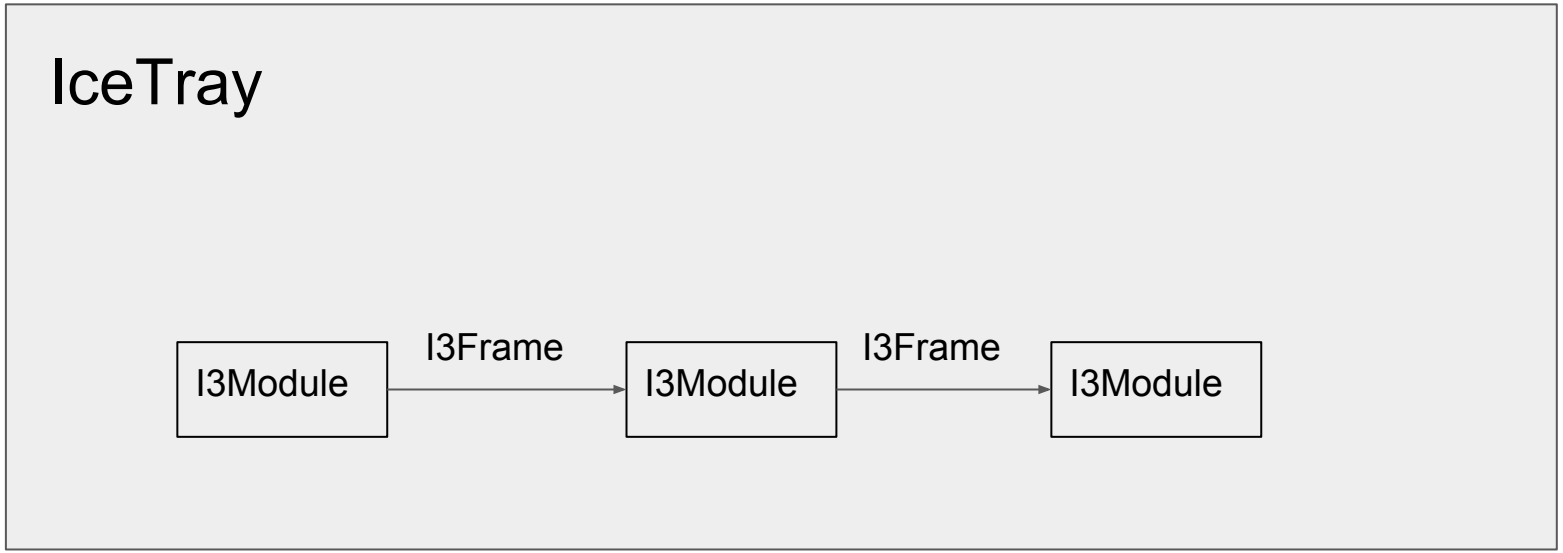
Modern IceTray

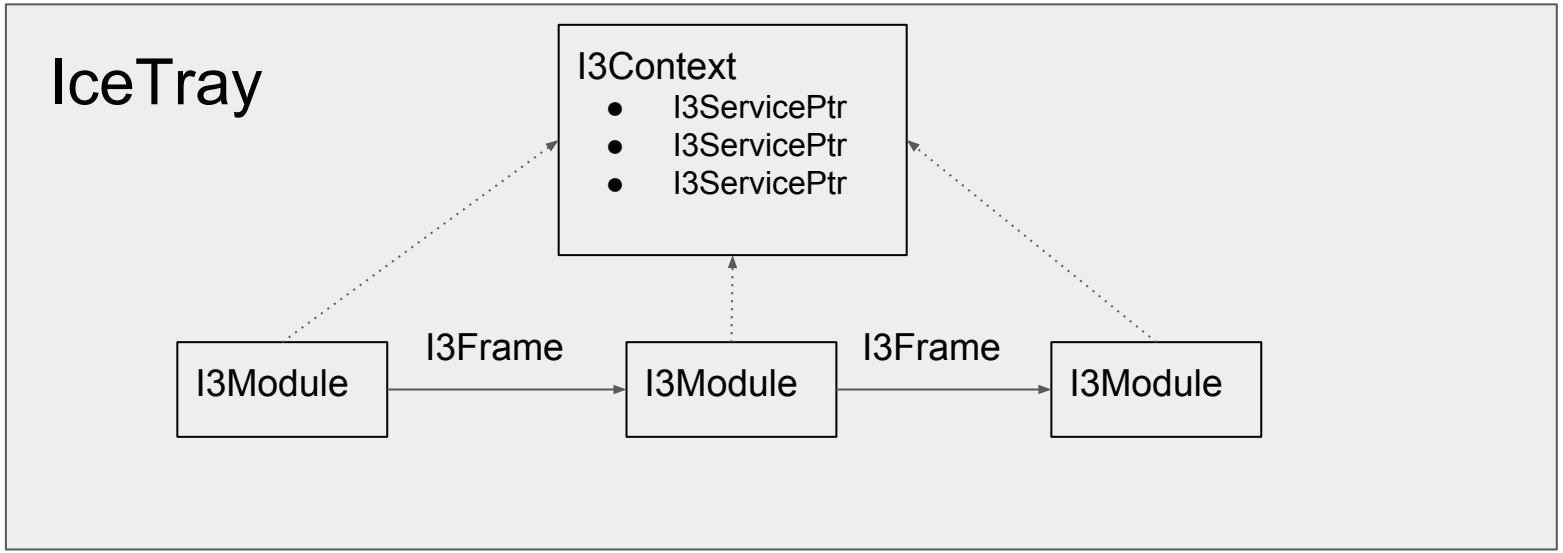
Alex Olivas

IceTray



This model still applies to both old and Modern IceTray.

IceTray



This model still applies to both old and Modern IceTray.

IceTray : Frames come in different flavors

I3Frame Types

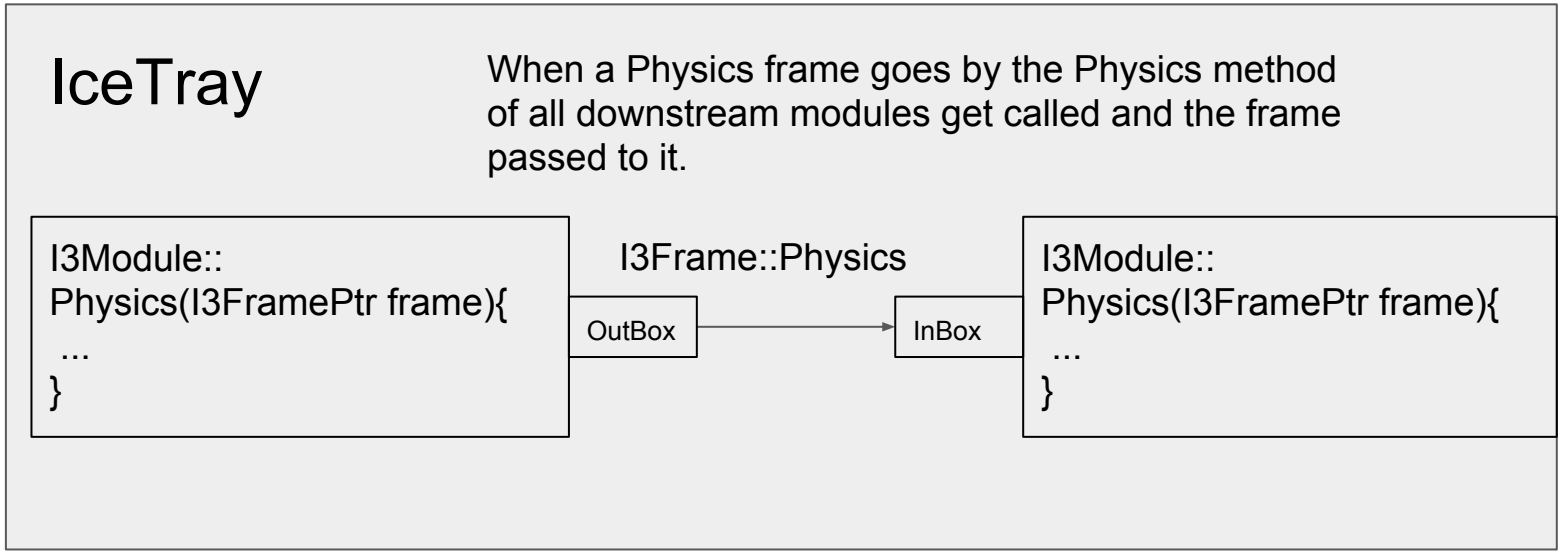
I3Frame::TrayInfo
I3Frame::Geometry
I3Frame::Calibration
I3Frame::DetectorStatus
I3Frame::Physics
I3Frame::DAQ

I3Module ‘Stops’

I3Module::Geometry
I3Module::Calibration
I3Module::DetectorStatus
I3Module::Physics
I3Module::DAQ

Modern IceTray added DAQ frame (aka “Q” frames)
Contains Trigger-level objects

IceTray : Frame-Stream-Stop Model



This model still applies to both old and Modern IceTray.

IceTray : Pre-2008

```
import sys
from I3Tray import *
load('libicetray')
load('libdataclasses')
load('libdataio')

tray = I3Tray()
tray.AddModule('I3Reader','reader')(('Filenamelist',sys.argv[1:]))
tray.AddModule('Dump','dumper')
tray.AddModule('TrashCan','can')
tray.Execute()
tray.Finish()
```

IceTray : Pre-2012

```
import sys
from I3Tray import *
from icecube import icetray,dataclasses,dataio,phys_service

tray = I3Tray()
tray.AddService('I3GSLRandomServiceFactory,'gsl',Seed=42)
tray.AddModule('I3Reader','reader',Filenamelist=sys.argv[1:])
tray.AddModule('Dump','dumper')
tray.AddModule('TrashCan','can')
tray.Execute()
tray.Finish()
```

IceTray : Post-2013

```
import sys
from I3Tray import *
from icecube import icetray,dataclasses,dataio,phys_service

tray = I3Tray()
tray.Add('I3GSLRandomServiceFactory,'gsl',Seed=42)
tray.Add('I3Reader',Filenamelist=sys.argv[1:])
tray.Add('Dump')
tray.Execute()
tray.Finish()
```

IceTray : Simplest IceTray Chain

```
#!/usr/bin/env python
from I3Tray import I3Tray
from icecube import icetray, dataio, dataclasses

tray = I3Tray()
tray.Add('I3InfiniteSource')
tray.Add('Dump')
tray.Finish()
tray.Execute(10)
```

IceTray : Functions as ‘Modules’

```
#!/usr/bin/env python
from I3Tray import I3Tray
from icecube import icetray, dataio, dataclasses

def generator(frame):
    frame["tree"] = dataclasses.I3MCTree()

tray = I3Tray()
tray.Add('I3InfiniteSource')
tray.Add(generator, streams=[icetray.I3Frame.DAQ])
tray.Add('Dump')
tray.Finish()
tray.Execute(10)
```

IceTray : Lambda as ‘Modules’

```
#!/usr/bin/env python
from I3Tray import I3Tray
from icecube import icetray, dataio, dataclasses

def generator(frame):
    frame["tree"] = dataclasses.I3MCTree()

tray = I3Tray()
tray.Add('I3InfiniteSource')
tray.Add(generator, streams=[icetray.I3Frame.DAQ])
tray.Add(lambda frame : frame.Has("tree"), streams=[icetray.I3Frame.DAQ])
tray.Add('Dump')
tray.Finish()
tray.Execute(10)
```

Segments

```
#!/usr/bin/env python
from I3Tray import I3Tray
from icecube import icetray, dataio, dataclasses

@icetray.traysegment
def GeneratorSegment(tray, name):
    def generator(frame):
        frame["tree"] = dataclasses.I3MCTree()

    tray.Add('I3InfiniteSource')
    tray.Add(generator, streams=[icetray.I3Frame.DAQ])
    tray.Add(lambda frame : frame.Has("tree"),
             streams=[icetray.I3Frame.DAQ])

tray = I3Tray()
tray.Add(GeneratorSegment)
tray.Add('Dump')
tray.Finish()
tray.Execute(10)
```

I3Module :
Post-Modern
Classic

```
class ExampleModule(icetray.I3Module):
    def __init__(self, context):
        icetray.I3Module.__init__(self, context)
        self.AddOutBox("OutBox")

    def Configure(self):
        pass
```

I3Module : Post-Modern Classic

```
@icetray.traysegment
def GeneratorSegment(tray, name):
    def generator(frame):
        frame["tree"] = dataclasses.I3MCTree()

    tray.Add('I3InfiniteSource')
    tray.Add(generator, streams=[icetray.I3Frame.DAQ])
    tray.Add(lambda frame : frame.Has("tree"),
             streams=[icetray.I3Frame.DAQ])

class ExampleModule(icetray.I3Module):
    def __init__(self, context):
        icetray.I3Module.__init__(self, context)
        self.AddOutBox("OutBox")

    def Configure(self):
        pass
```

```
tray = I3Tray()
tray.Add(GeneratorSegment)
tray.Add(ExampleModule)
tray.Add('Dump')
tray.Finish()
tray.Execute(10)
```

Services

Two Options

1) Parameter
2) Context

No need for service factories anymore

```
import sys
from icecube import phys_services
class ExampleModule(icetray.I3Module):
    def __init__(self, context):
        icetray.I3Module.__init__(self, context)
        self.AddParameter("RNG", "I3RandomService", None)
        self.AddOutBox("OutBox")

    def Configure(self):
        self.rng = self.GetParameter("RNG")
        if not self.rng:
            self.rng = self.context["I3RandomService"]
            if not self.rng:
                icetray.logging.log_fatal("No random service!!!")

    def DAQ(self, frame):
        random_number = self.rng.uniform(3.14)
        frame["RandomNumber"] = dataclasses.I3Double(random_number)
        self.PushFrame(frame)
```

```
tray = I3Tray()
tray.context['I3RandomService'] = phys_services.I3GSLRandomService(42)
tray.Add(GeneratorSegment)
tray.Add(ExampleModule)
tray.Add('Dump')
tray.Finish()
tray.Execute(10)
```

Modern Logging

Caveat : Debug and Trace statements are compiled out in most builds. Most defaults builds are either **RelWithAssert** or **RelWithDebInfo**. Build **Debug** if you want to see log_debug and log_trace.

Setting Logging Thresholds

The logging threshold is set by default to LOG_WARN for all modules (i.e. only `log_warn()` and more severe conditions will be logged). If you want to change this global threshold, this can be changed by I3Logger's `set_level()` method. For example, to get more information, the global log threshold can be reduced to LOG_INFO:

```
icetray.set_log_level(icetray.I3LogLevel.LOG_INFO)
```

This can also be changed on a per-logging stream level. For example, to get extremely verbose information just from I3Tray, while leaving all other subsystems at their normal levels:

```
icetray.set_log_level_for_unit('I3Tray', icetray.I3LogLevel.LOG_TRACE)
```