No.	category	sub-category	type of work	Item	Explanation	Documentation
1	app	archive	research	Standards for Long Term Archiving Complex Scientific Data	Scientific instruments, experiments, and numerical simulations generate extremely large quantities of data, some of which is collected in large repositories. Much of this data could be valuable for many decades to come, e.g., to investigate climate change or other long term phenomena. Today, accessing and using the data requires specific computer equipment and software. Given the rate of technological change, data cannot be maintained in their original format forever. In the case of scientific data, it is especially important to be able to distinguish changes that represent real physical phenomena from artifacts of technological evolution. How can we make it more likely that data will be usable after the systems that created it are gone? We propose to lead an effort to define an appropriate standard for archiving complex scientific data. Our research plan includes a comprehensive background study: • A survey of current approaches • A statement of key requirements and important open problems • A roadmap of proposed work	archive-study.doc+I34
2	app	audio & video	R&D	audio &/or video	Audio and video (AV) data can be a very valuable addition to other types of data stored HDF5. This work would add support for storing and retrieving AV data in/from HDF5 files. The overall design and implementation would satisfy the following requirements 1. Support storage of open-source AV data formats 2. Provide random access to the portions of AV stream based on index (time) 3. Provide a reference implementation for particular AV formats such as Ogg Theora and/or Ogg Vorbis in HDF5	Audio-video exploratory project.doc
3	app	biomedical	R&D	gene model data management	This is Phase II of an SBIR project by Geospiza and THG, which did not receive funding. In Phase I, we developed a prototype called BioHDF that demonstrated the power of HDF5 for managing high volume, highly complex DNA sequencing data. In the Phase II research project, Geospiza and THG proposed to develop a portable, scalable, and adaptable file technology for genotyping software. From the proposal, this project includes following aims: *1. Extend the BioHDF data model to support polymorphism discovery and genotyping, and integrate diverse types of data from multiple technologies.* *2. Build a complete application, accompanied by software tools and APIs, to support BioHDF use in the research community.* *3. Build a prototype application for whole genome association studies based on linkage disequilibrium.* *4. Research methods for incorporating BioHDF into enterprise applications for clinical research and diagnostics.* *The first products, based on BioHDF, will provide data models, APIs, software tools (I/O, algorithms), and a viewer based on HDFView, to support DNA polymorphism discovery and genotyping. Using BioHDF, researchers will be able perform resequencing-based SNP discovery, analyze genotyping data, a* *As a programming environment, BioHDF will be easily extended to accept data from new genotyping platforms and format data for interchange with many c* *Additionally, BioHDF will be able to be used to support studying and performing linkage disequilibrium (LD) calculations in very large data sets like HapMap* *During Phase II and in Geospiza's follow on Phase III efforts, Geospiza will use BioHDF to deliver scalable software applications to support clinical research* *In short, Geospiza and THG expect to provide three deliverables to the research community at the conclusion of this project:* *1. Complete prototype applications for polymorphism discovery and genotyping.* *2. Software tools and frameworks for working with large data sets like HapMap.* *3. Portable, scalable data storage technologies to support the next generation of instruments for genetic analysis.*	<Geospiza phase 2 proposal>
4	app	biomedical	R&D	medical image data management	Imaging plays an enormous role in biomedical applications, and advances in instrumentation are pushing the limits of the data management systems designed to handle the size and number of images. Furthermore, biomedical imaging spans many interlocking domains, and data integration across domains is very important. Recently, members of several biomedical communities have begun to look to HDF5 as a possible technology to help address these challenges. An R&D effort to investigate remedies to data integration and interoperability of software would help us understand the potential of HDF5 to address these problems, and could ultimately lead to a solution the would provide enormous benefits to many medical imaging communities.	Dougherty, M.T., et al. "Review of current and potential microscopy data formats, submitted to J. of Structural Biology.
5	app	biomedical	research	patient folder information	This research would examine the benefits of using HDF5 for managing patient data, as well as marketing opportunities. An example of this is the use of HDF Vsets by Massachusetts General Hospital to store patient folder information, including sonograms, X-ray images, and patient data. It may involve mapping the ACR-NEMA schema to HDF5, and a comparison between DICOM and HDF5. The application needs to store a variety of data, and data types may evolve over time. It needs to support metadata queries, interchange among many different formats, and exchange with non-local providers. Quality, data preservation, ease of use, and tool support are all important. Ultimately could include a High-level Medical Records (MR) API (This would not be the Medical Records application but an HL MR API that might serve as a foundation.)	
6	app	database	research	Integrate HDF5 with SQL server's CLR	"Common Language Runtime" (CLR) is based on .NET runtime, and any language can be wrapped to execute "assemblies" (libraries). In this context, one can define functions, tables, and aggregates that make sense to scientists, in essence keeping all the data an programs in one integrated place. Applying this to HDF5, HDF5 objects would become part of the package, with HDF5 library functions being native to the CLR. A full integration would mean that the HDF5 would be a native C# application – that is, the HDF5 library would be implemented in C#. Because a C# implementation would be a very large undertaking, it is recommended that a prototype first be created that would wrap the HDF5 library with C#. HDF5 data itself would reside primarily in HDF5 files, making them available for other uses just as they now are. However, when performance and other considerations dictated, some HDF5 data would also likely be brought into SQL server tables, and indexes would be created within SQL server for fast query and access to other HDF5 data. Applied Physics Lab (APL) has used this approach for astronomy data management and analysis, rewriting a large code base in C#.	<Microsoft trip report, Sept. 2006; also "HDF5 Integration in SQL Server 2005(+)," Gerd Heber, Cornell Theory Center, October, 2006 (http://hdf.ncsa.uiuc.edu/RFC/SQL-HDF5/)>
7	app	database	research	Integration of HDF5 with databases.	A number of applications have combined HDF5 with databases, in various ways. Some useful things: Tools for moving HDF5 to and from databases. Methods to map HDF5 data to databases. Interfaces that allow databases to access HDF5 data directly. Integration of HDF5 with specific databases, such as MySQL, SQL server, databases that support GIS. (See also "Integrate HDF5 with SQL server's CLR".)	
8	app	earth science	other	Develop and Adapt HDF5 Technologies and Provide HDF5 Services For NPOESS Data Production and Exploitation	NPOESS faces formidable data management challenges. Choosing HDF5 as its distribution format helps address those challenges, for HDF5 is a proven technology for managing large, complex scientific data. However, the capabilities of HDF5 can only be fully exploited for NPOESS by applying the best expertise in their use, and that expertise is available in The HDF Group (THG). THG experience in working with NPOESS staff, supplemented by a survey of key NPP and NPOESS participant groups, has identified three key areas in which The HDF Group can address important NPP and NPOESS requirements: Enhancing the HDF5 library to handle large data volume flow, at the same time to serving very diverse communities. This includes modifying and optimizing the library to achieve high I/O performance with NPOESS data structures, adding new features to meet the needs of the broad NPOESS user communities, and supporting HDF on current and future NPOESS user platforms. Developing new tools and technologies, and adapting existing tools, to enable NPOESS data producers and users to view data content, to translate data into other formats, and to manage metadata. Providing high priority support for NPOESS users by creating a support structure that gives priority to NPOESS HDF users at all levels. This work addresses two of the areas of need identified in the NPOESS IGS Program: 1. Data exploitation and new product development 2. Development of innovative technologies that may help to enable the NPOESS mission	NPOESS_BAA_THG_Proposal_final.doc
9	app	earth science	research	Earth Science Modeling Framework and HDF5 (netCDF4)	The best way to use HDF5 within ESMF is through NetCDF4. Some people working with ESMF have been asking for a netCDF4 I/O module for some time. Probably the best role we can play would be to consult with others using ESMF about any ESMF issues related to HDF5 performance or features underneath netCDF4.	
#	app	general capability	R&D	Remote access support expanded (SRB, etc.)	There are many options here, including web service support, OPeNDAP, Storage Resource Broker.> One idea is to use XML as a standard language to transfer information between client and server. (1) Use XML schema to describe HDF5 files (see "XML HDF5 schema" entry). (2) Implement SOAP for HDF. (3) Develop web service for HDF. (5) Test client-server model. Tech-X has done a web services implementation -- can we use that.	<Portable and Distributed HDF (PDH) Peter X. Cao, March 2003>
#	app	general capability	R&D	Units library	A library to allow applications to associate a measurement unit with a dataset, and convert data values of one unit into another compatible unit. This turns out to be a fairly complicated problem, for reasons that are addressed in the documentation.	<"A proposal of HDF Unit Library" and other docs>
#	app	general capability	R&D	Support for mime types.	A file with a particular MIME type could be encapsulated in HDF5 as one or more datasets, perhaps with a grouping structure. A standard set of attributes could be used to describe the MIME type, similar to the way that images are stored in HDF5, with a corresponding API. It is unclear what services might be desired for accessing MIME types. Simple MIME storage could involve nothing more than the use of a standard attribute, with the file itself stored as 1-D dataset. Higher level functions/tools could add other capabilities.	MIME support in HDF5.doc
#	app	general capability	R&D	XML schema as attribute	put XML schema as attribute on string dataset. As an HL interface supported by HDFView.	
#	app	general capability	research	HDF5 object repository	A repository of information about organizational structures in HDF. It would identify the structures, in-memory attributes, and methods for creating, modifying and working with the attributes.	<"An HDF Extension Object Repository" Barkstrom>

#	app	general capability	research	AMR mesh support	Many applications already support Adaptive Mesh Refinement in HDF5, but it would be good if there were some "blessed" methods, perhaps implemented as HDF5 profiles. There are clearly many ways to support AMR, and no best way. Rather the appropriateness of any give storage method depends on the type of mesh, the type of refinement, and sometimes the I/O and other needs of an application.	
#	app	general capability	research	Common API for HDF4 and HDF5	It is clear that HDF4 files will be around for a long time. At the same time, many applications are migrating from HDF4 to HDF5. These applications, which often need to support HDF4 and HDF5 for essentially the same operations, may benefit substantially from a common API to both HDF4 and HDF5. Because HDF4 and HDF5 are not exactly the same, the API would have to be able to deal meaningfully with those objects are not in the intersection of the two. A "Java object package" exists that does this for HDFView. What is needed is a library for C and Fortran.	Merging_H4_and_H5-ideas from 1997.doc; Also web page about Java Object Package: http://www.hdfgroup.org/hdf-java-html/hdf-object/index.html
#	app	general capability	research	"Designer's Manual"	A manual for data providers, tool builders, and data scholars, to help them design HDF5 files.	An HDF Design Manual-Barkstrom.doc
#	app	geospatial data	R&D	Support for common data model (CDM), "HDFgeo"	(Paraphrased from the web site:) *Unidata's Common Data Model (CDM) is an abstract data model for scientific datasets, the aim of this project is to unify scientific data access. It merges the OPeNDAP, netCDF, and HDF5 data models to create a common API for many types of data. As currently implemented by the NetCDF Java library, it can read (besides OPeNDAP, netCDF, and HDF5) GRIB 1 and 2, BUFR, NEXRAD, and GINI, among others. A pluggable framework allows other developers to add readers for their own specialized formats. The CDM also provides standard APIs for georeferencing coordinate systems, and specialized queries for scientific data types like Grid, Point, and Radial datasets.* The CDM also adds "Georeferencing Coordinate Systems" and specialized "Scientific Data Type" layers, which provide the semantics needed to convert datasets to other protocols and formats such as those required by GIS systems. Plans are in place to support CDM in netCDF, including netCDF 4, but it would be very good to support CDM for HDF5 files that are not netCDF 4. Other activities that may fall within this project: Harmonize CDM unstructured grids with CTT and other mesh work. Provide tool that gives good housekeeping seal for files conforming to CDM.	http://www.unidata.ucar.edu/software/netcdf java/CDM/index.html
#	app	geospatial data	research	geospatial applications	This research would investigate new ways to use for HDF5 for geospatial applications. HDF5 is believed to offer advantages over many existing formats and databases, especially when the data is large or complex. Uses include using HDF5 as scalable object stores for existing systems, as well as a standalone format. Demographics include state and federal government agencies, possibly GIS companies.	
#	app	imaging	R&D	Ultra high res image support	Very high resolution images are resulting in new requirements for managing and analyzing image data. Fields including bioinformatics (e.g. electron microscopy; linkage disequilibrium), art, and remote sensing are turning to HDF5 because it has the basic ability to handle large images. This work would expand HDF5 image support to dealing more efficiently with very high res raster images, such as images with 100K or more scan lines. Capabilities could be added such as multi-resolution support, progressive image transmission, new compression methods, and alternate linearization patterns like space-filling curves. HDFView would be enhanced to better support image panning and zooming, and could add filters to improve visualization of high-resolution data on limited-resolution devices. This may also be a good place to add new image features such as region annotations. (See also "Medical image data management.")	
#	app	imaging	R&D	Expand palette API	The HDF5 Image and Palette API supports simple 8-bit color lookup tables. There has been a good deal of interest in adding more sophisticated palette support. For example, HDF5 could support the notion of a range index table, as described in the "HDF5 Image and Palette Specification." A range index table defines an ascending ordered list of ranges that map dataset values to the palette. Other types of palettes may also be useful.	"HDF5 Image and Palette Specification." http://hdf.ncsa.uiuc.edu/HDF5/doc/ADGuide/ImageSpec.html
#	app	instrument data	research	Testing and measurement apps	This research would investigate a strategy for widespread use of HDF5 for data from instruments that measure things. It would identify potential markets, and possibly data models, libraries and tools for supporting the applications in those markets. Examples include medical testing, flight testing, and automotive testing. These applications need high speed/real time I/O; space efficient storage; data querying; data analysis and visualization. Currently there are many specialized formats, many of which do not scale well. The applications are often critical, so the products must be of high quality. Ease of use is another important factor.	
#	app	product model data	R&D	EXPRESS HL API	General support an HDF5 API and tools for the product model language EXPRESS. EXPRESS is a data modeling language used for product model data. It has been very successful, and is accompanied by a format called STEP (standard for exchange of product data, a text-based format), in which express entities are instantiated. A number of organizations have used HDF5 as a binary companion to STEP. The most promising, from a general usage perspective, seems to be that of EuroSTEP and the European Union, with some participation by The HDF Group (with support from the National Archives and Records Administration). See the web site for information.	See http://www.exff.org/, and click on "Projects/EXPRESS-binary".
#	fmt	doc	dev	Update data model	The HDF5 abstract data model needs updating. This project has been on hold due to lack of resources.	
#	fmt	feature	R&D	Enumerated type extension	Allow enumerated datatype to use non-integer "base" type. E.g. be able to store enumerated type for math constants, like 'pi' or 'e', etc. which need a double as base datatype.	
#	fmt	feature	R&D	Formula raw data storage	Represent a dataset as a formula, so that values are generated, not stored. Can be done in stages. Simple formula, continuous functions for formula (would allow for non-integer indices as input for retrieving the raw data values); step function for formula raw data. Could also implement coordinate system translation. One application asked for this in order to be able to compute new datasets based on existing dataset(s) in a file.	
#	fmt	feature	R&D	Support for union (discriminated, variant, etc.) datatypes.	A union, or discriminated, datatype is one in which several different types can be stored in the same location. This is similar to the union datatype in C. A variant of this is the "tagged union" type in which a tag field indicates just what type is being represented in any given instance, so that the data element is handled correctly when the element is accessed. Unions can save storage by overlapping storage areas for each type, since only one is in use at a time. There is a great deal of interest in this feature, but as yet we have not obtained the resources for it.	
#	fmt	feature	R&D	contiguous storage compression	Support compression of contiguously stored datasets. Currently only chunked datasets can be compressed. One user would like to be able to read compressed data (especially compressed text) out of an HDF5 dataset with a non-HDF5 program. Could this be accomplished in some other way, such as giving them a pointer to compressed chunks, or through a high level API?.	
#	fmt	feature	research	Ability to insert & delete dataset elements.	Add the ability to insert an element into a dataset, between two other elements. In computer science, a *list* structure has this capability, we may be talking about implementing a dataset with list capabilities.	
#	fmt	feature	research	annotations on cells	Enable association of attributes and/or other types of annotations to refer to cells or regions of cells in a dataset. You would want to allow complex annotations, such audio or video, or references to other objects. May be able to do w/o fmt change. of annotations with regions.	
#	fmt	feature	research	Improved support for variable length datatypes.	Variable length datatypes are very common in applications of HDF5, and their frequent use has turn up some shortcomings. Here is an explanation. Variable length datatypes are currently stored in a "heap" in HDF5. The actual data is pointed to by a heap reference. This means that access to a variable length data elements requires two access. It also means that extra storage is used by the heap reference. Furthermore, currently data heaps cannot be compressed, so datasets with variable length data do not compress well. A number of improvements to this are possible. The heap reference size could made smaller. Heap compression could be implemented. Alternate data structures could be put in place to improve access to variable length types. These improvements would be very valuable to a number of types of applications. Other notes on this: - Use fractal heap code introduced in HDF5 1.8 to store VL raw data (VL and region reference datatypes). - Allow users to control how large the fractal heap IDs are, so that "small enough" raw data can be encoded directly in heap ID ("tiny" object storage, in	
#	fmt	feature	research	Variable chunk size and shape	Add support for chunks of variable size and shapem possibly with overlapping elements. This would make it possible to match the storage of chunked datasets with needs of applications that work on portions of a dataset that are different in size and shape. The overlapping of elements would support "ghost zones". This may improve performance in parallel environments.	Folder: Variable sized chunks
#	fmt	feature	research	mark excluded data	Mark data points to be excluded from a dataset. Might make I/O go faster. This may relate to sparse array storage.	

#	fmt	feature	research	Support for pointer datatypes.	Some users would like support for datatypes similar to C pointers. There are many potential users. One user is trying to represent binary trees & linked lists in HDF5 file, so could use pointers in HDF5 concept. Useful in DNA sequence support, too. The PDB format supports pointers effectively.	
#	fmt	performance	research	append only support	"Append-only HDF5" refers to a way to write HDF5-readable data on a medium that does not allow random access reads or writes (e.g., tape). This data would appear to the user to be the contents of a normal HDF5 dataset, and could be included in a file with other data that was written when random-access I/O was possible. This application of HDF5 could be useful in environments in which the speed at which data can be written to disk is important (e.g., flight testing). Such data could be captured quickly, then accessed without the need to be recopied into another format. It could later be stored in the same HDF5 file with processed versions of the original data. If the implementation of append-only HDF5 is simple enough, it might lend itself well to embedded applications. Append-only HDF5 may require an extension to the HDF5 file format, and may consist of a separate library to create append-only HDF5 files and datasets.	Append-Only RFC.doc
#	fmt	performance	research	space-filling curve	Alternate storage format based on space filling curves, such as Hilbert curves, allowing faster access to spatially local data. (See also "Ultra high-res image support.")	
#	fmt	performance	research	sparse array support	We have heard from many applications that deal with sparse arrays, and would benefit greatly if HDF5 could provide one or more alternate formats for sparse arrays. One approach we have looked at is "Judy", but we would want to consider others.	http://judy.sourceforge.net/; Why sparse arrays in HDF5 -- one user.doc
#	lib	API	dev	Explicit open/close returning ID	Have H5open/H5close be explicit and return ID so that library can reference count # of clients using library. (This makes it easier for client applications and libraries to start up/shut down the HDF5 as appropriate & needed).	
#	lib	API	dev	Force-close routine or property spanning several objects	Add routine/property to "force close" all objects opened from links within a group (like "strong" file close degree, only on group ID). Extend this to "force close" attributes on an object when object is closed.	
#	lib	API	dev	Offset/length selection	Add routine to select offset/length within a selection. The functionality is already in library. It just needs an API.	
#	lib	API	dev	Property list improvements	Add serialize & de-serialize functions for generic property lists (and/or individual properties). This will probably need a serialize & deserialize callback routine for users.	
#	lib	API	dev	"Ismounted" routine	Add routine to determine when a file is mounted on a group	
#	lib	API	dev	change dataset type	Implement API and/or tool to convert all data in a file's datasets to another endianness, or from integer to float, etc.	
#	lib	API	R&D	H5move routine	Add routine to move (as in mv) part of a file to another file, keeping an external link in the original file to the new location of the information. (Also see "H5move tool.")	
#	lib	API	R&D	Routine to reserve space before writing	Add routine to reserve space in an HDF5 file where data is to be written. The could improve write speeds when writing large amounts of data. Need to be able to tell HDF5 how much space is needed. Would need to be willing to take performance hit up front, but perhaps that could be done a-priori.	
#	lib	API	research	UML to HDF5 hl API, tools	Implement API and/or tools to convert from UML to HDF5.	
#	lib	doc	dev	Better sample code, integrated with User's Guide	Create a comprehensive collection of code samples that is cross-linked with the documentation. Create a wide-ranging set of sample codes, well integrated with the docs (particularly User's Guide). • Sections of code, illustrating separate issues, would be prepared to fit together into a single program or set of programs. • Many of the required sample codes already exist in the Help Desk material, perhaps most of them. • Integrating these well with the HDF5 documentation would be a first step; additional sample codes could then be created as needed.	
#	lib	doc	dev	Illustrative sample calls per function in Ref Manual	Add illustrative sample calls to each Reference Manual function entry (when appropriate). • Sometimes one; sometimes several, illustrating different uses or edge cases. • Needs to be annotated.	
#	lib	doc	dev	Revise HDF5 Developers' Guide	Revise the "HDF5 Application Developers' Guide."	
#	lib	doc	dev	Rework Tech Notes	Rework the "HDF5 Technical "Notes."	
#	lib	doc	dev	Auto-update copyright	The current way that copyrights are replaced in the HDF source code has many known shortcomings and problems. This small project would improve our tool for inserting and/or replacing copyrights in the source code.	
#	lib	doc	dev	Finish HDF5 User's Guide	Important new chapters needed include: parallel HDF5; other languages (Fortran & C++); performance; filters; HL Interfaces. Also Provide more and better code examples in library, especially hyperslabs and VL. Document limitations of the library. Document backward/forward compatibility issues. Add a chapter on how to make HDF5 go fast•	
#	lib	doc	dev	HL User's Guide	Write user's guide for High Level interfaces.	
#	lib	doc	R&D	Auto-generation of Reference Manual	Implement system to automatically generate Reference Manual	
#	lib	feature	R&D	Concurrent access	Implement concurrent reading/writing between multiple processors	
#	lib	feature	R&D	architecture to transform data during transfer	HDF5 release 1.8 has a new feature that allows arithmetic operations (add/subtract/multiply/divide) to be performed on individual data elements as they are being written to/read from a file. A need has been expressed for transforms to perform operations on data aggregates, such as data reduction operations, node reordering on unstructured grids, or structured-to-unstructured grid conversions. An architecture is described in the accompanying talk that describes how this comprehensive approach might work.	transform_ideas-Jan_2002.ppt
#	lib	feature	R&D	Encryption filter.	A prototype API exists to support encryption in HDF5. This prototype could be made into a fully-supported filter, similar to the compression filters. For added security, there need to be other features that insure that encrypted data is not observable once it is loaded into memory. There is an RFC on this project.	Folder: encryption
#	lib	feature	R&D	Contiguous storage extendibility	Make it possible to extend contiguously stored datasets. A user wants to append to contiguous dataset (possibly compressed). May need to keep data contiguous (i.e. would need to re-copy the data in the file). This might be accomplished through a high level API.	
#	lib	feature	R&D	New compression support	Store popular types of compressed data, such as JPEG, JPEG 2000 or Mr. SID data into HDF5 file.	

#	lib	HL	dev	Wildcard search	Add wildcard search to query operations.	
#	lib	HL	R&D	decomposed arrays	From a loyal HDF5 user: “As far as we can tell, the array abstraction HDF5 supports is just a "global" abstraction. That is, the array in the file is a single coherent entity, not several pieces that are mapped into a single coherent entity. … Of course, HDF5 does support the notion of chunking but that is NOT the kind of "pieces" I am looking for here. I need to store an array in pieces, where each "piece" is probably a different size in each dimension AND where one "piece" might overlap with another and all pieces are some hyper-slab out of the larger, single coherent entity. … Basically, you'd have to add to a dataset the notion of a decomposition and the ability to read/write as a single, coherent, whole or as a decomposition.”	folder: Variable sized chunks
#	lib	HL	R&D	Dimension scale enhancements	The use of formulas for generating dimension scales would save space and would also make it easy for applications to know when a dimension scale matched to a formula. Examples including simple linear formula (e.g. origin + delta), step functions, and continuous functions. The latter would allow for non-integer indices as input for retrieving the raw data values. Coordinate System translation has also been suggested.	
#	lib	Java	dev	support writing variable length types in Java	Values of variable length (VLEN) data are stored in a C structure, "typedef struct { size_t len; void *p; } hvl_t;," which is not supported by the hdf-java model. Currently, hdf-java only reads VLEN data into strings and does not support writing VLEN data. It would be good to implement a Java class similar to the hvl_t C structure so that hdf-java would be able to write VLEN data to a file.	
#	lib	language	R&D	Perl API	Implement Perl API, perhaps based on PDL.	
#	lib	language	R&D	Python API	Implement Python API.	
#	lib	parallel	R&D	parallel H5lite	Implement parallel version of H5lite.	
#	lib	parallel	R&D	HDF5 - MPI datatype conversion	This feature would take an MPI type and map the type described into a corresponding HDF5 datatype description, if possible. And vice versa.	
#	lib	parallel	research	Flexible parallel HDF5.	Under certain circumstance, parallel I/O in hdf5 is very effective, but because of the complex data and metadata structures in HDF5, parallel can also be cumbersome and slow. Particularly difficult is the need for small metadata writes that can accompany the efficient large write data writes. A project called "flexible parallel HDF5," which used a set-aside process for managing data and metadata I/O, did address this problem, but was not very successful in achieving its goals of simpler and more efficient I/O. Other proposals have been put forward, but we have as yet not had the funding to pursue these. See the associated documents for a description of flexible parallel HDF5, and subsequent ideas. These ideas include: Complete the set-aside process work that has not been completed; make it work for chunked storage; add the ability to update dataset attributes; implement collective multiple dataset open; create 1 dataset per node in parallel; change API to allow flexible, parallel HDF5, so it doesn't need collective calls.	Folder: Flexible parallel HDF5
#	lib	parallel	research	segment file per process for better parallel access	Enable segmenting of file on per-process basis. Break up address space on a per-process basis. They want a group per process. Need mandate from ASCI people. Could do with independent open or flexible parallel. Flexible parallel might fix this too. Another description: "Windowing" on a file, where a group "owns" a section of the file and can operate on that section of the file independently of other parallel processes. It is also suggested that log-based I/O might help.	segment per process.doc (also See discussion in 3/11/03 trip report)
#	lib	performance	dev	Performance benchmark extensions	HDF5 performance benchmark suite is available for regular performance testing of HDF5. There are a number of useful ways to expand HDF5 perf: - Add parallel I/O performance tests (a tool h5perf currently tests HDF5 parallel I/O); - document MPE - Implement performance regression testing - Add more criteria for the benchmark (currently only time is used. We could add file size, overhead, for instance) - Add more sequential benchmarks into the framework	
#	lib	performance	dev	Sharability checking	Add a way to check for two indirect sharable objects referring to the same object. Also add a way to check for an indirect object pointing to a particular named object. Only applies to datatypes at this time. Probably different datasets -- you are checking to see is they use the same named datatype. Basic idea: if there are shared things, let's find a way to compare them.	
#	lib	performance	R&D	I/O benchmarking R&D effort	I/O benchmarking is very important to many HDF users, and currently is handled in a somewhat ad hoc manner. Much could be gained from having an organized, coordinated, coherent group of people and procedures to help with I/O benchmarking. We could engage benchmarking experts with experience in this area, gain familiarity with available tools in the field, and learn to apply these tools to problems of customers, and to projects that would improve HDF itself. This group would also solicit research funding and work with researchers in the field, such as the folks at the DoE labs, academia.	
#	lib	performance	R&D	parallel support for VL types	Make VL datatype access work in parallel. This has already implemented in an application called SSLIB, but H5dump and H5ls don't handle it.	
#	lib	performance	R&D	Vectorized I/O	Add "vectorized" read & write routines to VFL interface, to allow VFL to make more efficient I/O requests. (can interface to POSIX read()/write() routines)	
#	lib	performance	research	Group tiny writes for I/O efficiency	Develop strategies for making small writes, such as caching them together. Give the applications the ability to let the library know when this will happen.	
#	lib	performance	research	Fast writes: Streaming B-trees and cache-oblivious techniques	Research on streaming B-trees and “cache-oblivious” techniques claim large improvements in data writes, without major sacrifices in later search capabilities. Could streaming B-trees benefit HDF5 applications? See Bender paper at HECFSIO 2006 Workshop. Also see M. A. Bender, et al, "Cache-Oblivious B-Trees." SIAM Journal on Computing, 35(2):341-358, 2005. Has applications in biomedicine (see "Interactive Exploration of Large Remote Micro-CT Scans").	Techniques for Streaming File Systems and Databases Bender et al. http://institute.lanl.gov/hec-
#	lib	performance	research	Fast writes: log-based I/O	When I/O is write-intensive and/or writes to several different datasets occur at the same time. Two approaches identified: (1) write in to groups to log files, (2) write in background (active buffering). Requires mods to apps and library. Steps: (1) trap I/O calls, (2) log I/O calls, (3) replay in background. Need separate thread, or possibly spawn a separate process.	
#	lib	performance	research	HDF and Object Storage Devices	Object storage devices (OSD) represent a growing interest area among a number of major players, both academic and industrial. This paper "Applicability of Object-Based Storage Devices in Parallel File Systems,' by Pete Wyckoff is an example. How might HDF5 evolve to take advantage of OSD?	Applicability of Object-Based Storage Devices in Parallel File Systems," http://institute.lanl.gov
#	lib	performance	research	Middleware caching optimizations	Argonne and Northwestern U. are doing very interesting work in the area of middleware and file system optimizations. It should be possible to exploit this work in HDF5 to improve I/O on high end systems.	"Scalable I/O Middleware and File System Optimizations for High-Performance
#	lib	platform	research	embedded HDF	Increasingly, HDF5 is being used for instrument data. Sometimes such data is collected on embedded systems, using special operating systems such as VxWorks. Because HDF5 doesn't currently run on such systems, these applications must first collect their data in a raw form on the embedded device, the offload the data to a general purpose computing system and store it in HDF5. This approach bears the extra cost of moving the data to HDF5, and also loses the benefits of HDF5 when dealing with the data while it is still on the embedded system. In this project, HDF5 would be ported to an embedded operating system such as VxWorks.	
#	lib	qa/tst/cfg	dev	Create comprehensive JNI test suite	The current JNI (Java wrapper for HDF libraries) are not been fully tested. This task would be to develop a comprehensive test suite (include testing memory leaks) for the HDF4/5 JNI.	
#	lib	qa/tst/cfg	R&D	Extract run-time features that are not specific to HDF5	This is about getting the "infrastructure" routines out of the HDF5 library (such as the property list, error handling, ID management, etc.) and putting them in a separate library that doesn't have anything to do with HDF5 files. That would leave the "main" HDF5 library to just focus on doing I/O operation on HDF5 files. This would probably result in a *significant* bonus for maintenance on the library.	
#	lib	qa/tst/cfg	R&D	Implement error detection for contiguous datasets	I.e. Decide whether there should be format change of new error-detecting code for object header message. Propose API extensions for error-detecting code, enhance internal library functionality. Document these functions.	
#	lib	qa/tst/cfg	research	QoS/reliability	Crash recovery is just one issue related to Quality of Service (QoS) and reliability. Data corruption is another. Hdfcheck is another. We would like to take a comprehensive look at this issue, rather than address individual needs as they arise.	
#	lib	tools	dev	HDF5 tools library.	The HDF5 tools library has routines that are useful in building command line tools. It needs to be brought up to date and documented.	
#	other		dev	Conferences for HDF5 users	THG is interested in joining others to organize conferences or workshops offering in-depth information about certain aspects of HDF5. Examples of topics: the basic concepts of HDF5, innovative features in HDF5, data modeling with HDF5, applications of HDF5, tuning HDF5 for better performance, innovative data structures in the HDF5 format, interesting techniques in the HDF5 library.	

#	other		other	HDF standardization	Widespread acceptance of HDF depends on the confidence that potential adopters have in HDF. Confidence is built, at least in part, by perceptions that HDF is a widely accepted product, and that it is well-designed, well-implemented, and well-documented. Going through standardization processes is itself valuable to HDF, as it requires HDF go undergo a rigorous examination by a standards body, and thus to meet a certain level of stability, quality, and specificity.	
#	other		research	HDF book(s).	One or more books should be written to explain HDF to a wider audience, and to make HDF5 more accessible. Topics they might cover include the basic concepts of HDF5, innovative features in HDF5, data modeling with HDF5, applications of HDF5, tuning HDF5 for better performance, a specification of the format, innovative data structures in the HDF5 format, interesting techniques in the HDF5 library.	
#	tool	convert	dev	change dataset type	Implement API and/or tool to convert all data in a file's datasets to another endianness, or from integer to float, etc.	
#	tool	convert	dev	Excel import/export tools	Several tools are needed to export HDF5 data to Excel. This capability would also make HDF5 data readily available to other software that understands the Excel format. H5dump can convert HDF5 data to a comma-delimited format that Excel can read, but choosing the options to do this is a bit tricky, so this needs to be refined. A binary version would also be good. particularly for dealing with very large arrays. A somewhat larger project would be to create an HDF5 Excel plug-in that directly imports data from HDF5 to EXCEL. Exporting from Excel to HDF5 would also be useful.	
#	tool	convert	dev	More format converters	Converters for various common formats have been requested, including JPEG2000, TIFF, GeoTIFF, PNG, others. Also of interest are converters to special formats, such as FITS (astronomy). We welcome other suggestions.	
#	tool	convert	R&D	Adobe Photoshop plug-in	A plug-in that converts HDF files for use with Adobe Photoshop.	
#	tool	convert	R&D	DICOM harmonization	Digital Imaging and Communications in Medicine (DICOM) is a standard for managing medical images and related metadata. It includes a file format definition and a rigorous data dictionary, and is widely accepted in the biomedical community. At the same time, the biomedical community is using HDF5 increasingly for managing large images. A very valuable project would be to map DICOM to HDF, and create converters for converting between the two formats. This could be expanded to adapt tools that currently work with DICOM to support HDF5, and to adapt certain HDF5 tools to provide meaningful display and operations on DICOM-compliant medical images in HDF5. This would potentially be a fairly large project with high payoff. A group of interested parties has written a paper on this issue, submitted to the Journal of Structural Biology.	Dougherty, M. T., et al. “Review of current and potential microscopy data formats, submitted to J. of Structural Biology.
#	tool	convert	research	UML to HDF5 hl API, tools	Implement API and/or tools to convert from UML to HDF5.	
#	tool	data quality	R&D	File recovery tools.	An HDF5 file may be corrupted during transmission, while in storage, due to a system crash, and for other reasons. These tools include "H5doctor,", a tool to help restore all or part of a corrupted HDF5 file. Also needed is a utility to scan and repair hdf5 files.	
#	tool	feature	R&D	H5move tool	A tool to move (as in mv) part of a file to another file, keeping an external link in the original file to the new location of the information (See also "H5move routine")	
#	tool	general capability	other	Resurrect NCSA-Spyglass-Fortner tools	A large number of very useful tools have fallen into disuse because they are no longer supported by the organizations that developed them. These include in particular the NCSA suite of tools from the late 80's and early 90's, and commercial tools such as the Spyglass/Fortner tools. This project would acquire rights to these tools from their current owners, make them open source and support them.	
#	tool	general capability	R&D	HDF5 Shell tools.	An HDF5 shell would be a program that lets users type commands to directly query and operate on an HDF5 file. E.g. list, create, remove, dump, mount. A shell could also be implemented as a GUI. Should allow scripting for batch processing. (Could just expand utilities/tools to be a sufficiently complete set of command line utilities/tools that they can become a primitive-but-usable de facto shell.)	
#	tool	general capability	R&D	More tools and command-line utilities.	Tools to create/remove links in file, create attributes on objects, etc. (Please give us your suggestions.)	
#	tool	general capability	R&D	COTS support	Work with COTS tools to help them support HDF effectively. Tools mentioned by some of our major users include Labview, Matlab, pastran/nastran, Python, PV Wave (Visual Numerics), TecPlot, and Excel.	
#	tool	h5diff	R&D	H5diff descriptive stats	Add option to h5diff to provide descriptive statistics. E.g., a histogram of the differences, or max of all the differences. Use case: there are many small differences, and we need to determine if there are any big differences, but don't know absolute thresholds.	
#	tool	h5diff/h5ls	R&D	H5diff/H5ls plug-in	Provide a plug-in for h5ls & h5dump to register custom data dumping routines. This would be very nice for the higher level libraries (HDF-EOS, netCDF) to dump certain custom data structures in appropriate ways.	
#	tool	h5gen	dev	xml h5gen upgrade	h5gen is a tool for generating a regular binary HDF5 file from a corresponding XML HDF5 file. This is a valuable tool that has fallen into disrepair. It needs to be repaired, and also needs to be upgraded to work with the latest features of HDF5. (See also "XML HDF5 schema and tools.")	
#	tool	HDFView	dev	XML display	Adapt HDFView to display XML in a structured way. (Also see "New editing features", "import/export features", and "XML HDF5 schema and tools.")	
#	tool	HDFView	dev	New editing features	(a) Add support, when creating objects in HDF5, to specify more properties, such as checksum, compression, possibly other filters. (b) Allow annotations to be added to the cells in a dataset, similar to how excel does it. This should involve coming up with a standard way to represent such annotations. (c) Have a template of what you are viewing on the screen -- which datasets are open, etc. Be able to save that template and then when you load the next file you will look at the very same things. (d) When copying and pasting to excel, make it possible to include the column headings. (e) Buffer data in HDFView when loading a large dataset. (f) Support multi-threading while loading a large dataset	
#	tool	HDFView	dev	Import/export features	(a) Export/import to an Excel format. (Also see "Excel import/export tools.") (b) Add the ability to output to ESB and ESQ formats, used for flight test data. (See also "converters".)	
#	tool	HDFView	dev	Performance improvements	(a) Add warning when opening a dataset that is very large. (b) Make HDFView perform better when displaying large datasets (c) Make HDFView perform better when displaying large numbers of objects.	
#	tool	HDFView	dev	Viewing and analyzing objects and data	(a) Display attributes more conveniently, as when the mouse hovers over the icon for a dataset or group. (b) Allow one to click on the points on a plot to see what the (x,y) values are. (c) Make it possible to highlight every other row for readability. (d) Make it possible to draw a plot comparing the columns in two different datasets. (e) Allow one to filter data in a table with simple queries such as "(altitude > 700) and (airspeed < 500)". This would produce a table with only the rows that satisfy that query. (f) Make it possible to use h5dump directly from HDFView.	
#	tool	HDFView	dev	More sample files	(a) Provide sample files showing how compound datatypes can use enums and other datatypes effectively, such as an enum for "gear up/gear down". (b) Create other sample files showing realistic applications.	
#	tool	HDFView	R&D	Comprehensive testing for HDFView	A comprehensive test suite for the HDFView GUI is needed. There are GUI testing tools available.	
#	tool	HDFView	R&D	Web browser plug-in	The current Web browser plug-in is not fully tested. We need to test memory leaks and fix bugs. We also need to add more features to the plug-in.	http://www.hdfgroup.org/plugins/
#	tool	lib	research	HDF5 file wizard	Wizard-like tool that walks you through a set of steps in creating an HDF file. Options include creating skeleton file, creating XML (or other DDL) description, producing skeleton source code for reading/writing/querying. This might just be an extension to HDFView. (See also "XML HDF5 schema and tools.")	
#	tool	performance	R&D	Free-space tool	When HDF5 files are modified, it is common to create holes ("free space") in the file in which there is no useful information. This tool would detects free space in files, making it easier to decide whether or not to repack the file. It might also be used to determine when applications are creating free space in a file.	
#	tool	performance	R&D	Chunk tuning utility	It is often difficult to know what an appropriate chunk size might be for a given application. This utility would perhaps run some tests on a given system with a given file and choose a recommended chunk size. It might examine features of the system (e.g. disk block size) to use in its recommendations. Another approach would be to feed the utility information about the data and expected access patterns, and make a recommendation based on that information.	
#	tool	performance	R&D	Profiling interface	Implement a profiling API at the VFD level that provides useful information about how the file is organized. Examples of information include block alignment information.	