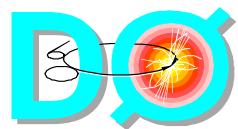
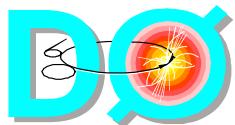

DØ Detector Download Tutorial

J. Frederick Bartlett



Outline - Part I

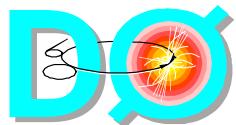
- Overview of the detector download system components
 - ◆ The relationship between COMICS and EPICS
 - ◆ The relationship between COMICS and COOR
 - ◆ The relationship between COMICS and ORACLE calibration databases



March 27, 2001

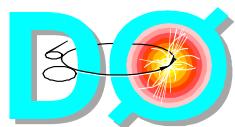
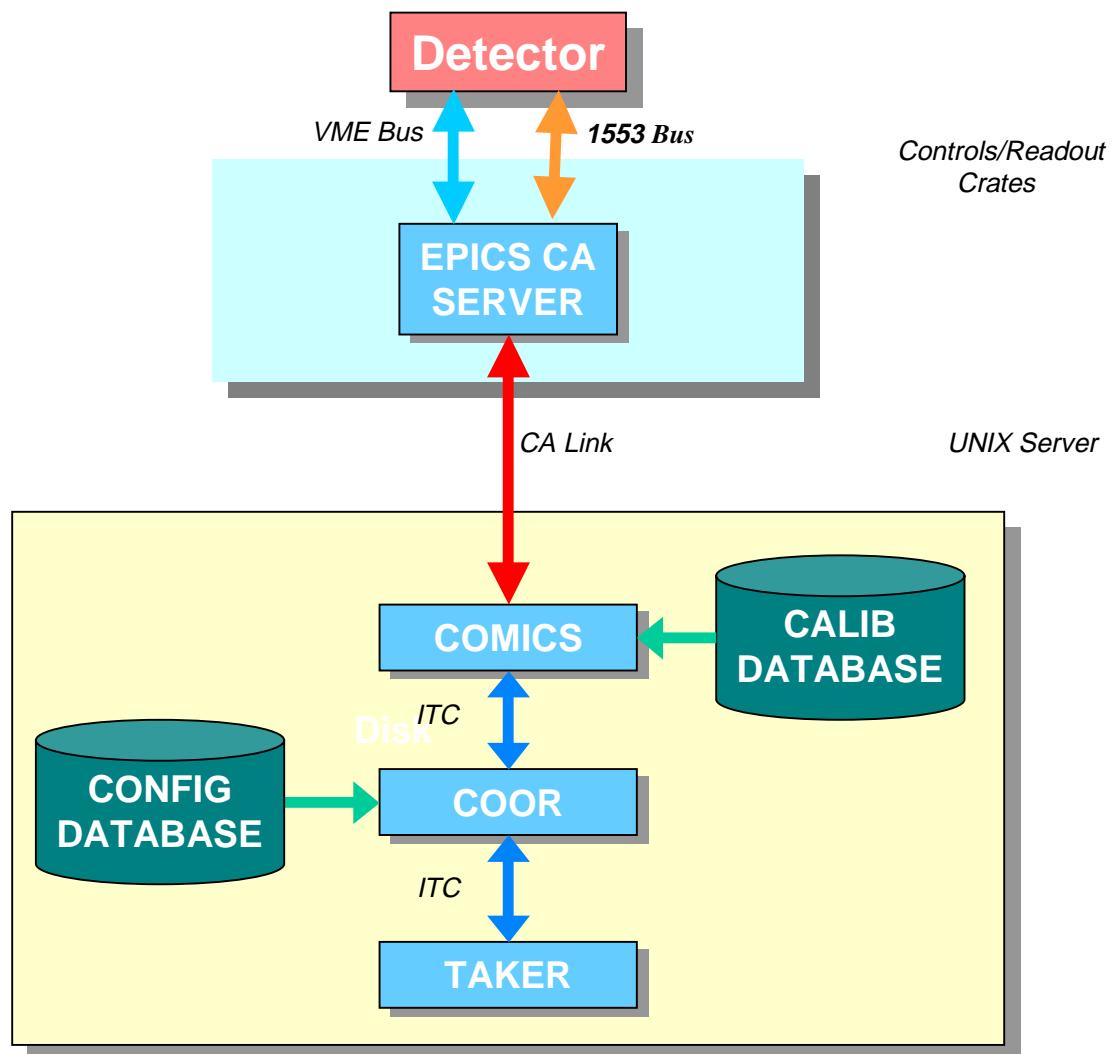
Outline - Part II

- How COMICS is structured
 - ◆ Download tree
 - ◆ Class diagram
 - ◆ Tree Nodes
 - ◆ Action Nodes



March 27, 2001

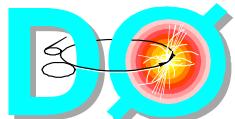
Download Overview



March 27, 2001

Download System Components

- **TAKER**
 - ◆ Operator interface for COOR
- **COOR**
 - ◆ Run state manager
- **COMICS**
 - ◆ Detector configuration manager
- **EPICS**
 - ◆ Distributed control system
- **Detector**
 - ◆ If you don't know what this is, you are in the wrong tutorial



March 27, 2001

EPICS Naming Convention

- Name elements

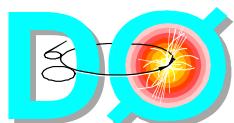
◆ Detector	<det>	CAL
◆ Sub-det	<sub>	N
◆ Device	<dev>	VBD
◆ Locator	<loc>	01
◆ Attribute	<attr>	STATUS
◆ I/O	<io>	W
◆ Field	<field>	SCAN

- Template

<det>[<sub>]<dev>_<loc>[/<attr>[:<io>]].<field>]

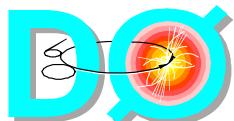
- Example

CALN_VBD_01/STATUS:W.SCAN



COOR Message

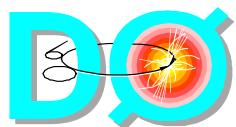
- Message is a text string composed of words separated by white space
- Order of word units
 - ◆ Sequence number (generated by COOR)
 - ◆ Sector name
 - ◆ Name/Value pairs
 - Unspecified number
 - Value may be:
 - ▲ String
 - ▲ Integer number
 - ▲ Floating point number



March 27, 2001

COOR Message Format

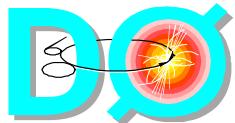
Sequence #
Sector ID
Name0
Value0
Name1
Value1
•
•
•
NameN
ValueN



March 27, 2001

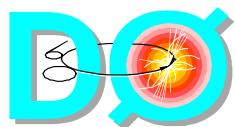
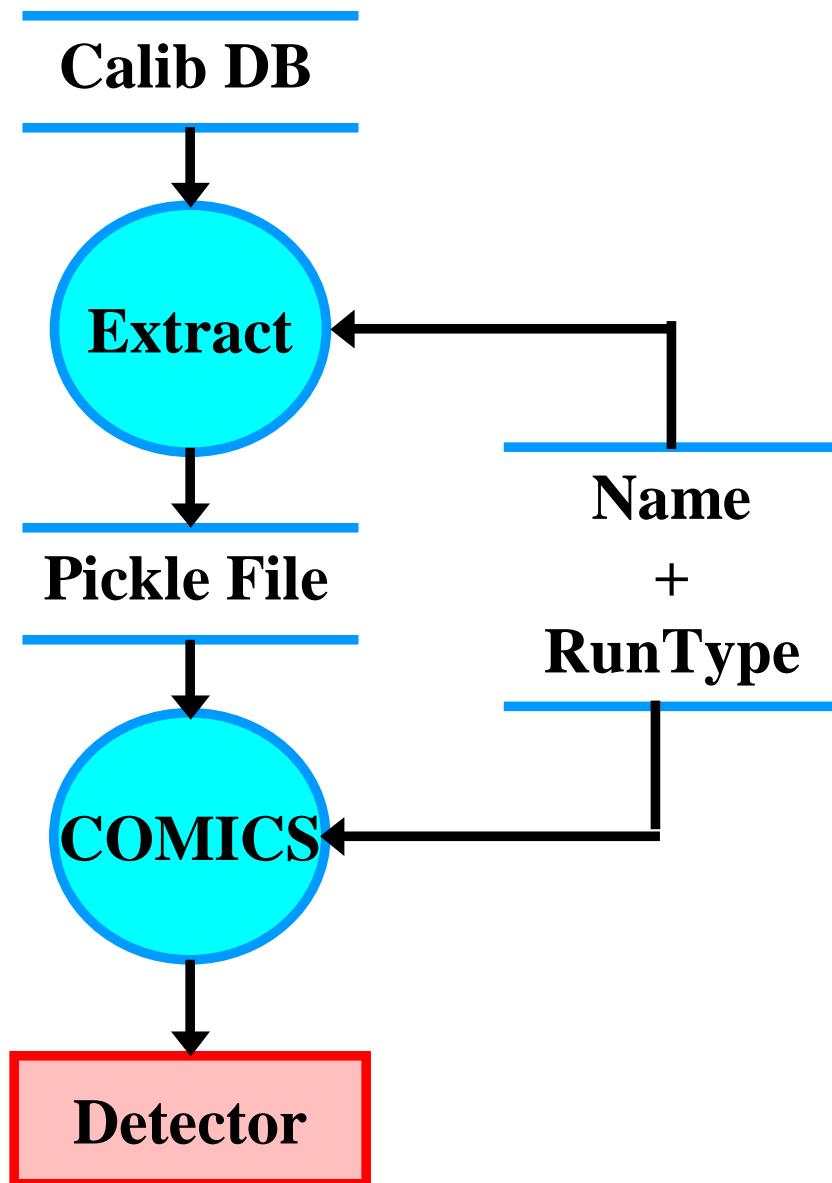
COOR Message

- **Proposed convention:** a required name/value pair designates the run type
 - ◆ Name - RUNTYPE
 - ◆ Values: DATA, CALIB, PEDS, GAINS, ... ?
 - ◆ Used to select a set of download values for each device



March 27, 2001

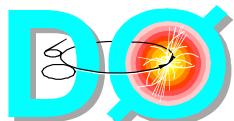
ORACLE Database Extraction



March 27, 2001

Pickle File Name Convention

- Pickle file name elements
 - ◆ EPICS PV name <name>
(CALN_VBD_01)
 - ◆ Run type (STD) <rtype>
 - ◆ Extension (.pcl)
- Template
 $<\text{name}>:<\text{rtype}>.\text{pcl}$
- Example
 - ◆ **CALN_VBD_01:CALIB.pcl**



March 27, 2001

Generating a Pickle File

```
# Python program to write a pickle file
```

```
import sys, pickle
```

```
# Construct the data table
```

```
table = [0, 1, 2, 3, 4]
```

```
# Construct the pickle file name
```

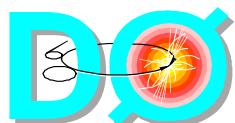
```
dirName = '/online/comics/muo/'  
devName = 'MUO_HV_01:CALIB'  
fileName = dirName + devName + '.pcl'
```

```
# Open the pickle-format file
```

```
file = open(fileName, 'w')  
p = pickle.Pickler(file, 1)
```

```
# Write the table to the file
```

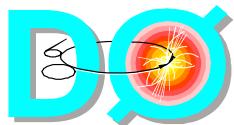
```
p.dump(table)  
file.close()
```



March 27, 2001

COMICS

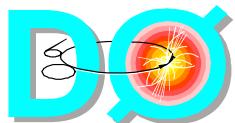
- Manages the configuration of the detector
- Coded in the Python language
- Receives sector load requests from COOR
 - ◆ A sector is the smallest detector component managed by COOR
 - ◆ Sectors may be shared by different runs if the sector configurations match



March 27, 2001

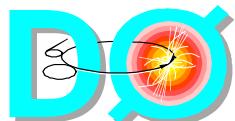
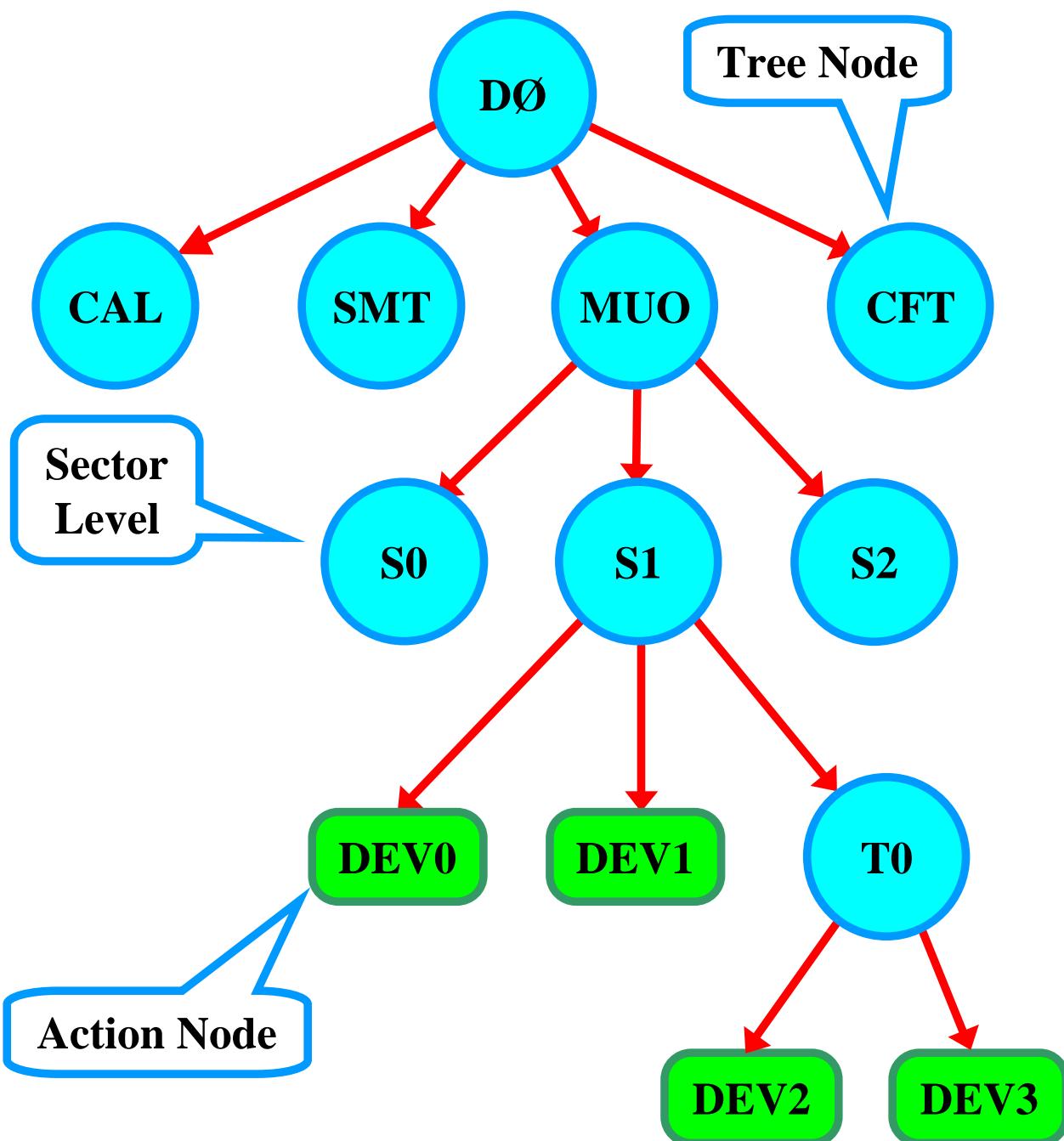
COMICS

- COMICS may be activated independently for configuring detector components
- Download map is a directed graph (tree)
 - ◆ Tree node (intermediate)
 - ◆ Action node (leaf)
- COOR run parameters
 - ◆ Encapsulated in a ComicsRunParams object
 - ◆ Passed to all nodes



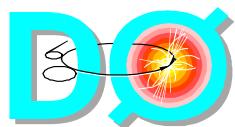
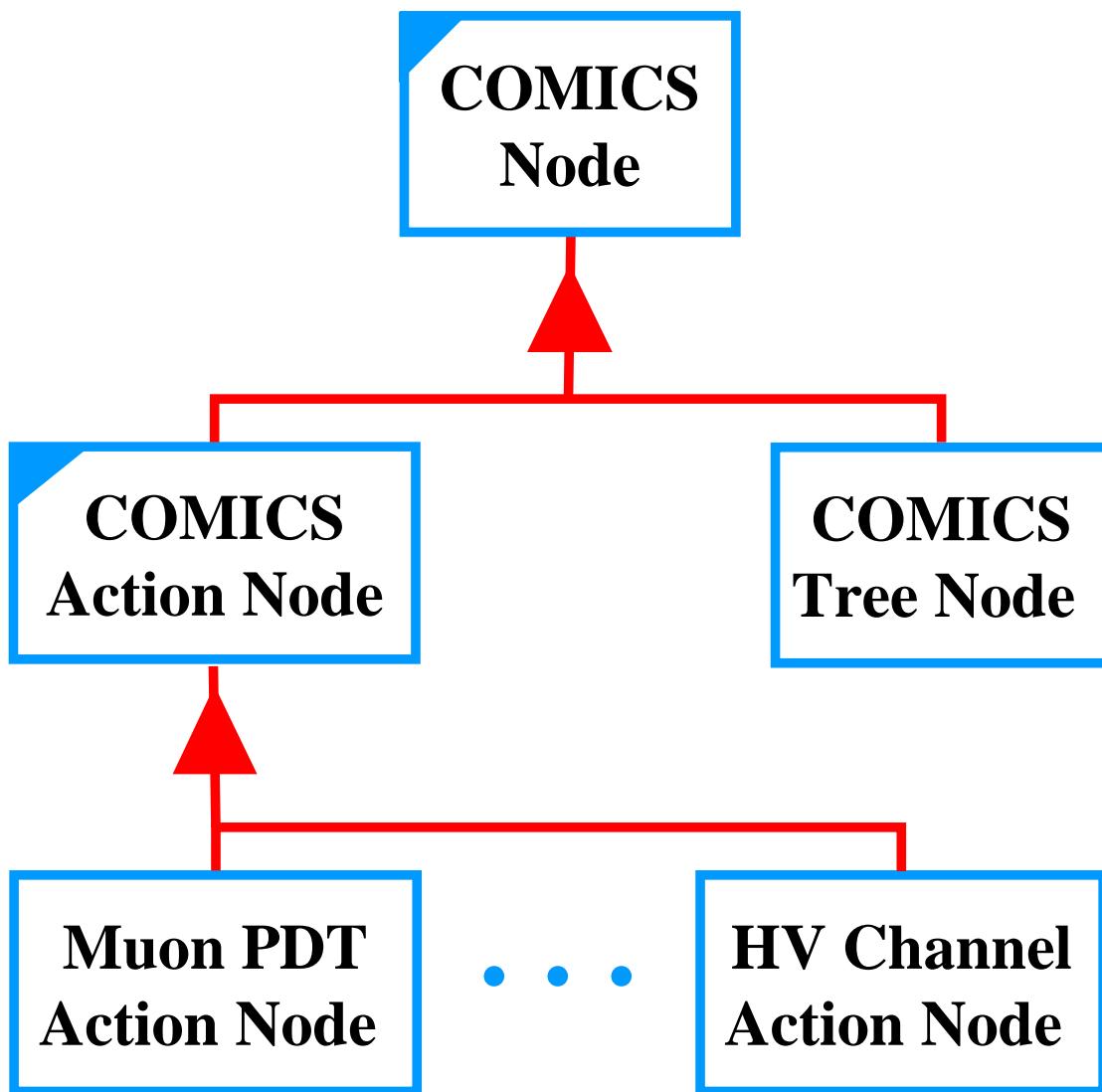
March 27, 2001

Download Tree Structure



March 27, 2001

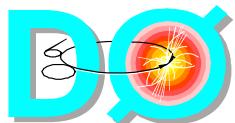
Class Inheritance Diagram



March 27, 2001

ComicsNode Base Class

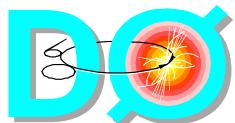
- Abstract class
- Inherited by:
 - ◆ ComicsTreeNode
 - ◆ ComicsActionNode
- Public methods
 - ◆ nameGet - returns current node name
 - ◆ nodeFindByName - returns named node object



March 27, 2001

ComicsTreeNode Class

- Purpose
 - ◆ Define a download tree structure
- Public methods
 - ◆ treeShow - display a sub-tree structure
 - ◆ nodeAdd - add a dependant node
 - ◆ load - download a sub-tree
 - Passes a ComicsRunParams object
- Recursive descent : depth first, left to right

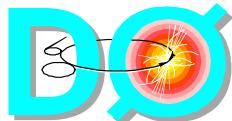


March 27, 2001

COMICS Download Tree

Definition

```
from ComicsNode import *
try:
    root = ComicsTreeNode('D0', None)
    cal = ComicsTreeNode('CAL', root)
        # ... Calorimeter sub-tree
    smt = ComicsTreeNode('SMT', root)
        # ... SMT sub-tree
    muo = ComicsTreeNode('MUO', root)
    s0 = ComicsTreeNode('MUO.S0', muo)
        # ... S0 sub-tree
    s1 = ComicsTreeNode('MUO.S1', muo)
    dev0 = ComicsHvc('MUO.DEV0', s1, 'MUO', '00')
    dev1 = ComicsHvc('MUO.DEV1', s1, 'MUO', '01')
    t0 = ComicsTreeNode('MUO.T0', s1)
    dev2 = ComicsHvc('MUO.DEV2', t0, 'MUO', '02')
    dev3 = ComicsHvc('MUO.DEV3', t0, 'MUO', '03')
    s2 = ComicsTreeNode('MUO.S2', muo)
        # ... S2 sub-tree
    smt = ComicsTreeNode('CFT', root)
        # ... CFT sub-tree
except RuntimeError, info:
    print 'Tree definition error - %s' % str(info)
    sys.exit()
# Show the tree structure
root.treeShow()
```



March 27, 2001

Output from treeShow

Method

D0

CAL - Empty subtree

SMT - Empty subtree

MUO

MUO.S0 - Empty subtree

MUO.S1

MUO.DEV0 - Device MUO_HVC_00

MUO.DEV1 - Device MUO_HVC_01

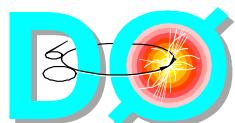
MUO.T0

MUO.DEV2 - Device MUO_HVC_02

MUO.DEV3 - Device MUO_HVC_03

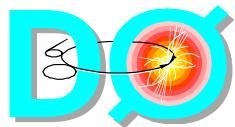
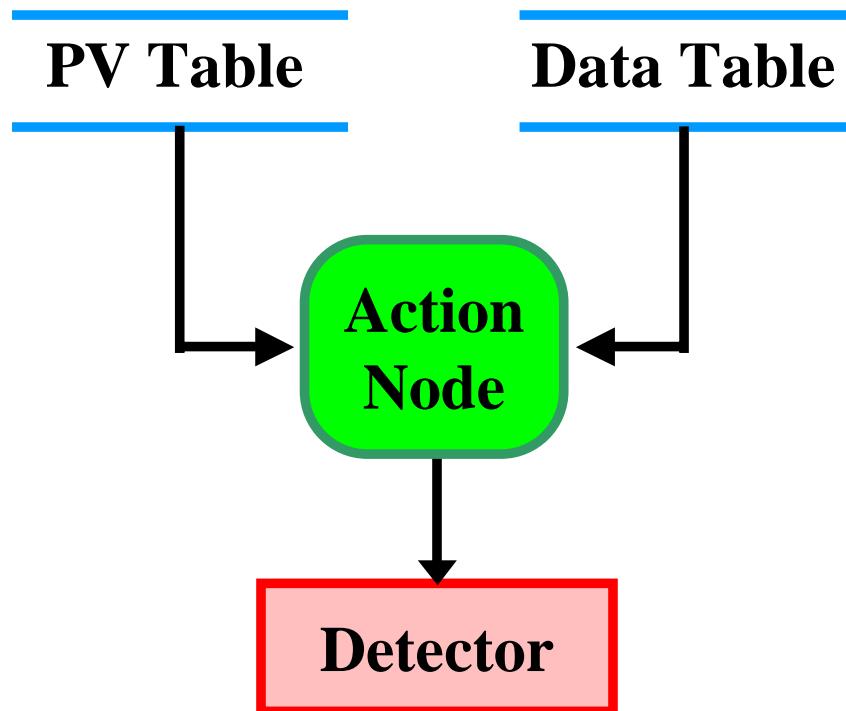
MUO.S2 - Empty subtree

CFT - Empty subtree



March 27, 2001

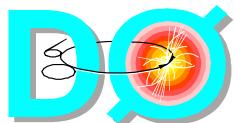
COMICS Action Node



March 27, 2001

ComicsActionNode Class

- **Abstract class**
- **Terminal node of the load tree**
- **Purpose:** download a specific device
- **Actions are table-driven**
 - ◆ PV table
 - ◆ Data table
- **Public methods**
 - ◆ **treeShow** - display this node
 - ◆ **load** - load a specific device

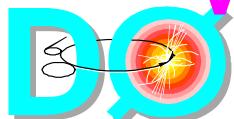


March 27, 2001

ComicsActionNode

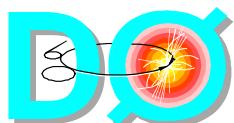
Class

- **Private methods**
 - ◆ **connect - create links to all of the process variables in the PVT**
 - ◆ **disconnect - destroy links to all of the process variables in the PVT**
 - ◆ **pvPut - write the values in the DT to the process variables in the PVT**
- **Virtual methods**
 - ◆ **dbGet - return the Data Table (DT)**
 - ◆ **pvGet - return the Process Variable Table (PVT)**



Sources for the Action Node Data Table

- Action node class variables
 - ◆ Assigned at coding time
 - ◆ Common to all instances of the class
- Action node instantiation variables
 - ◆ Assigned at tree-build time
 - ◆ Unique to instance of class
- Input file (pickle-format)
 - ◆ Assigned at calibration time
- Run parameters
 - ◆ Assigned at run time



March 27, 2001

Action Node for HV Device

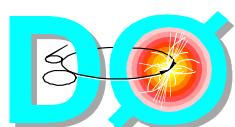
```
# Python program to load a Hv device

from ComicsNode import *

class ComicsHvc(ComicsActionNode):

    # Process variable table (PVT)
    PvTable = [
        ['/RATE', 1],
        ['/VTOL', 1],
        ['/MAXC', 1],
        ['/CSCAL', 1]
    ]

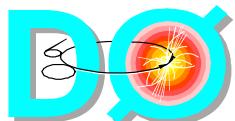
    # Data table (DT)
    DataTable = [
        [123, 2.0, 150, 1.5],
        [150, 5.0, 50, 2.0]
    ]
```



March 27, 2001

Action Node for HV Device

```
def __init__(self,  
            nodeName,  
            parent,  
            detName,  
            locName):  
  
    ComicsActionNode.__init__(self, nodeName, parent,  
                             detName, 'HVC', locName)  
    return  
  
def pvGet(self):  
  
    return ComicsHvc.PvTable  
  
def dbGet(self,  
         runParams):  
  
    runType = runParams.paramGet('RUNTYPE')  
    if (runType == 'STD'):   
        return ComicsHvc.DataTable[0]  
    elif (runType == 'CALIB'):   
        return ComicsHvc.DataTable[1]  
    else:  
        raise RuntimeError, 'Invalid run type'
```



March 27, 2001