Unidata

Providing data services, tools, & cyberinfrastructure leadership that advance Earth system science, enhance educational opportunities, & broaden participation

NetCDF and Scientific Data Durability

Russ Rew, UCAR Unidata

ESIP Federation Summer Meeting

2009-07-08

For preserving data, is format obsolescence a non-issue?

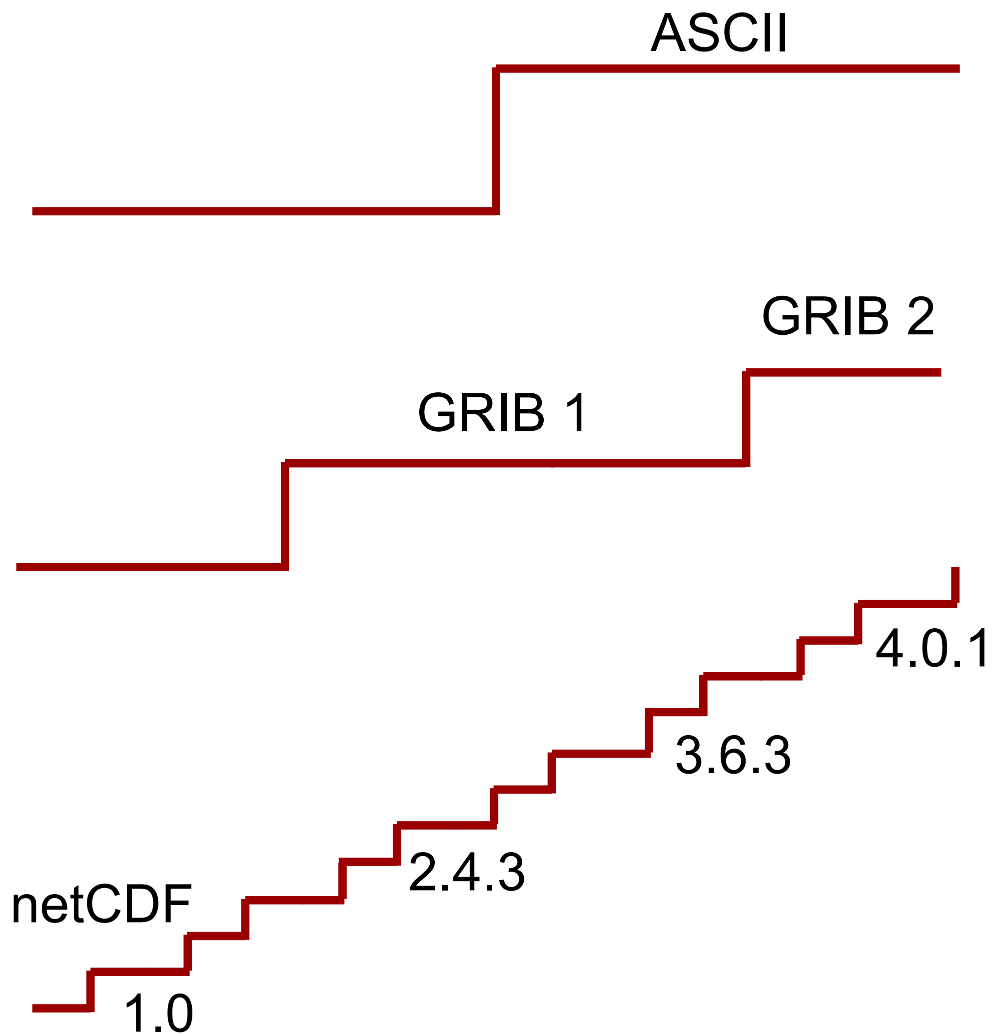
- Why do formats (and their access software) change?
- Are format changes consistent with data preservation and stewardship?
- How can the evolution of formats support preservation and stewardship?
- What principles should guide format developers in support of data durability?
- What are the most important threats to netCDF data in archives?

Why do open formats and their supporting software libraries change?

- To better represent data semantics
  - Capturing intent of data providers
  - Exploiting metadata advances, new conventions
- To improve performance, avoid obsolescence
  - Compression, caching, chunking, indexing, …
  - Parallel file systems
- To enhance interoperability
  - Replacing specialized formats with more general formats
- To fix mistakes
  - 32-bit offsets for data in files
  - ASCII characters for all metadata
- To respond to users' needs

How do formats change?

- Simple formats don't change, they're defined once and frozen forever
- Some formats change infrequently and usually incompatibly
- Complex formats (and their software) may evolve in lots of small increments



NetCDF: not just a format

- A **standard format** for platform-independent data (NASA ESDS-RFC-011)
- CF-netCDF is being proposed as a formal OGC binary encoding standard

But netCDF is also

- A **data model** for multidimensional and structured scientific data
- A **set of application programming interfaces** (C, Java, Fortran, C++, …) for data access
- A **reference implementation** for the APIs

How has netCDF changed?

Software	Formats	Features
2009: 4.1	netCDF-4 64-bit offset classic	OPeNDAP client support, integration/inclusion of udunits and libcf, improved HDF5 and HDF4 support
2008: 4.0	netCDF-4 64-bit offset classic	Enhanced data model, expanded APIs, HDF5 storage layer, compression, chunking, parallel I/O, Unicode names, …
2006: 3.6	64-bit offset classic	NcML, 64-bit offset format
2001: 3.5	classic	new Java API, Fortran-90 API
1998: 3.4	classic	Java API, limited large file support, performance enhancements
1997: 3.3	classic	C, F77 “version 3” type-safe APIs
1996: 2.4	classic	C++ API, optimizations, format spec published
1989: 1.0	classic	C, F77 APIs

Ways to deal with format changes

- Use only published standards for archives
  - Format standardization is *slow*
    - GRIB1 (1985) to GRIB2 (2001)
  - Impractical if many intermediate versions (e.g CF Conventions 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, …)
- Convert archived data periodically
  - Upgrading older formats is costly, risky
  - Migrating to a more general format may break older access software
- Save data access software versions with data
  - Requires data archives to become software version control repositories
  - Imposes often unnecessary burden on data access
- Rely on a commitment to compatibility by format developers, maintainers, and responsible organization

Compatibility commitment

- For scientific data, preserving access to data for future generations should be sacrosanct
- Strong commitment is needed to ensure practical access to old data by new programs
- Careful library evolution can ensure data and API compatibility
- An example public commitment presented at American Meteorological Society annual meeting, January 2006 …

Declaration of Compatibility

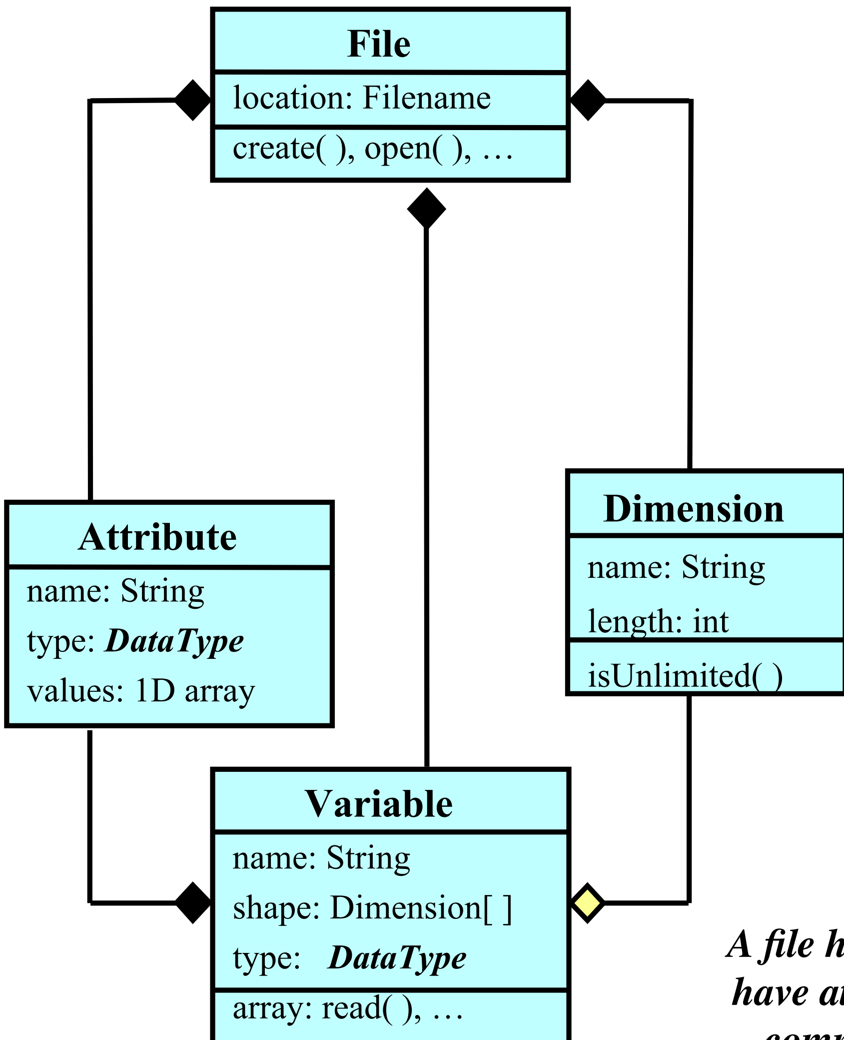
For future access to archives, netCDF development will continue to ensure the compatibility of:

- **Data access:** netCDF software will provide both read and write access to all earlier forms of netCDF data.
- **Programming interfaces:** C and Fortran programs using documented netCDF APIs from previous versions will continue to work after recompiling and relinking (if needed).
- **Future versions:** netCDF will continue to support both data access compatibility and API compatibility in future releases.

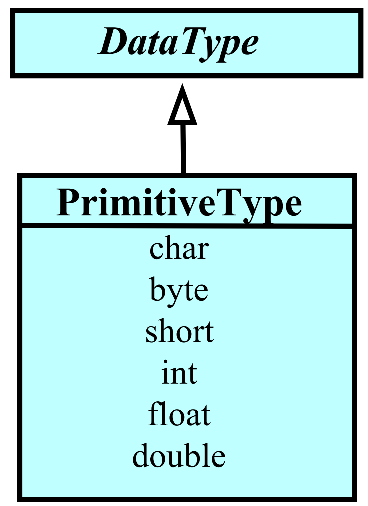
Aspects of compatibility

- **Costs**
  - Effort to support older interfaces and formats
  - Comprehensive compatibility testing with every software release
- **Benefits**
  - Data in archives don't have to change
  - Client program sources don't have to change
  - Software can access archived data without being aware of format version
- Implemented by compatibly evolving data model
  - Add or grow abstractions, instead of replacing them
  - Ensure previous data model is included in enhanced data model

Classic netCDF data model



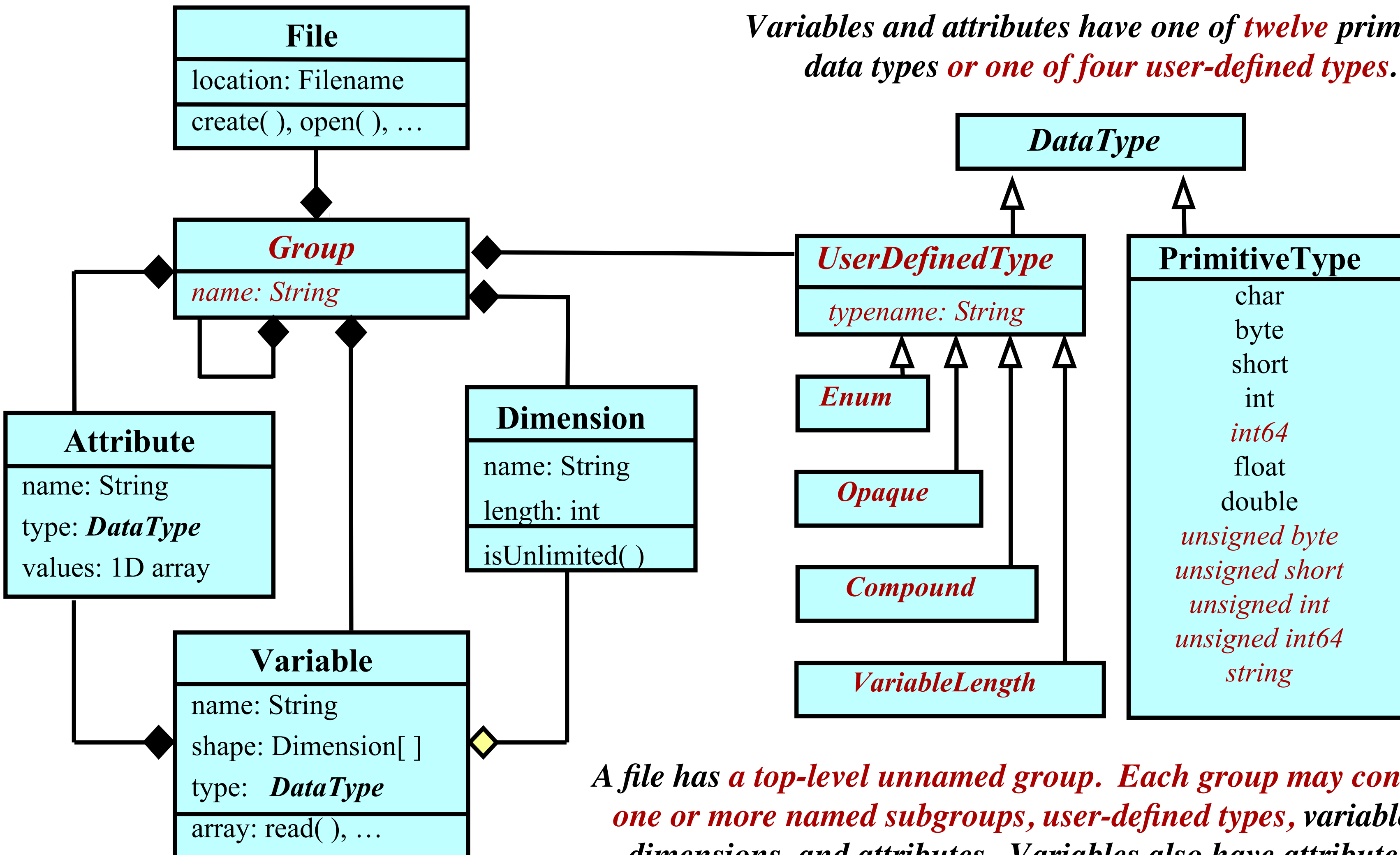
Variables and attributes have one of six primitive data types.



A file has variables, dimensions, and attributes. Variables also have attributes. Variables may share dimensions, indicating a common grid. One dimension may be of unlimited length.

Enhanced netCDF data model, for netCDF-4

Variables and attributes have one of twelve primitive data types or one of four user-defined types.



A file has a top-level unnamed group. Each group may contain one or more named subgroups, user-defined types, variables, dimensions, and attributes. Variables also have attributes. Variables may share dimensions, indicating a common grid. One or more dimensions may be of unlimited length.

NetCDF-4 classic-model: a transitional format

netCDF-4	• Not compatible with some existing applications • Enhanced data model and API, more complex, powerful
netCDF-4 classic model	• Uses classic API for compatibility • Uses netCDF-4/HDF5 storage for compression, chunking, performance • To use, just recompile, relink
netCDF-3	• Compatible with existing applications • Simplest data model and API

Other ways netCDF supports data durability

- CF Conventions add earth-science specific semantics to low-level data model, without changing format
- Java netCDF reads multiple data formats through an abstract Common Data Model interface
  - HDF4, HDF5, HDF-EOS, GRIB1, GRIB2, BUFR, GEMPAK, GINI, DMSP, NEXRAD, …
- NcML wrappers support efficient addition of new metadata, virtual aggregations
- “history” attribute for provenance automatically maintained by utilities like NCO

Concluding remarks

- Format obsolescence need not be an issue for data durability
- Evolve data models by extension, not by incompatible modification
- Preserve previous programming interfaces
- Support previous format variants transparently
- Avoid gratuitous invention of new formats
- Data preservation and stewardship requires much more than dealing with format evolution
  - Economic failures
  - Organizational failures
  - Operator or administrative errors
  - Hardware problems
  - Software errors