

Short Name	Full Name and the www link	Description
CDF	Common Data Format (http://cdf.gsfc.nasa.gov/)	The National Space Science Data Center's (NSSDC) Common Data Format (CDF) is a self-describing data abstraction for the storage and manipulation of multidimensional data in a discipline-independent fashion. CDF is a scientific data management package (known as the "CDF Library") which allows programmers and application developers to manage and manipulate scalar, vector, and multi-dimensional data arrays. The basic component of CDF is a programming interface that is device independent. Client software library called CSCDF can be used with the CDF library to provide applications access to remote CDF datasets.
FITS	Flexible Image Transport System (http://heasarc.gsfc.nasa.gov/docs/heasarc/fits.html)	FITS is the standard data interchange and archival format of the worldwide astronomy community. It is endorsed both by NASA and IAU. FITS is much more than an image format (such as JPG or GIF) and is primarily designed to store scientific data sets consisting of multi-dimensional arrays (1-D spectra, 2-D images or 3-D data cubes) and 2-dimensional tables containing rows and columns of data. A FITS file consists of one or more Header + Data Units (HDUs), where the first HDU is called the 'Primary HDU', or 'Primary Array'. The primary array contains an N-dimensional array of pixels, such as a 1-D spectrum, a 2-D image, or a 3-D data cube. Five different primary data types are supported: unsigned 8-bit bytes, 16 and 32-bit signed integers, and 32 and 64-bit single or double precision floating point reals. FITS can also store 16 and 32-bit unsigned integers. Any number of additional HDUs may follow the primary array; these additional HDUs are called FITS 'extensions'. There are currently 3 types of extensions defined by the FITS Standard : Image Extension - a N-dimensional array of pixels, like in a primary array, ASCII Table Extension - rows and columns of data in ASCII character format, Binary Table Extension - rows and columns of data in binary representation
GRIB	Grid in Binary (http://www.wmo.int/pages/prog/www/WMOCodes/Guides/GRIB/GRIB1-Contents.html)	GRIB is a general purpose, bit-oriented data exchange format, designated FM 92-VIII Ext. GRIB (GRIdded Binary). It is an efficient vehicle for transmitting large volumes of gridded data to automated centers over high-speed telecommunication lines using modern protocols. By packing information into the GRIB code, messages (or records - the terms are synonymous in this context) can be made more compact than character oriented bulletins, which will produce faster computer-to-computer transmissions. GRIB can equally well serve as a data storage format, generating the same efficiencies relative to information storage and retrieval devices.
HDF	Hierarchical Data Format (http://hdf.ncsa.uiuc.edu/)	The Hierarchical Data Format is a multi-object file format that facilitates the transfer of various types of data between machines and operating systems. It allows self-definitions of data content and is easily extensible for future enhancements or compatibility with other standard formats. The latest versions of HDF supports the complete netCDF interface. The HDF library contains interfaces for storing and retrieving compressed and uncompressed raster images with palettes and an interface for storing and retrieving n-Dimensional scientific datasets together with the information about the data, such as labels, units, formats and scales for all dimensions.
NetCDF	Network Common Data Format (http://www.unidata.ucar.edu/)	The network Common Data Form is an interface for scientific data access and a library that provides an implementation of the interface. It also defines a machine-independent format for representing data. Data stored in the netCDF format is self-describing, network transparent, direct-access, appendable, and sharable. There is a netCDF interface to HDF available. NcML is an XML representation of netCDF metadata, (roughly) the header information one gets from a netCDF file with the "ncdump -h" command. NcML is similar to the netCDF CDL (network Common data form Description Language), except, of course, it uses XML syntax. The CF conventions for climate and forecast metadata are designed to promote the processing and sharing of files created with the netCDF API. The conventions define metadata that provide a definitive description of what the data in each variable represents, and of the spatial and temporal properties of the data. This enables users of data from different sources to decide which quantities are comparable, and facilitates building applications with powerful extraction, regridding, and display capabilities. NetCDF data is: <ul style="list-style-type: none"> • <i>Self-Describing</i>. A netCDF file includes information about the data it contains. • <i>Portable</i>. A netCDF file can be accessed by computers with different ways of storing integers, characters, and floating-point numbers. • <i>Direct-access</i>. A small subset of a large dataset may be accessed efficiently, without first reading through all the preceding data. • <i>Appendable</i>. Data may be appended to a properly structured netCDF file without copying the dataset or redefining its structure. • <i>Sharable</i>. One writer and multiple readers may simultaneously access the same netCDF file. • <i>Archivable</i>. Access to all earlier forms of netCDF data will be supported by current and future versions of the software.
VICAR	Video Image Communication and Retrieval	VICAR is a collection of image processing programs supported by the Multimission Image Processing Laboratory (MIPL) at the Jet Propulsion Laboratory (JPL), for use in manipulating and analyzing spacecraft images. The image format used by VICAR programs, and for all or most data from JPL-managed missions, is referred to as VICAR format.
SAIF	Spatial Archive and Interchange Format (http://s2k-ftp.cs.berkeley.edu:8000/sequoia/schema/html/saif/saif31spec.html)	The Spatial Archive and Interchange Format was developed as a means of sharing spatial and spatiotemporal data. SAIF's primary objectives are as follows: <ol style="list-style-type: none"> 1. it must be appropriate for modelling and moving data in general; that is, it should be able to deal with both spatiotemporal and traditional information, 2. it must handle virtually any kind of geographic data, including those: (i), with or without extensive attribute descriptions, and (ii), with geometry defined by vector or raster structures in two or three dimensions, (e.g., typical topographic, cadastral, and thematic data, but also,

		<p>subsurface geologic data, climate data, hydrographic data, etc.),</p> <p>3. it must address time such that temporal events and relationships can be handled (e.g., moving oil spills, vehicle navigation, general monitoring activities),</p> <p>4. it must address data management requirements (such as support for updates, ability to integrate with multimedia data, applicability to both large and small data volumes, the ability to interface well with database queries, and compatibility with catalogue developments),</p> <p>SAIF is designed to facilitate interoperability, particularly in the context of data exchange. It also represents an efficient means of archiving data in a vendor neutral format. An overriding consideration with SAIF is to be able to treat geographic data as simply another kind of data; that is, the roots of SAIF are in the information sciences.</p>
SDTS	<p>Spatial Data Transfer Standard</p> <p>(http://mcmcweb.er.usgs.gov/sdts/) (http://data.geocomm.com/dem/)</p>	SDTS is a Federal standard for transfer of geologic and other spatial data.
HDS	<p>Hierarchical Data System</p> <p>(http://www.sesp.cse.clrc.ac.uk/Publications/data-management/report/node14.html)</p>	HDS is a freely available database system. It is particularly suited to the storage of large multidimensional arrays, where efficiency of access is a requirement. It is presently used in astronomy for storing images, spectra and time series. The advantage of HDS is that it allows many different kinds of data to be stored in a consistent and logical fashion. It is also very flexible, in that objects can be added or deleted whilst retaining the logical structure. HDS also provides portability of data, so that the same data objects may be accessed from different types of computer despite the fact that each may actually format its files and data in different ways.
OpenMath	<p>(http://www.openmath.org/overview/index.html)</p>	<p>OpenMath is an emerging standard for representing mathematical objects with their semantics, allowing them to be exchanged between computer programs, stored in databases, or published on the worldwide web. There is a strong relationship to the MathML recommendation from the WWC, and a large overlap between the two developer communities. MathML deals principally with the <i>presentation</i> of mathematical objects, while OpenMath is solely concerned with their semantic meaning or <i>content</i>. While MathML does have some limited facilities for dealing with content, it also allows semantic information encoded in OpenMath to be embedded inside a MathML structure. Thus the two technologies may be seen as highly complementary.</p> <p>Mathematical objects encoded in OpenMath can be</p> <ul style="list-style-type: none"> • displayed in a browser • exchanged between software systems • cut and pasted for use in different contexts • verified as being mathematically sound (or not!) • used to make interactive documents really interactive.
DEM	<p>Digital Elevation Model</p> <p>(http://tahoe.usgs.gov/DEM.html) (http://www.oregonexplorer.info/craterlake/dem_brief.html)</p>	<p>A Digital Elevation Model (DEM) is a digital cartographic/geographic dataset of elevations in xyz coordinates. The terrain elevations for ground positions are sampled at regularly spaced horizontal intervals. DEMs are derived from hypsographic data (contour lines) and/or photogrammetric methods using USGS 7.5-minute, 15-minute, 2-arc-second (30- by 60-minute), and 1-degree (1:250,000-scale) topographic quadrangle maps.</p> <p>Many DEMs provided by the USGS use a 30-m grid. In other words, the area covered in the DEM is split into squares with 30-m sides. Hills or valleys smaller than the 30-m cells will not show up.</p>
DLG	<p>Digital Line Graph</p> <p>(http://edc.usgs.gov/guides/dlg.html) (http://tahoe.usgs.gov/DEM.html)</p>	A Digital Line Graph (DLG) is digital vector data representing cartographic information. DLGs contain a wide variety of information depicting geographic features (for example, hypsography, hydrography, boundaries, roads, utility lines, etc). DLGs are derived from hypsographic data (contour lines) using USGS 7.5-minute, 15-minute, 2-arc-second (30- by 60-minute), and 1:2 million-scale topographic quadrangle maps.
VTK file format	<p>Visualization Toolkit</p> <p>(http://www.vtk.org/what-is-vtk.php)</p>	<p>The Visualization ToolKit (VTK) is an open source, freely available software system for 3D computer graphics, image processing, and visualization.</p> <p>VTK file format: The first part is the file version and identifier. This part contains the single line: # vtk DataFile Version x.x. This line must be exactly as shown with the exception of the version number x.x, which will vary with different releases of VTK. (Note: the current version number is 3.0. Version 1.0 and 2.0 files are compatible with version 3.0 files.)</p> <p>2. The second part is the header. The header consists of a character string terminated by end-of-line character \n. The header is 256 characters maximum. The header can be used to describe the data and include any other pertinent information.</p> <p>3. The next part is the file format. The file format describes the type of file, either ASCII or binary. On this line the single word ASCII or BINARY must appear.</p> <p>4. The fourth part is the dataset structure. The geometry part describes the geometry and topology of the dataset. This part begins with a line containing the keyword DATASET followed by a keyword describing the type of dataset.</p>

		<p>Then, depending upon the type of dataset, other keyword/data combinations define the actual data.</p> <p>5. The final part describes the dataset attributes. This part begins with the keywords POINT_DATA or CELL_DATA, followed by an integer number specifying the number of points or cells, respectively. (It doesn't matter whether POINT_DATA or CELL_DATA comes first.) Other keyword/data combinations then define the actual dataset attribute values (i.e., scalars, vectors, tensors, normals, texture coordinates, or field data).</p>
IDL	<p>Interactive data language</p> <p>(http://idlastro.gsfc.nasa.gov/idl_html_help/home.html)</p> <p>(http://www.itvvis.com/)</p>	<p>IDL is vectorized, numerical, and interactive, and it is commonly used for interactive processing of large amounts of data (including image processing). The syntax includes many constructs from Fortran and some from C.</p>
XML	<p>Extensible Markup Language</p> <p>(http://www.w3.org/XML/)</p>	<p>The Extensible Markup Language (XML) is a general-purpose <i>specification</i> for creating custom markup languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet, [2] and it is used both to encode documents and to serialize data. In the latter context, it is comparable with other text-based serialization languages such as JSON and YAML. The first problem is that XML is not a very compact way of representing information. For instance, it would be possible to take an XML document and rework it into a proprietary binary format that would be much smaller even without the use of modern compression techniques. The second problem is that, while XML is easy to read, it is not in an optimal format for a parser to process and interpret. This means that more processing time is spent than may be necessary to get to the information stored in the document.</p>
YAML	<p>Yet Another Markup Language</p> <p>(http://www.yaml.org/)</p> <p>(http://www.yaml.org/spec/)</p>	<p>YAML™ (rhymes with “camel”) is a human-friendly, cross language, Unicode based data serialization language designed around the common native data structures of agile programming languages. It is broadly useful for programming needs ranging from configuration files to Internet messaging to object persistence to data auditing. There are hundreds of different languages for programming, but only a handful of languages for storing and transferring data. Even though its potential is virtually boundless, YAML was specifically created to work well for common use cases such as: configuration files, log files, interprocess messaging, cross-language data sharing, object persistence and debugging of complex data structures. When data is easy to view and understand, programming becomes a simpler task.</p>
SDXF	<p>Structured Data Exchange format</p> <p>(http://www.ietf.org/rfc/rfc3072.txt)</p>	<p>The purpose of this format: arbitrary structured data of different types are assembled together for exchanging between computers of different architectures. With SDXF you can structure your data in any levels of deepness. The particular data elements (called "Chunks") are self-describing. The format is very simple, but nevertheless transparent to the programmer, he does not need and should not care about the bits and bytes of the structure. To access and create the structure the programmer uses a set of functions. The ability to arbitrarily structure your data and serialize it into a self-describing format is reminiscent of XML, but SDXF is not a text format (as XML) --you cannot manipulate an SDXF structure with a text editor. It is used for Platform Independent Network Programming Interface. SDXF is usable as a network interchange data format. This data format is not limited to any application, the demand for this format is that it is usable as a text format for word-processing, as a picture format, a sound format, for remote procedure calls with complex parameters, suitable for document formats, for interchanging business data, etc. SDXF is self-describing, every program can unpack every SDXF-data without knowing the meaning of the individual data elements. Together with the description of the data format a set of functions will be introduced. With the help of these functions one can create and access the data elements of SDXF. The idea is that a programmer should only use these functions instead of maintaining the structure by himself on the level of bits and bytes. (In the speech of object-oriented programming these functions are methods of an object which works as a handle for a given SDXF data block.)</p>
PyTables	<p>Python Tables</p>	<p>PyTables is a package for managing hierarchical datasets and designed to efficiently and easily cope with extremely large amounts of data. PyTables is built on top of the HDF5 library, using the Python language and the NumPy package (it also supports numarray and Numeric</p> <p>PyTables has several characteristics that, when taken together, make it unique over other tools:</p> <ul style="list-style-type: none"> * Easy of use -You can save data containers from NumPy, numarray or Numeric packages in an straightforward manner. Some regular Python containers (mainly lists and tuples) are also supported in a transparent way. * Supports a hierarchical data model - Allows the user to endow a clear structure to all his data. * Natural naming support - PyTables builds up an object tree in memory that replicates the underlying file data structure. Access to the datasets is achieved by walking through and manipulating the attributes of the objects in the tree. See the NaturalNaming section for more info. * Support for table entities - It allows you to create tables by merely define a simple class to describe the record fields. Also, you can tailor your data adding or deleting records in your tables. * User defined metadata - Besides supporting system metadata (number of rows of a table, shape, flavor, ...) the user may specify its own metadata (as for example, room temperature, or protocol for IP traffic that was collected) that complement the meaning of his actual data. * Unlimited datasets size - Allows working with tables and/or arrays with a very large number of rows (up to 2**63), i.e. that don't fit in memory. * On-line data compression - It supports data compression (through the use of the zlib, LZO and bzip2 libraries) out of the box. This become important when you have repetitive data patterns. * High performance I/O - On modern systems, and for large amounts of data, tables and array objects can be read and written at a speed only

		<p>limited by the performance of the underlying I/O subsystem. Moreover, if your data is compressible, even this limit is surmountable!.</p> <ul style="list-style-type: none"> * Support of files bigger than 2 GB - So that you won't be limited if you want to deal with very large datasets. In fact, PyTables support full 64-bit file addressing even on 32-bit platforms (provided that the underlying filesystem does so too, of course). * Architecture-independent
XDMF	<p>Extensible Data Model and Format</p> <p>http://www.xdmf.org/index.php/Main_Page</p> <p>http://www.arl.hpc.mil/ice/</p>	<p>XDMF is an active, common data hub used to pass values and metadata in a standard fashion between application modules. XDMF views data as consisting of two basic types : Light data and Heavy data. Light data is both metadata and small amounts of values. Heavy data typically consists of large arrays of values. All light data in XDMF is stored using the eXtensible Markup Language (XML). Heavy data is currently confined to HDF5 but will eventually include formats like Oracle and Plot3D. XDMF is both a data model and format. That means that the size and shape of the data is described as is its' intended use (i.e. XYZ Position vs. 3D Vectors). Data format refers to the raw data to be manipulated. Information like number type (float, integer, etc.), precision, location, rank, and dimensions completely describe the any dataset regardless of its size. The description of the data is also separate from the values themselves. We refer to the description of the data as Light data and the values themselves as Heavy data. Light data is small and can be passed between modules easily.</p>
XSIL	<p>Extensible Scientific Interchange Language</p> <p>http://www.cacr.caltech.edu/SDA/xsil/</p>	<p>XSIL is a flexible, hierarchical, extensible, transport language for scientific data objects. The entire object may be represented in the file, or there may be metadata in the XSIL file, with a powerful, fault-tolerant linking mechanism to external data. The language is based on XML, and is designed not only for parsing and processing by machines, but also for presentation to humans through web browsers and web-database technology.</p>
XDF	<p>Extensible Data Format</p> <p>http://nssdc.gsfc.nasa.gov/nssdc_news/june01/xdft.html</p>	<p>XDF is a common scientific data format based on XML and general mathematical principles that can be used throughout the scientific disciplines. It includes these key features: hierarchical data structures, any dimensional arrays merged with coordinate information, high dimensional tables merged with field information, variable resolution, easy wrapping of existing data, user specified coordinate systems, searchable ASCII meta-data, and extensibility to new features/data formats. One of the most interesting usages of XDF is the XML-ization of FITS (Flexible Image Transport System) that is used primarily in astronomy. It includes these key features: hierarchical data structures, any dimensional arrays merged with coordinate information, high dimensional tables merged with field information, variable resolution, easy wrapping of existing data, user specified coordinate systems, searchable ASCII meta-data, and extensibility to new features/data formats.</p>
DFDL	<p>Data Format Description Language</p> <p>http://forge.gridforum.org/projects/dfdl-wg/</p>	<p>DFDL (it is XML based) is used for describing the structure of binary and character encoded (ASCII/Unicode) files and data streams so that their format, structure, and metadata can be exposed. This effort specifically does not aim to create a generic data representation language. Rather, DFDL endeavors to describe existing formats in an actionable manner that makes the data in its current format accessible through generic mechanisms. The DFDL description would sit in a (logically) separate file from the data itself. The description would provide a hierarchical description that would structure and semantically label the underlying bits. It would capture: how bits are to be interpreted as parts of low-level data types (ints, floats, strings) how low-level types are assembled into scientifically relevant forms such as arrays how meaning is assigned to these forms through association with variable names and metadata such as units how arrays and the overall structure of the binary file are parameterized based on array dimensions, flags specifying optional file components, etc. Further, if the data file contains highly repetitive structures, such as large arrays or tables, such a description can be very concise. DFDL is a Global Grid Forum standard activity that is building on BinX and other work to provide a general and extensible platform for describing data formats. The idea is to add a DFDL formatted file that describes a given legacy file. An application thus can read and interpret this one without needing a specific reader, only the DFDL reader. BinX is used to describe the content, structure and physical layout (endianess, blocksize) of binary files. It will be a reference implementation for DFDL.</p>
BFD	<p>Binary Format Description</p> <p>http://collaboratory.emsl.pnl.gov/sam/bfd/</p>	<p>The Binary Format Description (BFD) language is an XML dialect based on the eXtensible Scientific Interchange Language(XSIL) that supports the executable documentation of 'arbitrary' binary and ascii data sets. Applying a BFD template to a set of files produces an XML output containing the original data in an XML-tagged format that can be interpreted by other programs or subjected to further processing (i.e. using XSLT).</p>
ESML	<p>Earth Science Markup Language</p>	<p>An interchange technology is an enabling technology that utilizes external metadata to allow applications to plug and play seamlessly with datasets in heterogeneous formats. An interchange technology can be utilized to solve the data/application interoperability problem. The Earth Science Markup Language (ESML) is one such interchange technology. Based on XML it consists of the ESML Schema, ESML Files and the ESML Library. ESML Files contain descriptions of the content, structure, and semantics of a particular set of data files. The ESML Schema defines rules for creating the ESML file. Because ESML Files are external files (i.e. not contained within the data files), both data producers and consumers can create and use these descriptions at any time. A key point is that the ESML Files do not modify the application or the data file itself. The ESML Library is utilized by applications to parse the ESML file and to decode the data format. Application developers can now build data format independent applications utilizing the ESML Library. Furthermore, the applications will not require modification in order to access new formats as they become available.</p>

More information on different data formats can be found under <http://stommel.tamu.edu/~baum/graphics-formats.html>