SDR XML Format

```
<devices>
    <device>
        <name>Name</name>
        <dev_name>Secondary Name</dev_name>
        <sensor_id>[Sensor ID in decimal]</sensor_id>
        <entity_id>[Entity ID in hex]</entity_id>
        <sensor_type>[Sensor type in hex]</sensor_type>
        <entity_instance>[Instance number in hex]</entity_instance>
    </device>
    …
</devices>
```

When the “Import SDR” button is used in Serverwiz, it will import a file of this format. It will attempt to match up the entity id from the ENTITY_ID_LOOKUP under a given target and the IPMI_INSTANCE attribute to the entity_id and entity_instance fields from the SDR import file.