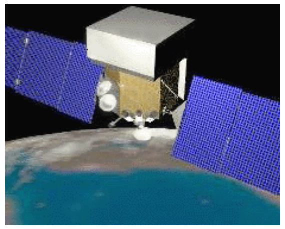
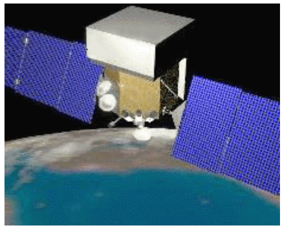


ROOT and Persistency

What Atlas may have to offer

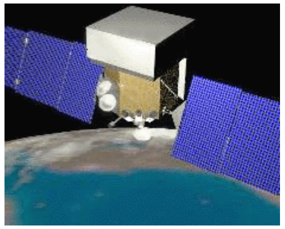


ROOT Features

- Latest version 3.02.07 – released last week.
- Self-documenting - TStreamerInfo

  Will be able to read ROOT files today and 15 years from now.
- Schema Evolution

  Useful if the classes do not change dramatically.
- ROOT team has pledged backward & forward compatibility.



ROOT features contd.

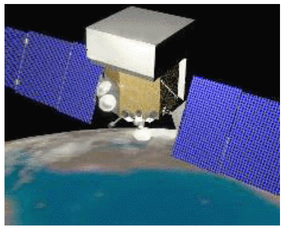
- TRef, TRefArray

  A new class that allows persistent representation of a link/pointer to TObject.

  Useful for MC data, for example.
- Currently we have been storing our data in TTrees
- TTrees provide branched I/O

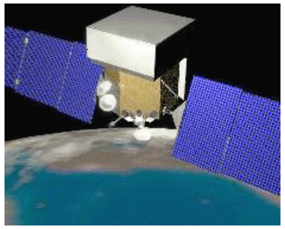
  Read in only that portion of the data that is desired.

GLAST



What is a Persistency Service?

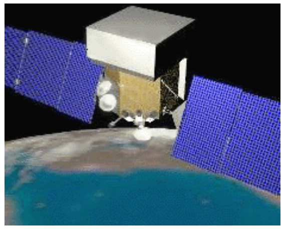
- An interface to a persistent form: disk file
- When data is requested by a Gaudi Algorithm – and it is not available on the TDS – an appropriate Persistency Service is used to retrieve the data from a file.
- At the end of a Gaudi event, if one has requested to write out data, the Persistency Service is used to add data to a disk file.



Persistency Service

- A separate persistency service is required for each format we support, i.e. ROOT, FITS…
- Some basic functionality is required:
  - Gaudi I/O service that provide an interface to the particular I/O library.
  - Gaudi converters that will handle reading data in from its persistent form, storing that data on the TDS as well as taking data off the TDS and writing that data out in some persistent format.

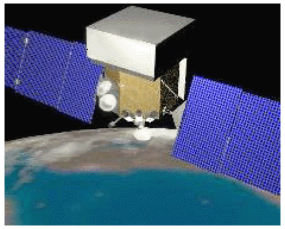
GLAST



A Quick Word about Ntuples

- Ntuples are currently written using the Gaudi NtupleSvc and our own NtupleWriterSvc.

  Ntuples are memory resident until the end of a run
- Handles Row-Wise Ntuple (RWN) creation.
- Use requires:
  tag name of an ntuple item
  corresponding value
  output file id



ROOT Persistency Service

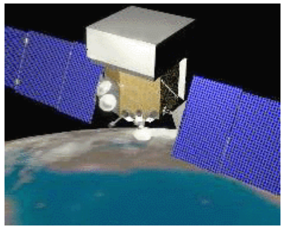
- Atlas has produced a "real" ROOT service.
  http://www.usatlas.bnl.gov/computing/software/db/rootio.html
  - ROOT I/O
  - ROOT interactive session by demand
  - ROOT share library dynamic loading by demand
  - ROOT control over the Gaudi algorithms
- Can we use what they have?

  Will we be able to benefit from future development?

  Should we just take what they have and run?

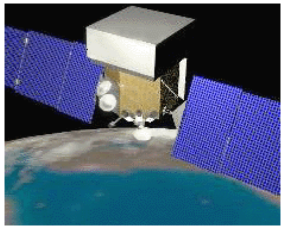
  Do we use their code as a model and run?

GLAST



Taking a look

- Obtained and compiled code with our existing version of the Gaudi checkout package GaudiSys v7.
- Works with ROOT v3.01.06
- The code is light, should not require much (or any) modification when upgrading to Gaudi v9 or ROOT 3.02.07

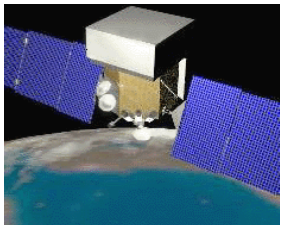


Status

- RootKernel – defines the basic ROOT classes
  - TDataSet, TTable, TObjectSet
  - This package is independent of Gaudi.
- RootSvcModules
  - ROOT I/O
  - ROOT interactive session by demand
  - ROOT share library dynamic loading by demand
  - ROOT control over the Gaudi algorithms
- EventSelectorGaudiRoot

  An interface to the RootSvc, loading events on demand.

GLAST



TDataSet

♦ Assumes organization of data in TDataSets.

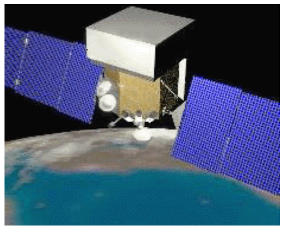
```
                        event  dataset
                                 |
               __________________|____________..._______
               |            |            |              |
          sub-dataset  sub-dataset     table        sub_dataset
          _____|__...__
          |    |       |
       table  table    table
```

• Each TDataSet contains objects derived from TDataSet such as TTable, TObjectSet.

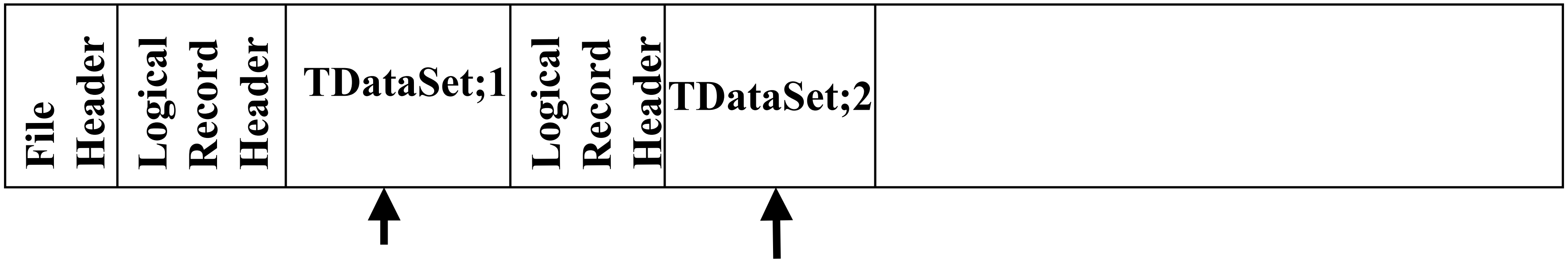
-TTable uses a C structure to define a table.

-TObjectSet stores a ptr to a TObject.

- Requires a specialized converter for each class.



What would a ROOT file look like?

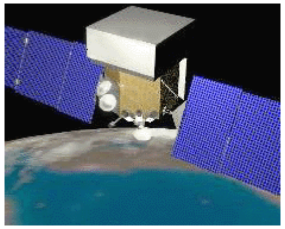


Data for Event #1 Data for Event #2

- Does not use the TTree hierarchy.

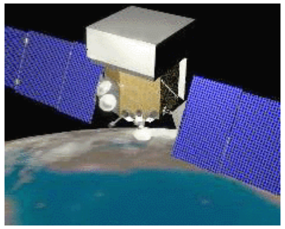
  Hence, we cannot start using this service immediately.

GLAST



Can we use it?

- Use of TTables would mean our persistent data would probably not mirror the data structure on the TDS
  - TRef not supported
  - One the positive side, it would make mapping to FITS files a breeze.
- Using TObjectSet would allow us to define our own classes…and mirror the class structure on the TDS
  - - However without TTree we would lose branch IO.



Can we use it? Contd.

- Using it, as is, means placing restrictions on our classes.
  - There is no free lunch!
- May be able to work in conjunction with the Atlas team to extend their code.
- Or we can use the current code as a model.
  - It does provide a nice example of a ROOT service
  - Create our own specific converters – which we'd have to do anyway if we want to support TObjects.