Final Report of IVS Working Group 4 (WG4) on Data Structures

John Gipson[1], Johannes Böhm[2], Sergei Bolotin[1], David Gordon[1], Thomas Hobiger[3], Chris Jacobs[4], Sergey Kurdobov[5], Axel Nothnagel[6], Ojars Sovers[7], Oleg Titov[8], Hiroshi Takiguchi[9]

[1]NVI, Inc at NASA Goddard Spaceflight Center, United States,
[2]Department of Geodesy and Geoinformation, Vienna University of Technology, Austria
[3]National Institute of Information and Communications Technology, Japan
[4]Jet Propulsion Laboratory, United States
[5]Institute of Applied Astronomy, St. Petersburg, Russia
[6]Institute of Geodesy and Geoinformation, University of Bonn, Germany
[7]Remote Sensing Analysis, Inc., United States
[8]Geosciences Australia, Australia
[9]Auckland University of Technology, New Zealand

1 History

At the 15 September 2007 IVS Directing Board meeting I proposed establishing a "Working Group on VLBI Data Structures". The thrust of the presentation was that, although the VLBI database system has served us very well these last 30 years, it is time for a new data structure that is more modern, flexible, and extensible. This proposal was unanimously accepted, and the board established IVS Working Group 4. Quoting from the IVS Web site [1]:

"The Working Group will examine the data structure currently used in VLBI data processing and investigate what data structure is likely to be needed in the future. It will design a data structure that meets current and anticipated requirements for individual VLBI sessions including a cataloging, archiving, and distribution system. Further, it will prepare the transition capability through conversion of the current data structure as well as cataloging and archiving softwares to the new system."

2 Organization of the Working Group

Any change to the VLBI data format affects everyone in the VLBI community. Therefore, it is important that the working group have representatives from a broad cross-section of the IVS community. The initial membership was arrived at in consultation with the IVS Directing Board. Table 1 lists the current and past members of WG4 together with the

Table 1. Past and Current Members of IVS Working Group 4	
John Gipson	Chair/Solve
Sergei Bolotin	Steelbreeze
Roger Capallo	Correlators
Axel Nothnagel	Analysis Coordinator

David Gordon	Calc/Solve
Chris Jacobs Ojars Sovers	Modest
Oleg Titov Volker Tesmer	Occam
Johannes Böhm	Vievs
Sergey Kurdobov	IAA
Anne-Marie Gontier	PIVEX
Thomas Hobiger Hiroshi Takiguchi	NICT/C5++

We were all saddened at the premature death of Anne-Marie Gontier during this period. In addition some members left the Working group because of a change in professional status or retirement.

3 Earlier Related Work

There have been proposals to redesign how geodetic VLBI data is stored and archived. We want to particularly mention two of these. We want to particularly mention:

1. A memo written by Leonid Petrov "Specifications of a geo-VLBI format" in the late 1990s. [See reference to undated memo by L. Petrov in Reference.]
2. The work of Anne-Marie Gontier on the Gloria database.

In addition, at the second IVS Analysis Workshop in February 2001, a working group was set up to develop a VLBI exchange format independent of platforms and operating systems. This working group included Gontier and Petrov, as well as other members of the VLBI community. The working group resulted in the definition of the PIVEX format [Gontier, 2002]. Several Mark3 databases, including an Intensive (01DEC31XU), a NEOS (01DEC11XE) and an RDV session (01MAY09XA) were converted to PIVEX format. Unfortunately, PIVEX was never widely adopted in the VLBI community.

Since the goals of this earlier working group were substantially similar to IVS Working Group 4 It is not surprising that WG4 was strongly influenced by this earlier work since the goals of this earlier working group were so similar. We learned from both the successes and failures of this earlier work. In terms of success much of the way the data is organized is similar to the organization proposed by Petrov and/or Gontier. In particular, organizing data by scope (how broadly applicable is the data) and the concept of wrapper are very similar to L. Petrov. Both of these are discussed below. A major difference is that WG4 proposes relying on an established format (NetCDF) to store the VLBI data instead of defining an entirely new format.

4 History and Goals

WG4 held its first meeting at the 2008 IVS General Meeting in St. Petersburg, Russsia. This meeting was open to the general IVS community. Roughly 25 scientists attended: ten WG4 members and fifteen others. This meeting was held after a long day of proceedings. The number

of participants and the lively discussion that ensued is strong evidence of the interest in this subject.

A set of design goals, displayed in Table 1, emerged from this discussion. In some sense the design goals imply a combination and extension of the current VLBI databases, the information contained on the IVS session Web-pages, and much more information [2].

During the next year the working group communicated via email and telecon and discussed how to meet the goals that emerged from the St. Petersburg meeting. A consensus began to emerge about how to achieve most of these goals.

Table 1. Design Goals of Working Group IV

Goal	Description
Provenance	Users should be able to determine the origin of the data and what was done to it.
Compactness	The data structure should minimize redundancy and the storage format should emphasize compactness.
Speed	Data retrieval should be fsat.
Platform/OS/ Language Support	Data should be accessible by programs written in different languages, running on a variety of computers and operating systems.
Extensible	It should be easy to add new data types.
Open	Data should be accessible without the need of proprietary software.
Decoupled	Different types of data should be separate from each other.
Multiple data levels	Data should be available at different levels of abstraction. For example, levels most users are only interested in the delay and rate observables. Specialists may be interested in correlator output.
Completeness	All VLBI data required to process (and understand) a VLBI session from start to finish should be available: schedule files, email, log-files, correlator output, and final 'database'.
Web Accessible	All data should be available via the Web.

The next face-to-face meeting of WG4 was held at the 2009 European VLBI Meeting in Bordeaux, France. This meeting was also open to the IVS community. At this meeting a proposal was put forward to split the data contained in the current Mark3 databases into smaller files which are organized by a special ASCII file called a wrapper. Overall the reaction was positive. In the summer of 2009 we worked on elaborating these ideas, and in July a draft proposal was circulated to Working Group 4 members. The ideas continued to be refined over the next years.

Because of the desire for the new format to be open, and as a nod to Mark3 database structure, we originally called the new format openDB. A subsequent internet search revealed that this name was already taken, and the new format was renamed to vgosDB.

5 Current Organization of VLBI Data

Currently the smallest piece of VLBI data that is routinely analyzed is a VLBI session. This information is archived and stored in Mark3 database. These databases contain information used in the analysis of a VLBI session, which are usually 1-hour (intensives) or 24-hours. With very few exceptions, there are usually gaps between sessions, and hence a VLBI session is a natural piece of VLBI data to work with.

5.1 Mark3 Databases

The Mark3 database organizes data by "Lcodes" with each Lcode corresponding to a different data item. The data associated with a given Lcode can be stored as ASCII Strings, Integer*2, Integer*4 or Real*8. The Mark3 database was designed to contain all[1] the data necessary to analyze a VLBI session within a single file. The database file contains the observables but it also contains theoretical values, partials and calibrations.

There are two types of Lcodes:

1. Type-1 Lcodes contain data that is applicable for the entire session.
    a. Examples: station names and positions, source names and positions, session name, etc.
    b. This information occurs only once in the database.
    c. There are roughly 100 different Lcodes.
2. Type-2 and -3 Lcodes are conceptually identical. Type-3 Lcodes were introduced because of limitations of the HP operating system in the 1980s. These Lcodes contain observation dependent data:
    a. Examples: EOP data, a priori nutation, various partials, delay, rate, sigmas.
    b. The database contains data for each Lcode and each observation, e.g., each observation has an associated EOP value, met values, etc.
    c. There are around 400 different Type-2 and Type-3 Lcodes.

The Mark3 databases are fundamentally organized by observation, as illustrated below.

Table 2. Mark3 databases are organized by session- and observation-dependent data.

Type 1 Lcodes: Session Data						
Source List	Station List	Correlator	Principle Investigator	Flags		Etc...
Type 2 and 3 Lcodes: Observation Data						
	Lcode1 SourceName	Lcode2 1st Station	Lcode3 2nd Station	Lcode3 EOP	…	LcodeM Observable
Obs1						
Obs2						
…						
ObsN						

[1] Over time this proved impractical, and some of data is now stored in external files. Examples include EOP files, pressure loading, episodic motion, etc.

It is important to note that the Mark3 database format is both a method of organizing data and a means of storing data. Data is organized by Lcodes, where the Lcodes are Session-dependent or Observation dependent. The data is stored in a proprietary format.

The Mark3 database format has some nice features which we do not want to lose. Among these are:

1. Table of Contents. You can easily see what data is available in a given database.
2. Self-descriptive data. Each data item has a brief description of what it is.
3. History. Each database contains a history of its processing.

The specific Lcodes within a database vary depending on the age of the database and how the data was processed. Older databases contain obsolete Lcodes which are relics of how the data was analyzed at the time. In addition databases contain information about the fringing process, and this information is different for each kind of correlator. The number of Lcodes has increased over time as a result of model changes, the desire to use new kinds of data, etc. A consequence of this is that a Mark3 database contains data that is obsolete and never used.

Some problems associated with Mark3 databases are:

1. Requires proprietary software.
2. Only used by the *calc/solve* user community.
3. Redundancy.
    a. Much VLBI data is really scan-dependent, not observation dependent.
    b. There is one database for each band.
4. Mixing of observations and theoretical models.
5. Changing a model or adding new kinds of data means updating the entire database.
6. Difficult or impossible to exchange partial information, i.e., ambiguity resolution or editing criteria.
7. Contains obsolete data and models.
8. Contains data that is very seldom used.
9. Contains data that is *calc/solve* specific.
10. Slow data access which makes it prohibitively time-consuming to use the Mark3 database in large VLBI solutions.

In spite of the above problems, the Mark3 database has been in use for over 35 years which is a testament to the many virtues it has.

5.2 NGS Card Format

Because of the proprietary nature of the Mark3 database an alternative format called "NGS card" format was developed to exchange VLBI information. This consists of a single ASCII file with a series of lines. The top of the file contains header information which describes the session as a whole, such as stations, sources and their positions. This is analogous to the Type-1 Lcodes in the Mark3 database. This is followed by information about the observations. This is analogous to the information contained in the Type-2 Lcodes.

The advantage of the NGS format is that it is fairly easy to write software to parse the file. Some of the disadvantages are:

1. Inflexible. Hard to add new data types.
2. Does not contain all of the VLBI data needed to analyze a solution from the beginning. Hence errors in the initial data editing and ambiguity resolution are 'baked-into' the data and become impossible to fix down-stream.
3. Machine access is slower than for binary files.

5.3 PIVEX Format

Pivex was an ASCII format designed to archive and store VLBI data [Gontier 2002]. It was never widely adopted.

5.4 Other Formats

Because of the speed advantages that storing data in binary files has, most VLBI analysis software uses a custom format specific to the particular software.

For doing large global solutions which combine data from many sessions, *solve* stores data in 'superfiles'. These superfiles are essentially binary dumps of Fortran common blocks which contain subsets of the data in a Mark3 database. The organization is also roughly similar to that of Mark3 databases. One common block contains information common to the session as a whole. Another common block contains information applicable to a given observation. Solve uses this information by reading in the common-block for a particular observation *en-masse*.

Other software packages such as Steelbreeze, Occam and VieVs use their own proprietary format. This makes it difficult to exchange data.

As mentioned above, one of the primary reasons for using a proprietary format is that Mark3 database access is slow. Proprietary binary formats were developed in part as a reaction to this. On the other hand the NetCDF format is designed for fast access. One of the goals of the vgosDB is to encourage the use of a common format for data processing and exchange.

6 Overview of vgosDB format

In this section we present a brief overview of the new format.

6.1 Organizing Data by Sessions

The smallest piece of VLBI data routinely analyzed is the data contained in a Mark3 database. Each database contains the data for a single session. Sessions are usually 24 hours (standard sessions) or 1 hour (intensives). With a few exceptions such as the CONT series (campaigns where VLBI data is taken over an extended period of time, usually around 2 weeks), there are usually gaps between sessions, and hence a VLBI session is natural piece of VLBI data to work with.

In the most optimistic VLBI2010 scenario, there are never gaps in the observing. Some stations may stop observing for a time (for example, for scheduled maintenance), but there will always be a number of stations observing. This is analogous to the situation in GPS and SLR, where some instruments are always on. However, both of these techniques find it useful to divide data into smaller pieces for analysis.

A crucial difference between VLBI and the other space-geodetic techniques is that VLBI is a co-operative venture—stations must observe in a coordinated manner, i.e., two or more stations must observe the same source at the same time, and the observing modes must be the same or similar. If the observing mode is substantially different, then you cannot correlate the data between two stations, and hence there are no observables.

We propose to continue to organize data by session. However, instead of storing most of the data related to a particular session in a Mark3 database, the vgosDB format breaks the data into smaller pieces which are stored in files (see below). All of the data associated with a particular session is stored under a directory named after that session, e.g., 10JAN04XA would contain all of the information related to IVS session R1412.

6.2 Modularization

A solution to many of the design goals of Table 3 is to modularize the data, that is to break up the data associated with a session into smaller pieces. These smaller pieces are organized by ‘type’; e.g., group delay observable, pressure data, temperature data, editing criteria, station names, and station positions. In many, though not all, cases, each ‘type’ corresponds to a Mark3 database Lcode. We refer to each data item as **an vgosDB variable**.

Different data types are stored in different files, with generally only one or a few closely related data types in each file. For example, it is logical to store all of the met-data for a station together in a single file. This data usually comes from a single instrument. However, there is no compelling reason to store the met-data together with pointing information. Splitting the data in this way has numerous advantages, some of which are outlined below. The first three directly address the design goals. The last was not originally specified, but is a consequence of this design decision.

1. Separable. Users can retrieve only that part of the data in which they are interested.

2. Extensible. As new data types become used, for example, source maps, they can be easily added without having to rewrite the whole scheme. All you need to do is specify a new data type and the file format.
3. Decoupled. Different kinds of data are separated from each other. Observables are separated from models. Data that won't change is separated from data that might change.
4. Partial Data Update. Instead of updating the entire database, as is currently done, you only need to update that part of the data that has changed.

6.2.1 Reducing Redundancy.

Data will also be organized by 'scope', that is how broadly applicable it is: A) The entire session (for example, source position); B) A particular scan (EOP); C) A particular scan and station (met-data); D) or for a particular observation (delay observable and sigma). The current Mark3 database is observation oriented: all data required to process a given observation is stored once for each observation. This results in tremendous redundancy for some data. For example, in an N-station scan, there are N*(N-1)/2 observations, and each station participates in N −1 observations. Scan dependent data, such as EOP or source information, is the same for all observations in a scan. However in Mark3 databases, this information is stored once for each observation, resulting in an N*(N-1)/2 times redundancy. Station dependent data which is the same for all observations in a scan, such as pointing or met-data, is stored N-1 once for each observation the station participates in the scan, resulting in an (N − 1)-fold redundancy. Organizing data by scope allows you to reduce redundancy.

6.3 NetCDF as Default Storage Format

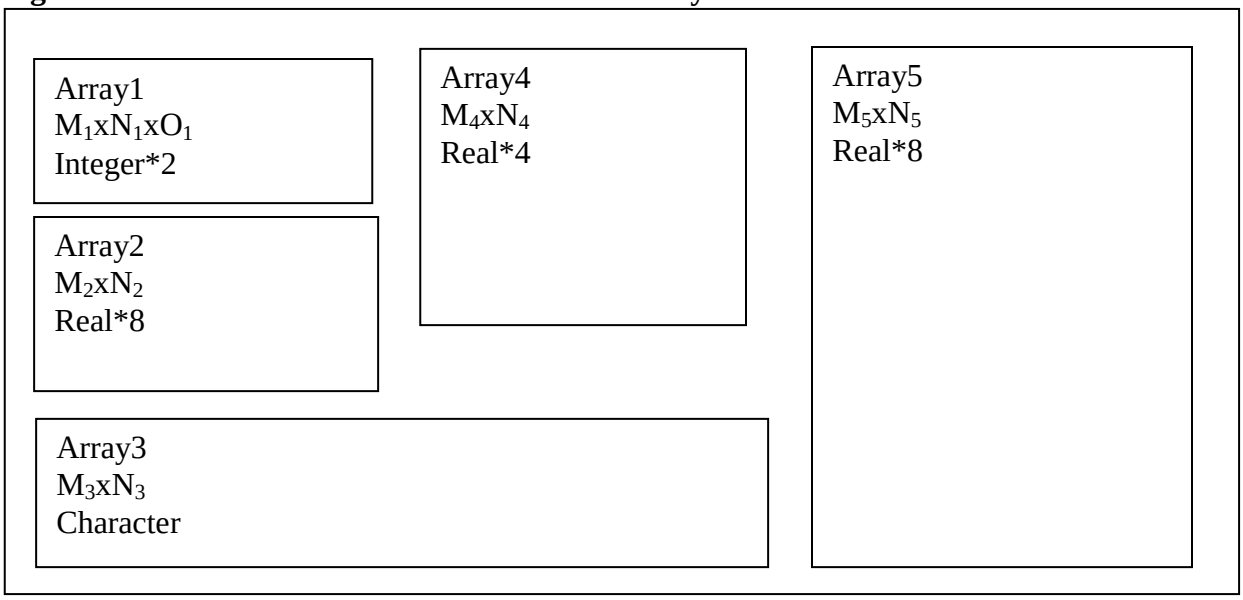
Working Group 4 reviewed a variety of data storage formats including NetCDF, HCDF, CDF, and FITS. In some sense, all of these formats are equivalent—there exist utilities to convert from one format to another. Ultimately we decided to use NetCDF, because it has a large user community since numerical weather models are stored in NetCDF files. Several members of the Working Group have experience using NetCDF. In addition, Thomas Hobiger [Hobiger 2008] wrote a program to store Mark3 databases in NetCDF format and developed analysis software that uses the NetCDF format [Hobiger 2010].

At its most abstract, NetCDF is a means of storing arrays in files. The arrays can be of different sizes and shapes, and contain different kinds of data—strings, integer, real, double, etc. Most VLBI data used in analysis is some kind of array. From this point of view using NetCDF is a natural choice. These files can contain history entries which aid in provenance. Storing data in NetCDF format has the following advantages:

1. Platform/OS/Language Support. NetCDF has interface libraries to all commonly used computer languages running on a variety of platforms and operating systems.
2. Speed. NetCDF is designed to access data fast.
3. Compactness. The data is stored in binary format, and the overhead is low. A NetCDF file is much smaller than an ASCII file storing the same information.
4. Open. NetCDF is an open standard, and software to read/write NetCDF files is freely available.

5. Transportability. NetCDF files use the same internal format regardless of the machine architecture. Access to the files is transparent. For example, the interface libraries take care of automatically converting from big-endian to little-endian.
6. Large User Community. Because of the large user community, there are many tools developed to work with NetCDF files.

Figure 1. A NetCDF file is a container to hold arrays.



6.3.1 NetCDF Attributes.

Another feature of NetCDF is the ability to easily store meta-data related to a variable. This meta data is called an 'attribute' arbitrary, and can be used to store information such as:

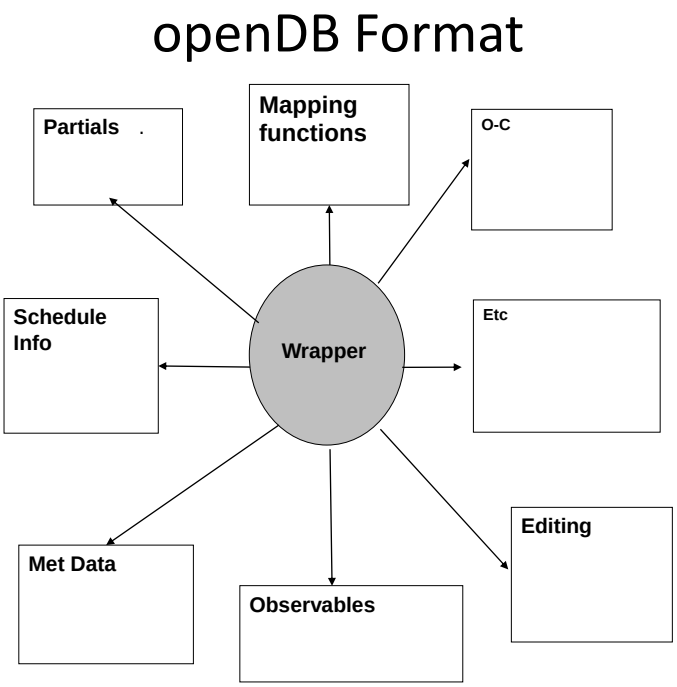
1. Units
2. Definition
3. Creation date of data.
4. Corresponding LCODE name if any.
5. Any other used-specified characteristic.

6.4 Organizing Data Within a Session Using Wrappers

In contrast to the current Mark3 databases where all (actually most) of the data required to analyze a data is one file, the new scheme proposes dividing the data up into smaller pieces. This allows updating the individual pieces separately and gives great flexibility in what is used. Because we split the data into smaller pieces, there must be another means of organizing the data. Wrappers solve this problem.

1. A wrapper is an ASCII file which contains
    a. Information about the session
    b. Pointers to files which contain the actual data

2. A wrapper file is distinguished by the extension ".wrp".
3. Wrappers are never over written. Instead, as the results of analysis, or as needs change, a new wrapper is created.



The wrapper is an ASCII file that contains pointers to other files in netCDF format that contain the data.
Typically each of these other files contain a 'few' data items corresponding to different Lcodes.

Wrappers have many advantages, a few of which are mentioned below.

1. Ability to easily test different models by pointing to different netCDF files.
2. Ability to only update that part of the data which has changed.
3. Ability to try different editing criteria.

6.5 VgosDB Manual

The above is mentioned as an overview of the vgosDB format. It is not meant to completely define the format—instead it is meant to give the flavor of the format. A preliminary manual is avialable via anonymous ftp from:
ftp::/gemini.gsfc.nasa.gov/pub/misc/jmg/vgosDBv_v2013Jun11.pdf
The final version of the manual will be put on the IVS website when it is completed.

7 Feasibility Demonstrations

In August of 2009 John Gipson began a partial implementation of these ideas and wrote software to convert a subset of the data in a Mark3 database into the new format. This subset of data included all of the data available in NGS card format. The subset was chosen because many VLBI analysis packages including Occam, Steelbreeze, and *VieVS* use NGS cards as input. The GSFC VLBI group made available via anonymous FTP an Intensive, an R1 and a RDV session.

In the fall of 2010, Andrea Pany of the Technical University of Vienna developed an interface to VieVS working with the draft proposal. During this process the definition of a few of the data items needed to be clarified, which emphasizes the importance of working with the data hands on. With these changes VieVs was able to process data in the new format without problem.

At roughly the same time at NASA's Goddard Spaceflight center, Sergei Bolotin interfaced a variant of this format to *Steelbreeze.* Steelbreeze uses its own proprietary format, and one motivation for interfacing to the new format was to see if there was a performance penalty associated with using the new format. Bolotin found a small performance penalty of 40μs/ observation. Processing all 7 million then-available VLBI observations would result in a total performance penalty of 280 seconds, or 6 minutes 40 seconds. This seemed to be a relative modest price to pay for the many advantages of this format.

In late 2010 and early 2011 Gipson modified the VLBI analysis program *solve* to use a subset of the data stored in vgosDB format. This subset contained some observation dependent data such as the delay observable and pointing information. The remaining data was extracted from the solve-*superfiles.* (A superfile is a binary file containing a sub-set of all the data in the Mark3 database that is used in global solutions. Superfiles lack the flexibility of Mark3 databases, but data access is much faster.) This test had two distinct purposes. First, it was a demonstration of the feasibility of using netCDF files to store VLBI data. Second, it was a required first step in the conversion of solve to use the new format.

8 Conversion of Mark3 Database to VgosDB format

In 2011 and 2012 Gipson worked on a utility *db2openDB* to convert all of the data in all of Mark3 databases into the openDB format. As mentioned above, this format was subsequently renamed to vgosDB. (A modified version of this program called *db2vgosDB* is available as part of the *calc/solve* distribution.) As there approximately 500 different Lcodes this process took longer than anticipated. In addition many of the Mark3 databases, especially the older ones, had problems that needed to be fixed. A partial listing of some of the problems follows:

1. All of the Mark3 databases have an LCODE 'NUM OBS' that is supposed give the total number of observations in the database. However, several of the older databases actually had fewer observations than indicated, while a few had more.
2. Many of the Mark3 databases had duplicated data. There were two different Lcodes (a type-2 and a type-3 lcode) containing exactly the same data. In these cases the duplicate data was removed.

3. Many databases had bad or incorrect values for some of the data items. Each case had to be examined to determine how to handle the situation. In some cases the missing data could be inferred from data that was present in the database. In other cases the observation had to be flagged as bad.

In the spring of 2012 the conversion was essentially complete. All of the existing Mark3 databases were converted to openDB format and the results made accessible via anonymous ftp at: ftp:://gemini.gsfc.nasa.gov/pub/openDB.

8.1 Conversion of Calc/Solve to Use vgosDB Format

Historically the default storage format for geodetic VLBI sessions is Mark3 databases. This is the data format that the VLBI data is archived in, and the format that many IVS analysis centers use to process and store their results in. Other formats, such as NGS cards, or the custom format used by other analysis packages are all derived either directly or indirectly from the Mark3 databases. The software that produces edited and fully resolved Mark3 databases is the *Calc/Solve* analysis suite. Because of this, as part of the transition to the vgosDB format it is *necessary* to modify Calc/Solve to use the new format.

Figure 3 on the following pages lists the necessary processing steps before VLBI data is ready for use. We give a brief summary of each of the steps below. Depending on the VLBI analysis package the names of the programs involved may differ, but the steps remain the same.

1. The correlator takes the recorded data from the stations and produces correlator output files.
2. *dbedit* reads the correlator files, extracts the relevant information, and produces an initial Mark3 database. It actually produces both an X-band and S-band database.
3. *Calc* adds additional information to the X-band database. Some of this information, such as pointing information, is used by all analysis packages. Other information is specific to the solve.
4. Cable-cal and met-data is added to the X-band database by *dbcal.* The order of steps two and three can be reversed.
5. The data is read into *solve.* An analyst performs an initial solution where they resolve ambiguities and edit the data. The X-band database is written out and now includes a subset of the S-band data.

At this stage the Mark3 database is ready to be used in geodetic analysis. Many analysis packages use NGS-card format as input. (NGS cards contain a subset of the data in the Mark3 database in an ASCII file). Most analysis software, including *calc/solve,* convert the data into a proprietary binary format before using it. In the case of calc/solve this format is called ‘superfiles.

In converting to the vgosDB format we need to develop software that reproduces all of the above steps. Since the Goddard VLBI group is the group that maintains *calc/solve,* this work was undertaken at Goddard.

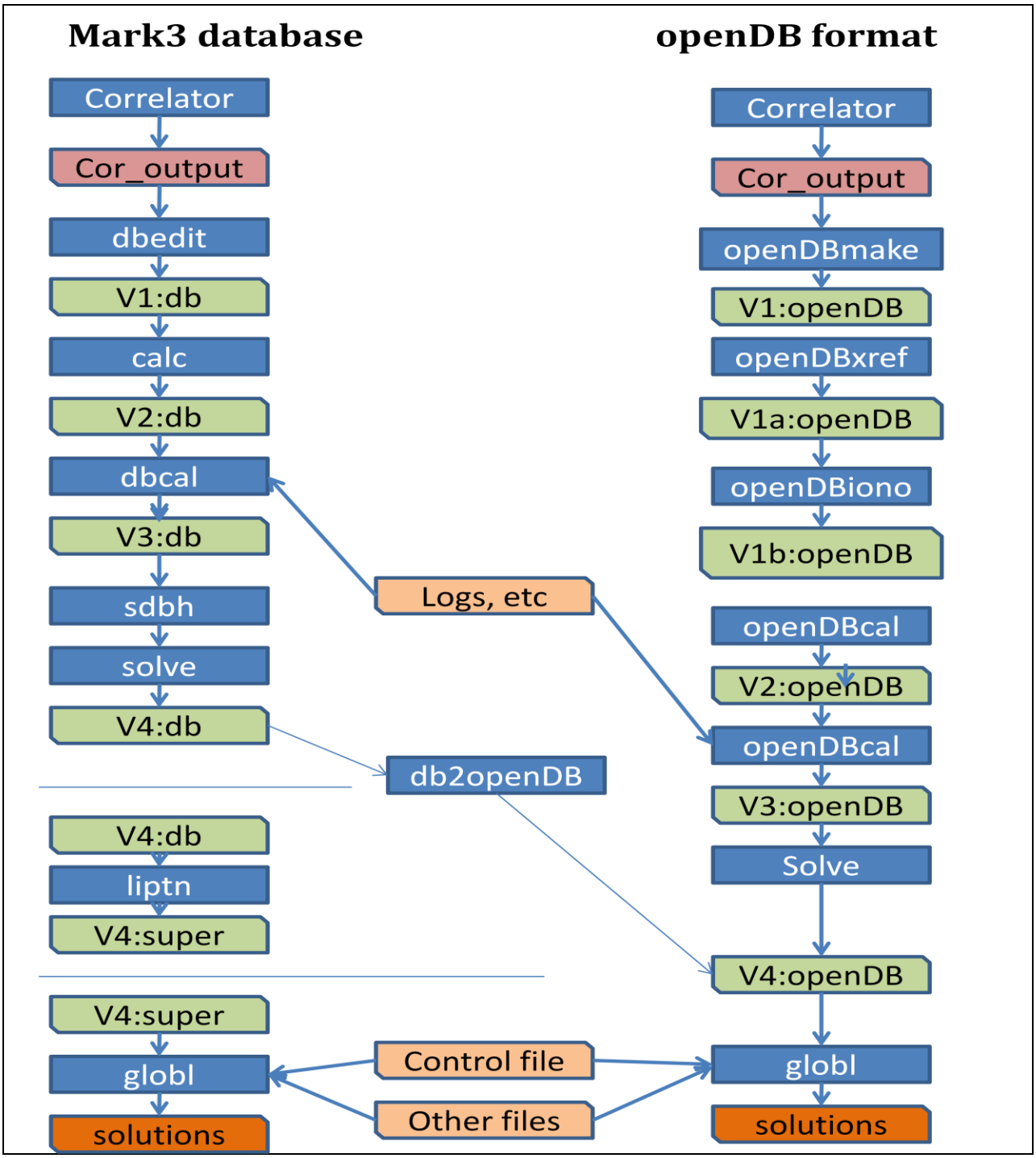


Figure 2. VLBI processing steps.

The approach was to start at the end of the processing chain and work backwards. The first step was to modify the *globl* mode of solve to use the vgosDB format instead of superfiles. The utility *db2vgosDB* read in edited and fully resolved Mark3 databases and produced vgosDB format sessions. Conversion of *globl* to use vgosDB sessions was completed in the summer of 2012. Timing tests showed that smaller sessions such as intensives run somewhat slower using the vgosDB format. In contrast larger sessions such as the RDV sessions or CONT11 sessions run up to 40% faster. A solution using all of the VLBI runs slightly faster using the vgosDB format.

The remaining programs displayed on the right-hand-side of Figure 2 were developed at Goddard during late 2012 and early 2013. Several sessions including an R1s, RDVs and intensives were processed completely, starting from correlator output and ending in fully resolved vgosDB sessions. The results were consistent with or identical with the normal processing using Mark3 databases. (In the process of developing this software the Goddard VLBI group found and fixed some bugs in the normal processing chain.)

The process of developing the processing software was very useful and lead to some refinements in the vgosDB specification. One example of this involves changing which NetCDF files contain some particular data items. For example, in the first implementation of the vgosDB format, both the Names and Positions of stations were stored in the file 'Head.nc' which contains general information about the session. This worked satisfactorily when we were starting with a version 4 a fully resolved Mark3 database which contains all of this information. However if we are creating the vgosDB session by scratch the station position information is not available until after the data has been *calc*-ed. Because of this it is logical to remove the station position (and source position) information from Head.nc and store it separately. Other examples include renaming vgosDB variables and filenames to make them more consistent.

9 Next Steps

In this section we describe next steps in the implementation and decimation of the vgosDB format to the IVS community.

The 'development' version of the *Calc/Solve* analysis suite at Goddard has the ability to use either superfiles or vgosDB files in global solutions. In addition, υSolve, the replacement for interactive solve, can read and write vgosDB files. These capabilities will be made widely available with the next general release of Calc/Solve scheduled for fall 2013.

Beginning in 2014, the Goddard VLBI group will take on responsibility for producing vgosDB versions of all VLBI sessions available to IVS. This includes sessions for which Goddard has primary responsibility (e.g., R1s and RDVs) and also sessions for which other analysis centers have primary responsibility. This will be done by using the utility *db2vgosDB* to convert fully edited and resolved Mark3 databases to vgosDB format. As the year progresses and as other *Calc/Solve* analysis centers (which are responsible for producing the Mark3 databases) get familiar with the handling and producing vgosDB sessions Goddard will reduce the number of vgosDB sessions it produces. For at least 1-year both Mark3 databases and vgosDB sessions will be available on the IVS website.

The VLBI analysis software suite VieVs, developed at the Technical University of Vienna, can use vgosDB sessions in its analysis. This capability will be maintained and enhanced as time permits. For example, currently VieVs must start with a fully resolved Mark3 database. As time permits this capability will be added to VieVs and it will use the vgosDB format to store the related information.

It is hoped that other software packages such as Occam, C5++ are modified to read and write vgosDB format. This will allow the easy exchange of information among these software packages.

10 References

1. Petrov, L. "Specifications of a proposed geo-VLBI dataformat", http://gemini.gsfc.nasa.gov/development/gvf/gvf.html. Undated memo
2. Abdekar, K., Gontier, A-M, "The Gloria VLBI tables", http:
3. Gontier, A-M. and Feissel, M., "PIVEX: a Proposal for a Platform Independent VLBI Exchange Format", The 2nd IVS General Meeting Proceedings, 2002 p. 248-254.
4. Gipson, J., "IVS Working Group 4 on VLBI Data Structures", The 5th IVS General Meeting Proceedings, 2008 p. 143-152
5. Hobiger, et al, "MK3TOOLS – Seamless Interfaces for the Creation of VLBI Databases from Post-Correlation Output", The 5th IVS General Meeting Proceedings, 2008 p. 143-153-156.
6. Gipson, J., "IVS Working Group 4 on VLBI Data Structures", The 6th IVS General Meeting Proceedings, 2010 p. 187-191
7. Hobiger, et al., "C5++ - Multi-technique Analysis Software for Next Generation Geodetic Instruments", The 6th IVS General Meeting Proceedings, 2010 p. 212-216.
8. Gipson, J., Report on IVS-WG4, 20th EVGA Meeting, p 142-146.
9. Gipson, J., IVS Working Group 4 on VLBI Data Structures, The 7th IVS General Meeting Proceedings, 2012 p. 212-221.
10. Gipson, J. IVS Working Group 4 on VLBI Data Structures,
11. Draft openDB Manual version 2013Jun11: ftp::/gemini.gsfc.nasa.gov/pub/misc/jmg/openDBv_v2013Jun11.pdf