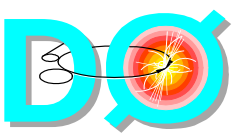
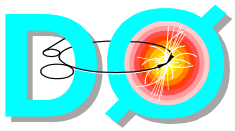
Detector Configuration Management

J. Frederick Bartlett



Outline - Part I

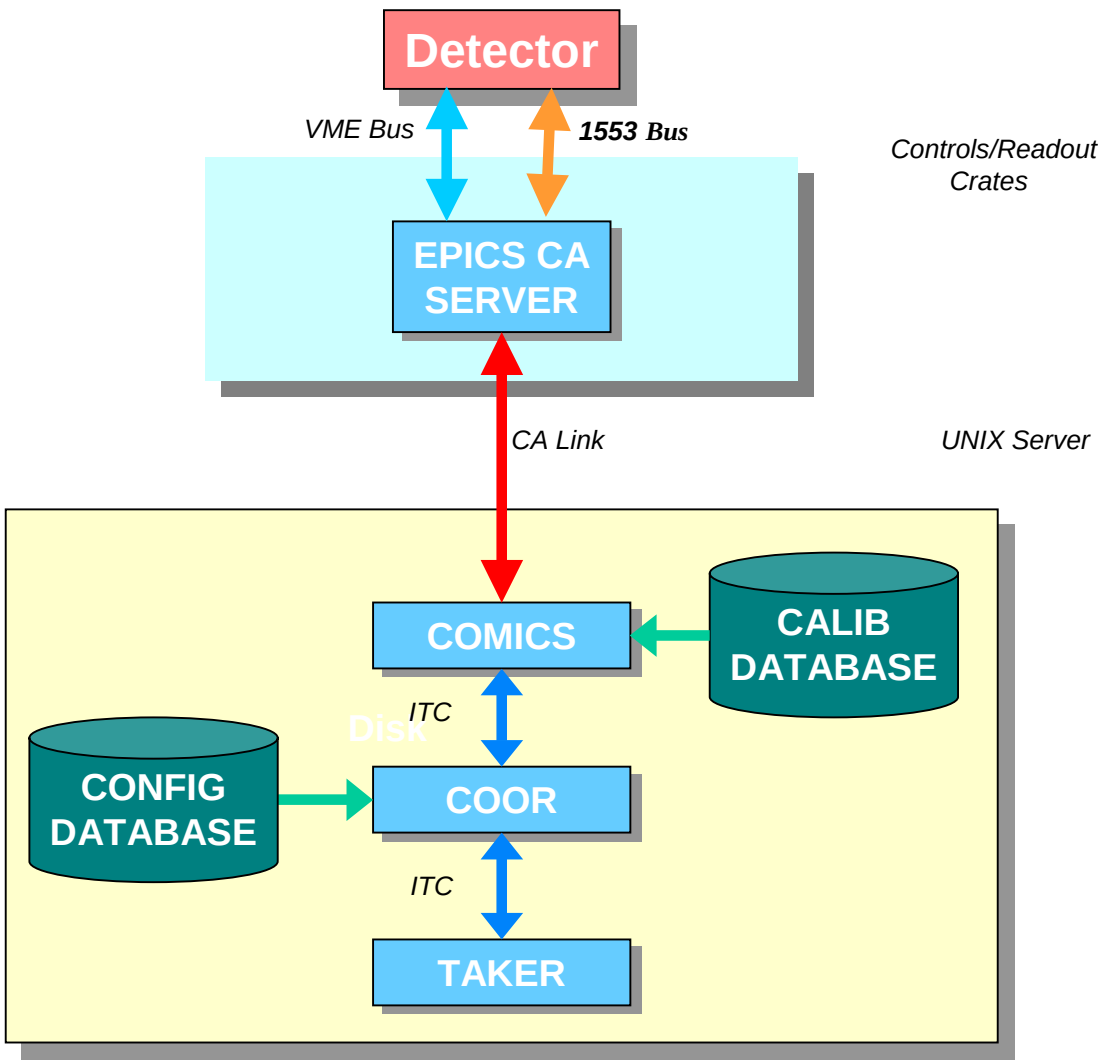
- Overview of the detector download system components
  - The relationship between COMICS and COOR
  - The relationship between COMICS and EPICS
  - The relationship between COMICS and ORACLE calibration databases



Outline - Part II

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

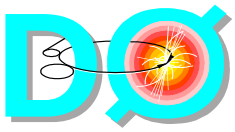
Download Overview



January 24, 2002

Download System Components

- TAKER
  - Operator interface for COOR
- COOR
  - Run state manager
  - Maintains allocation state for assignable detector components
- COMICS
  - Detector load utility
- EPICS
  - Distributed control system



- **Why not use an existing save/restore utility?**
  - **The combination of all possible trigger states and run conditions very large**

EPICS PV Naming Convention

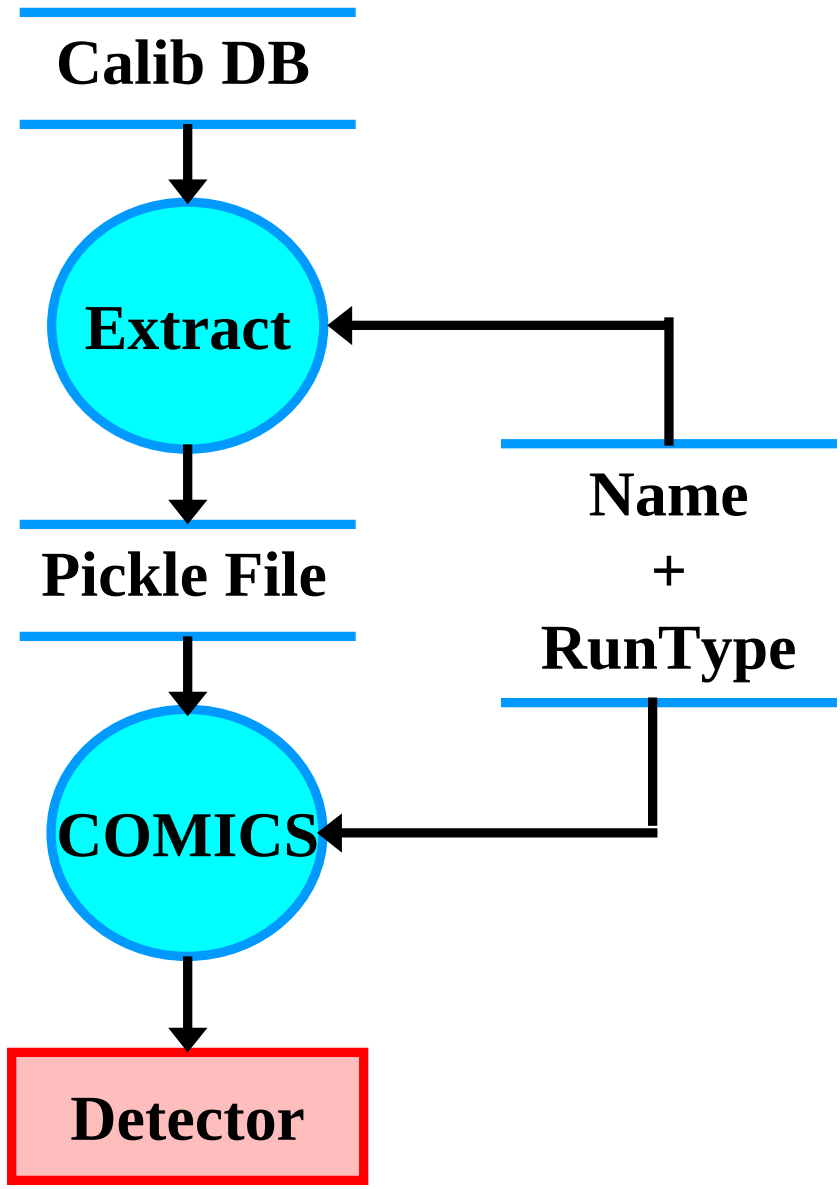
- **Name elements**
  - Detector <det> CAL
  - Sub-det <sub> N
  - Device <dev> VBD
  - Locator <loc> 012
  - Attribute <attr> STATUS
  - I/O <io> R
  - Field <field> SCAN
- **Template**

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

- **Example**

CALN_VBD_012/STATUS:R.SCAN

ORACLE Database Extraction



January 24, 2002

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

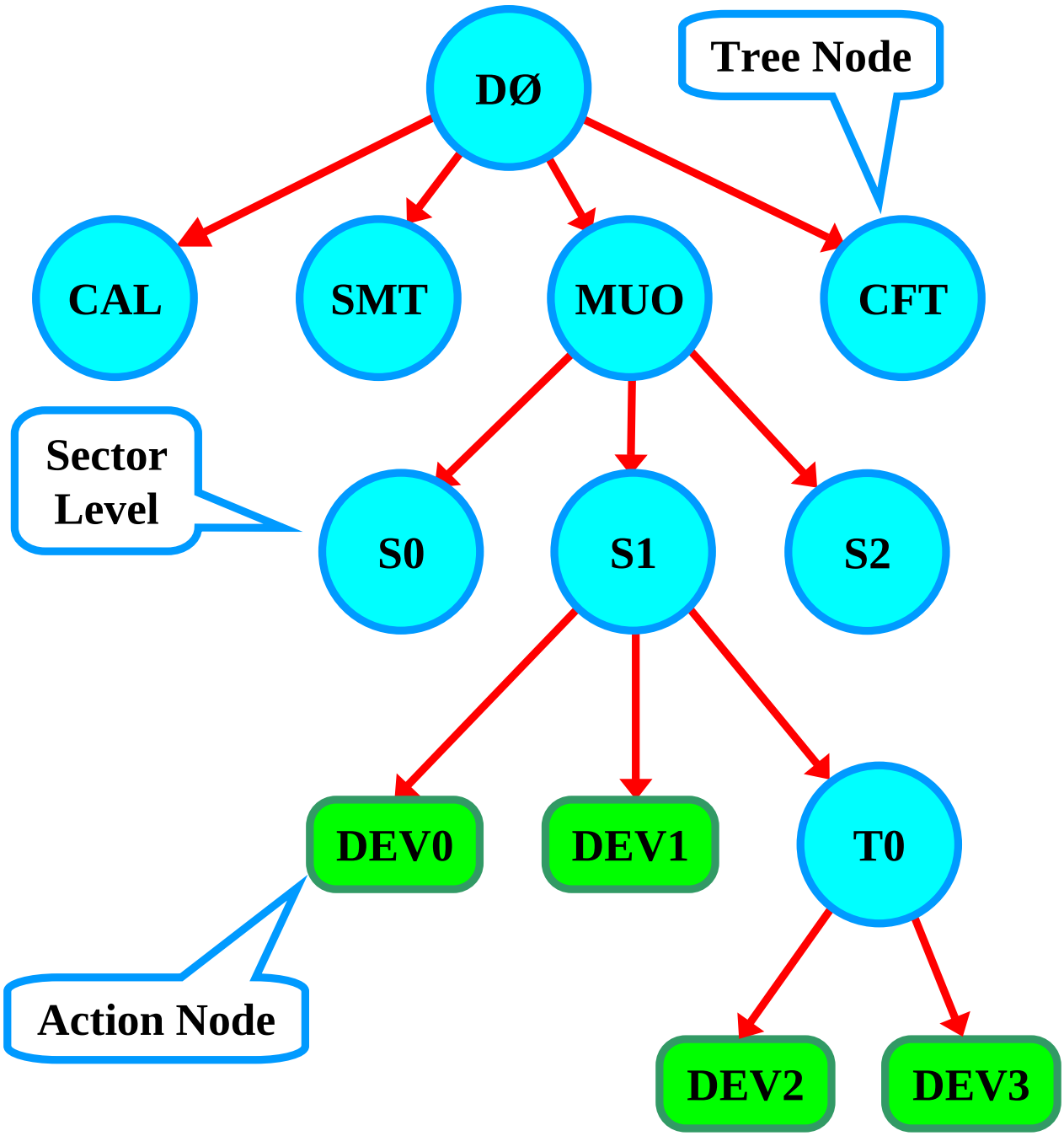
COMICS

- Download map is a directed graph (tree)
  - Tree node (intermediate)
  - Action node (leaf)
- Run parameters
  - Encapsulated in a ComicsRunParams object
  - Passed to all nodes
  - Components
    - Node name – base of the load sub-tree
    - Name/Value pairs
      - Value may be:
        - String
        - Integer number
        - Floating point number

Comics

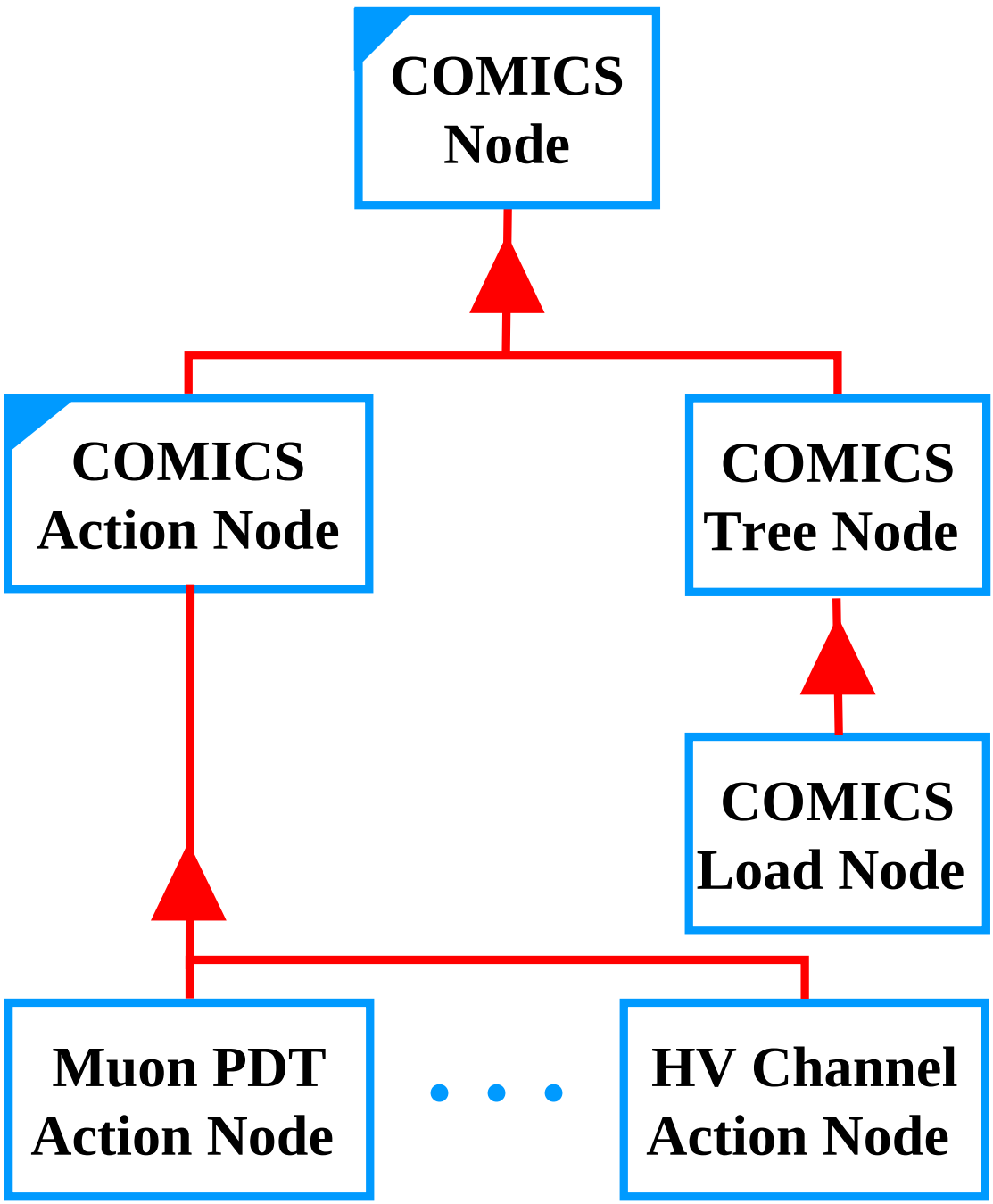
- Constructed on the server model with multiple clients sending commands
- May be activated by separate operator utility for independently configuring detector components
  - Used primarily for testing and diagnostics

Download Tree Structure



January 24, 2002

Class Inheritance Diagram



January 24, 2002

ComicsNode Base Class

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

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

COMICS Download Tree Definition

```
from ComicsNode import *

root = ComicsTreeNode('D0', None)
cal = ComicsTreeNode('CAL', root)
  # … Calorimeter sub-tree
smt = ComicsTreeNode('SMT', root)
  # … SMT sub-tree
muo = ComicsTreeNode('MUO', root)
s0 = ComicsTreeNode('S0', muo)
  # … S0 sub-tree
s1 = ComicsTreeNode('S1', muo)
dev0 = ComicsHvc('DEV0', s1, 'MUO', '00')
dev1 = ComicsHvc('DEV1', s1, 'MUO', '01')
t0 = ComicsTreeNode('T0', s1)
dev2 = ComicsHvc('DEV2', t0, 'MUO', '02')
dev3 = ComicsHvc('DEV3', t0, 'MUO', '03')
s2 = ComicsTreeNode('S2', muo)
  # … S2 sub-tree
smt = ComicsTreeNode('CFT', root)
  # … CFT sub-tree
```

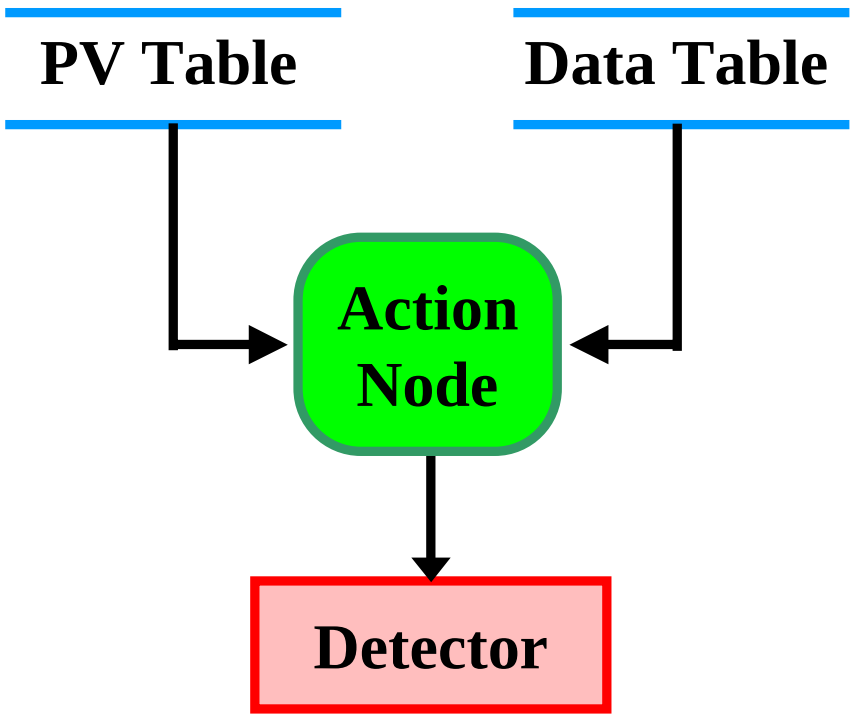
ComicsActionNode Class

- 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

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)

COMICS Action Node



January 24, 2002

Sources for the 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

Action Node for HV Device

```
# Python program to load a Hv device

from ComicsNode import *

class ComicsHvc(ComicsActionNode):

  # Process variable table (PVT)
  PvTable = [
                  ['/RATE', PvLoad, 1],
                  ['/VTOL', PvLoad, 1],
                  ['/MAXC', P vLoad,1],
                  ['/CSCAL', PvLoad, 1],
                  ['/STATE', PvVerify, 1]
                 ]

  # Data table (DT)
  DataTable = [
                    [123, 2.0, 150, 1.5, None],
                    [150, 5.0, 50, 2.0, None]
                   ]
```

January 24, 2002

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'
```