

BaBar Data Distribution Tools

User's Manual

This document describes the tools used to transfer BaBar data between the Computer Centers. It is available online from

`http://www.slac.stanford.edu/BFROOT/www/Computing/Offline/DataDist/`

Second Edition

June 2000

Version history:

- 12th August 1999, Jean-Noel Albert
First Edition
- 17th August 1999, Tim Adye
Minor edits, mostly to fix the English. First version for the web.
- 16th June 2000, Moreno Marzolla
Added documentation about metadata information.
- 20th June 2000, Tim Adye
Added BdbMessage commands.

Contents

CONTENTS	III
1 INTRODUCTION	1-1
1.1 Export and Import Overview	1-1
1.2 Document Structure	1-1
1.3 Typographical Conventions	1-1
1.4 Commands Summary	1-2
2 GETTING STARTED	2-1
2.1 Simple Examples	2-1
2.2 Advanced Examples	2-7
2.3 Mastering MetaData.	2-10
2.4 Notes	2-13
3 BDBCOLLEXPOR	3-1
3.1 Abstract	3-1
3.2 Usage	3-1
3.3 Options	3-1
3.4 Description	3-3
3.5 Restrictions	3-3
4 BDBCOLLIMPORT	4-4
4.1 Abstract	4-4
4.2 Usage	4-4
4.3 Options	4-4
4.4 Description	4-5
4.5 Restrictions	4-5
5 BDBCOLLTRANSFER	5-6
5.1 Abstract	5-6
5.2 Usage	5-6
5.3 Options	5-6

5.4	Description	5-7
6	BDBDBIMPORT	6-8
6.1	Abstract	6-8
6.2	Usage	6-8
6.3	Options	6-8
6.4	Commands	6-8
6.5	Description	6-9
6.6	See Also	6-9
7	BDBDISTBROWSER	7-10
7.1	Abstract	7-10
7.2	Usage	7-10
7.3	Options	7-10
7.4	Description	7-11
7.5	See Also	7-11
8	BDBDISTCENTER	8-12
8.1	Abstract	8-12
8.2	Usage	8-12
8.3	Options	8-12
8.4	Description	8-14
8.5	Restrictions	8-14
9	BDBDISTCOLLLOADER	9-15
9.1	Abstract	9-15
9.2	Usage	9-15
9.3	Options	9-15
9.4	Description	9-15
10	BDBDISTCOPY	10-16
10.1	Abstract	10-16
10.2	Usage	10-16
10.3	Options	10-16
10.4	Description	10-17
10.5	See Also	10-17
11	BDBDISTINSTALL	11-18

11.1	Abstract	11-18
11.2	Usage	11-18
11.3	Options	11-18
11.4	Description	11-19
11.5	See Also	11-19
12	BDBDISTMANAGER	12-20
12.1	Abstract	12-20
12.2	Usage	12-20
12.3	Options	12-20
12.4	Commands	12-20
12.5	Script	12-20
12.6	Description	12-20
12.7	See Also	12-21
13	BDBDISTMIRROR	13-22
13.1	Abstract	13-22
13.2	Usage	13-22
13.3	Context Options	13-22
13.4	Options	13-22
13.5	Description	13-23
13.6	Restriction	13-23
14	BDBDISTSCAN	14-24
14.1	Abstract	14-24
14.2	Usage	14-24
14.3	Options	14-24
14.4	Description	14-25
14.5	See Also	14-25
15	BDBDISTSUMMARY	15-26
15.1	Abstract	15-26
15.2	Usage	15-26
15.3	Context Options	15-26
15.4	Options	15-26
15.5	Description	15-27
15.6	See Also	15-27

16	BDBDISTDFILTER	16-28
16.1	Abstract	16-28
16.2	Usage	16-28
16.3	Options	16-28
16.4	Description	16-28
17	BDBDISTWRITETAPE	17-29
17.1	Abstract	17-29
17.2	Usage	17-29
17.3	Options	17-29
17.4	Description	17-29
18	BDBDMNEXPORT	18-30
18.1	Abstract	18-30
18.2	Usage	18-30
18.3	Options	18-30
18.4	Description	18-32
18.5	Restrictions	18-32
19	BDBDMNIMPORT	19-33
19.1	Abstract	19-33
19.2	Usage	19-33
19.3	Options	19-33
19.4	Description	19-34
19.5	Restrictions	19-34
20	BDBSTAGEIN	20-35
20.1	Abstract	20-35
20.2	Usage	20-35
20.3	Options	20-35
20.4	Description	20-35
20.5	See Also	20-36
21	COLLSTAGEIN	21-37
21.1	Abstract	21-37
21.2	Usage	21-37
21.3	Options	21-37
21.4	Description	21-38

21.5	See Also	21-38
22	BDBDISTMANAGER COMMANDS	22-1
22.1	Overview	22-1
22.2	BdbDistManager Built-in Commands	22-1
22.3	Library facilities	22-19
22.4	General Procedures	22-20
23	SITE SPECIFIC CUSTOMIZATIONS	23-1
23.1	Overview	23-1
23.2	Tape Support Functions	23-1
24	TRANSFER DESCRIPTION FORMAT	24-1
24.1	Overview	24-1
24.2	Simplified Example	24-2
24.3	Reading a TDF File	24-3
25	KNOWN PROBLEMS AND TRAPS	25-1

1 Introduction

1.1 Export and Import Overview

The tools described in this note are intended to transfer BaBar data between the computer centers involved in the experiment. The goal is to reproduce locally the organization of the database available at SLAC. The transfers are performed at the files level.

There are 2 different types of transfers: the events of the Events Store; and the other domains, like the Conditions or the Configuration data.

The transfers are performed in 3 steps: the selected databases are *extracted* from the source federation and packed in archives; those archives are copied to the target federation, then they are imported.

Depending of the tool used for the export, the selection key is the name of the event collections or the name of the domains.

During the export, the databases selected are copied to a working directory specified by the user, then they are grouped in archives and optionally compressed using *gzip*. The export tools are able to handle tapes. The user has to specify the labels of the tapes to use and the tool copies to packed databases to those tapes then clean the working directory. If the export is not done on tape, the exported files will stay in the working directory. The user has to copy it to its destination site, for example using FTP.

At the end of the export, a description of the databases, archives and tapes is created in the working directory. This file, named TDF (for Transfer Description File) is the key of the import. It must be copied to the remote site. If the exported files are not exported using tape, they have to be installed in the same directory than the TDF.

The import is driven by the TDF. If the local staging system is supported, the import tools are able to handle tapes. The compressed files are uncompressed, the archives expanded and the databases are copied to the federation disks.

The organization of the original federation is respected. The databases are copied to the same subdirectories as at the source; their database identification number is preserved.

1.2 Document Structure

This section is missing...

1.3 Typographical Conventions

This section is missing...

1.4 Commands Summary

1.4.1 *Export*

- BdbCollExport:** Export the databases containing the data related to a list of collections
- BdbDmnExport:** Export the databases of a list of domains
- BdbDistCenter:** Export the databases of the Events Store based on file criteria, like date of modification
- BdbDistWriteTape:** Save to tapes files exported on disks

1.4.2 *Import*

- BdbCollImport:** Import the Events Store databases exported using *BdbCollExport* or *BdbDistCenter*
- BdbDmnImport:** Import domain databases exported using *BdbDmnExport*

1.4.3 *Direct Transfer*

- BdbCollTransfer:** Directly copy the databases associated to a list of collections from one federation to another
- BdbDistMirror:** Copy the databases from a production federation to an analysis one

1.4.4 *Information*

- BdbDistSummary:** Print a summary about the databases selected for an export, or contained in an import

1.4.5 *Facilities*

- BdbDistTdfFilter:** Filter an existing TDF file, producing a new one
- collstagein:** Download from HPSS tapes databases associated to a list of collections which are not on disk
- bdbstagein:** Download from HPSS tapes databases designed by their system names

1.4.6 *Utilities*

- BdbDistScan:** Display information about the databases associated to a list of collections
- BdbDistBrowser:** Display information about the databases of a list of domains
- BdbDistCopy:** Copy a database from a directory to another under a read lock protection
- BdbDistInstall:** Install a database in the federation
- BdbDistCollLoader:** Attach to the federation the collections defined in a collection database

1.4.7 Internal Utilities

- BdbDbImport:** Copy and install the imported databases in the target federation
- BdbDistManager:** TCL shell providing wrappers around the C++ classes used in the import/export operations

2 Getting Started

2.1 Simple Examples

2.1.1 Collections Transfer

Exporting a Collection

The databases containing the data of the selected collections are exported using *BdbCollExport*.

The databases to export are copied to a working directory by a program (the *PUD daemon*) running on the server of the disks. The working directory must be accessible by this program:

- the working disk must be visible by the PUD daemon: take care of the NFS mounts; ask your system manager about the disks you can use;
- the PUD daemon must be authorized to write in the working directory.

In the following example, a directory is created on a disk traditionally used for the database purpose. Its protection is changed to allow the PUD daemon to write in (at SLAC, the PUD daemon is running in the *bfactory* group).

```
% mkdir /nfs/objyserv1/objy/databases/users/albert/export
% chgrp bfactory /nfs/objyserv1/objy/databases/users/albert/export
% chgrp g+s /nfs/objyserv1/objy/databases/users/albert/export
% chmod g+w /nfs/objyserv1/objy/databases/users/albert/export
```

The name of the collection is given as argument to *BdbCollExport*. Many collections can be specified.

The working directory to use is specified by the `-workdir` option.

```
% BdbCollExport -inc tag -export demol -wait \
  -work /nfs/objyserv1/objy/databases/users/albert/export \
  /groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/99-07-08.06:10:11-021-1
```

```
SLAC local parameters
Export demol starting at 6-Aug-1999 16:43:04
Scanning the collections (can be long...)
Scanning collection /groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/99-07-08.06:10:11-021-1
Organizing the export
Starting to export the databases...
Starting to export the databases...
Creating archive demol.tar.gz (266.265625 MB)
Extracting evs_g_isPhysicsEvents_col004D80: 47.4 MB (wait mode)
16:45: Trying to lock evs_g_isPhysicsEvents_col004D80 (Wait mode)
Extracting evs_g_isPhysicsEvents_evt004D5B: 117.8 MB (wait mode)
16:46: Trying to lock evs_g_isPhysicsEvents_evt004D5B (Wait mode)
Extracting evs_g_isPhysicsEvents_tag004D81: 101.0 MB (wait mode)
```

```

16:48: Trying to lock evs_g_isPhysicsEvents_tag004D81 (Wait mode)
Archiving and compressing to demo1.tar.gz
Effective compression: 74.5 (266.3 MB -> 67.9 MB)
Removing files evs_g_isPhysicsEvents_col004D80.bdb evs_g_isPhysicsEvents_evt004D5B.bdb
evs_g_isPhysicsEvents_tag004D81.bdb
Export demo1 completed at 6-Aug-1999 16:50:36

```

The options begins with a dash (“-”). They can be shorted if there are no ambiguity. For example, **-work** is understood as **-workdir**. However, **-e** would be ambiguous. There is no way to decide if the user want to say **-exclude** or **-export**.

All the tools accept a **-help** option giving the list of the valid options.

At the end of the export, the working directory contains:

- the exported databases, packed in compressed tar archive(s);
- a description of the export: the TDF file;
- a trace of the export (log file).

```

% ls -l /nfs/objyserv1/objy/databases/users/albert/export
total 154144
-rw-r--r-- 1 albert bfactory 4368 Aug 6 16:50 demo1.log
-rw-r--r-- 1 albert bfactory 71150751 Aug 6 16:50 demo1.tar.gz
-rw-r--r-- 1 albert bfactory 1340 Aug 6 16:50 demo1.tdf

```

Notes:

- The **-wait** option is used to allow the export tools to access databases used by other users – in this case, it asks for a read-lock on the database, preventing corruption of the data, then copies the file and releases the lock.

Export Name

The **-export** option names the export. This name is given to the archives, to the TDF file, to the log file, and to the recovery file (see *Recovering from a Crash* page 2-6). Each of those files has a specific extension, but all begin with the export name.

For a collection export, the default export name is the name of the first specified collection, without its directory specification.

For a domain export, the default export name is the name of the first specified domain.

Importing a Collection

The import uses the TDF file created by the export tool. You have to copy the TDF file as well as the exported archives on your site (AFS, FTP...). The exported archives and the TDF file **must** be in the same directory.

```

% BdbCollImport \
  /nfs/objyserv1/objy/databases/users/albert/export/demo1.tdf \
  -work /nfs/objyserv1/objy/databases/users/albert/work
SLAC local parameters
Import demo1 starting at 10-Aug-1999 12:42:07
Uncompressing and expanding /nfs/objyserv1/objy/databases/users/albert/export/demo1.tar.gz
Importing database evs_g_isPhysicsEvents_col004D80
(/nfs/objyserv1/objy/databases/users/albert/work/evs_g_isPhysicsEvents_col004D80.bdb)
Importing database evs_g_isPhysicsEvents_evt004D5B
(/nfs/objyserv1/objy/databases/users/albert/work/evs_g_isPhysicsEvents_evt004D5B.bdb)
Importing database evs_g_isPhysicsEvents_tag004D81
(/nfs/objyserv1/objy/databases/users/albert/work/evs_g_isPhysicsEvents_tag004D81.bdb)

```

```

Removing files /nfs/objyserv1/objy/databases/users/albert/work/evs_g_isPhysicsEvents_col004D80.bdb
/nfs/objyserv1/objy/databases/users/albert/work/evs_g_isPhysicsEvents_evt004D5B.bdb
/nfs/objyserv1/objy/databases/users/albert/work/evs_g_isPhysicsEvents_tag004D81.bdb
Attaching collection /groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/99-07-
08.06:10:11-021-1
Import demo1 completed at 10-Aug-1999 12:49:20

```

The `-workdir` option is used to specify a directory where the databases to import will be unpacked. This directory is automatically cleaned at the end of the import.

Accelerating the Export

Some options can be used to speed up the export (and import) operations:

- Don't pack large databases in tar archives –
The exported databases are automatically grouped in large tar files to optimize the usage of the tapes. If the files are already large, grouping them in an archive just wastes time. However, if you plan to export the files to another site, for example using FTP, reducing the number of files to handle can simplify your copy. The `-archivesize` option allows you to change the maximum size of an archive, and the maximum size of the file packed in such archives.
- Consider the `-nocompress` option –
By compressing the exported files, you save space on the disk, but the compression takes time. However, if you plan to copy the export to another site via the network, compressing the files can save time during the transfer. (But if the transfer method is compressed, e.g., AFS, then this will not help much.)

2.1.2 Conditions and Configuration Transfer

Exporting the Conditions and Configuration Databases

The databases of the Conditions and Configuration domains are exported using *BdbDmnExport*. This tool is very close to *BdbCollExport*, but works with BaBar domain names.

Snapshots of the Conditions and Configuration databases are periodically performed and can be found in the subdirectory `databases/export` of the `babar-ftp.slac.stanford.edu` anonymous server.

If you want to build your own export, you can use something like:

```

% BdbDmnExport -wait -export myConfig \
    -work /nfs/objyserv1/objy/databases/users/my

```

Importing the Conditions and Configuration Databases

To import the Conditions and Configuration databases, you have to use *BdbDmnImport*. Here is a complete sequence for importing the Configuration databases to a private federation at Lyon.

```

% setboot
% mkdir /objy . . . /a/albert/import
% cd /objy . . . /a/albert/import
% ftp babar-ftp.slac.stanford.edu
ftp> . . .
ftp> prompt
ftp> cd databases/export/ . . .
ftp> binary
ftp> mget *
. . .

```

```
ftp> bye
```

```
% BdbDmnImport -replace . . .tdf
```

The **-replace** option instructs the import tool to replace databases already existing in the federation by the newly imported ones. This allows you to upgrade your federation with a recent snapshot.

By default, existing databases are not replaced but an error is signaled.

2.1.3 Tape Transfer

Exporting a Collection on Tape

When the amount of data to export is really large, copying the files using the network is not practical. Tapes have to be used. The export and import tools can handle tapes ¹.

To save your export on tapes, you have to specify their labels with the **-label** option, and the type of those tapes with the **-tapemodel** option. Actually, the supported models are Redwood (~ 50 GB, keyword: **Redwood**) and DLT 3 (~ 10 GB, keyword **DLTIII**) and DLT 4 (~ 35 GB, keyword: **DLTIV**).

```
% BdbCollExport -export demo2 -wait \  
-label MY0001 -tape Redwood \  
-workdir /nfs/objyserv1/objy/databases/users/albert/export \  
/groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/allevnts
```

At the end of the export, the working directory is automatically cleared. Only a log of the export operation and the TDF file stays in this directory. You have to transfer the TDF file yourself (FTP, e-mail...).

Accelerating the Export

Some options can be used to speed up the export (and import) operations:

- Don't pack large databases in tar archives–
Tape systems have better performances with “large files”. How large a large file depends on the tape system. For the Redwood, 50 or 100 MB seems to give “reasonable” performances. You can use for example: **-archivesize 128**.
- Don't compress the files –
Most types of tape hardware are able to automatically compress the files. You have to use the 2 following options: **-nocompress** and **-factor 50**. The first one is to instruct the export tool to not pre-compress the file (using *gzip*). The second one promises to the export tool an efficiency of 50% of the tape compressor². Given this information, the export tool is able to organize the placement of the files on the tapes.

For example, the previous export will be faster with the following command:

```
% BdbCollExport -export demo2 -wait \  
-archive 128 -nocompress -factor 45 \  
-label MY0001 -tape Redwood \  
-workdir /nfs/objyserv1/objy/databases/users/albert/export \  
/groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/allevnts
```

Note:

¹ Currently, the SLAC, CCIN2P3 and RAL staging systems are supported.

² 50% is an example. The compression efficiency can depend of the hardware and of the data. However, 50 seems reasonable for a DLT. 45% is probably more accurate for a Redwood.

The databases are exported tape by tape. This means that the maximum of space you need in the working directory is around the size of the data put on a tape (for example: 100 GB for a Redwood, with a compression factor of ~50%).

If you don't have so much space, you can use the **-diskspace** option. This option specifies the maximum of Megabytes the export can use in the working directory. The databases are copied and organized in tar files, eventually compressed up to reach this limit. Then they are saved on a tape and removed from the disk. The operation continues with the following databases in the same way.

Estimating the Number of Needed Tapes

BdbDistSummary is intended to help you to determine how many tapes are needed by an export. This command accepts most of the **BdbCollExport** command. It prints a report about the selected databases, how they are packed in archives and how those archives are saved on tapes.

```
% BdbDistSummary -export demo2 \  
-archive 128 -nocompress -factor 45 \  
-label MY0001 -tape Redwood \  
-workdir /nfs/objyserv1/objy/databases/users/albert/export \  
/groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/allevnts  
  
SLAC local parameters  
Scanning collection /groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/allevnts  
Organizing the export  
Export summary (estimation)  
Bootfile: /nfs/objyserv1/objy/databases/Production/physics/V1/production/analysis/BaBar.BOOT  
Number of collections: 1  
No pre-compression  
Tape usage: Yes  
Tape model: REDWOOD  
Tape capacity: 50000  
Number of tapes: 1  
Tape labels: MY0001  
Archive capacity: 128 MB  
Number of archives: 9  
Number of databases: 9  
Data size: 88836.4 MB
```

In this example, we have the size of the export: **Data size: 88833.9 MB** (~ 86 GB), and the number of tapes needed: **Number of tapes: 2**.

Importing a Tape Export

The import uses the same **BdbCollImport** command. This time the files are directly read from the tapes. There is no special option to use. All the necessary information is in the TDF file.

However, if the local tape system is not supported, you have to manually download the files to disk, and run the import command using the special **-nostaging** command. **BdbDistSummary**, with its **-details** option, will help you to have a description of the archives, their size and their position on the tapes.

```
% BdbDistSummary -detail demo2.tdf  
SLAC local parameters  
Export summary  
Bootfile: /nfs/objyserv1/objy/databases/Production/physics/V1/production/analysis/BaBar.BOOT  
Number of collections: 1  
No pre-compression  
Tape usage: Yes  
Tape model: REDWOOD  
Tape capacity: 50000  
Number of tapes: 1  
Tape labels: MY0001  
Number of archives: 9  
Number of databases: 9  
Data size: 88835.1 MB
```

```

Tape 1, label: MY0001, size: 88835.1 MB, usage: 177.7 %

Archive 1, Name: demo2-1.tar, Total Size: 47.4 MB
  Database 1, Name: evs_g_isPhysicsEvents_col004D80, Size: 47.4 MB

Archive 2, Name: demo2-2.tar, Total Size: 117.8 MB
  Database 2, Name: evs_g_isPhysicsEvents_evt004D5B, Size: 117.8 MB

Archive 3, Name: demo2-3.tar, Total Size: 104.0 MB
  Database 3, Name: evs_g_isPhysicsEvents_tag004D81, Size: 104.0 MB

Archive 4, Name: demo2-4.xdb, Total Size: 619.0 MB
  Database 4, Name: evs_g_isPhysicsEvents_evshdr004D51, Size: 619.0 MB

Archive 5, Name: demo2-5.xdb, Total Size: 44663.8 MB
  Database 5, Name: evs_g_isPhysicsEvents_rec004C77, Size: 44663.8 MB

Archive 6, Name: demo2-6.xdb, Total Size: 981.4 MB
  Database 6, Name: evs_g_isPhysicsEvents_aod004D98, Size: 981.4 MB

Archive 7, Name: demo2-7.xdb, Total Size: 417.8 MB
  Database 7, Name: evs_g_isPhysicsEvents_rawhdr004D3B, Size: 417.8 MB

Archive 8, Name: demo2-8.xdb, Total Size: 41546.3 MB
  Database 8, Name: evs_g_isPhysicsEvents_raw004C76, Size: 41546.3 MB

Archive 9, Name: demo2-9.xdb, Total Size: 337.5 MB
  Database 9, Name: evs_g_isPhysicsEvents_esd004D50, Size: 337.5 MB

```

2.1.4 Recovering from a Crash

Recovering after a crash is automatic. All the operations successfully executed are traced in a recovery file, saved in the working directory. This file has the name of the export with the `rec` extension.

The only thing to do is to restart the same command, or at least specifying the same working directory and export name. Optionally, the `-recover` option can be used to specify the recovery file to use.

Caution: in a recovery context, the scanning of the federation and the organization of the export are not done. If the previous operation failed because you have been too optimistic with the hardware compressor, for example, or if the tapes are not available, the recovery can't help you. In such a case, you have to manually delete the recovery file, or use the `-norecover` option, and restart the operation from the beginning.

```

% BdbCollExport -export demo2 -wait -label MY0001 -tape Redwood \
  -nocomp -archive 128 -factor 45 \
  -workdir /nfs/objyserv1/objy/databases/users/albert/export \
  /groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/allevnts
SLAC local parameters
Export demo2 starting at 11-Aug-1999 11:05:52
Scanning the collections (can be long...)
Scanning collection /groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/allevnts
Organizing the export
Starting to export the databases...
Starting to export the databases...
Creating tape MY0001 (88835.078125 MB)
Creating archive demo2-1.tar (47.421875 MB)
Extracting evs_g_isPhysicsEvents_col004D80: 47.4 MB (wait mode)
11:06: Trying to lock evs_g_isPhysicsEvents_col004D80 (Wait mode)
Archiving to demo2-1.tar
Removing files evs_g_isPhysicsEvents_col004D80.bdb
Creating archive demo2-2.tar (117.84375 MB)
Extracting evs_g_isPhysicsEvents_evt004D5B: 117.8 MB (wait mode)
11:06: Trying to lock evs_g_isPhysicsEvents_evt004D5B (Wait mode)
Archiving to demo2-2.tar
Removing files evs_g_isPhysicsEvents_evt004D5B.bdb
Creating archive demo2-3.tar (104.046875 MB)
Extracting evs_g_isPhysicsEvents_tag004D81: 104.0 MB (wait mode)

```

```
11:07: Trying to lock evs_g_isPhysicsEvents_tag004D81 (Wait mode)
...crash...
```

The working directory contains:

```
% ls -l /nfs/objyserv1/objy/databases/users/albert/export
total 617488
-rw-r--r--  1 albert  bfactory 49726976 Aug 11 11:06 demo2-1.tar
-rw-r--r--  1 albert  bfactory 123569664 Aug 11 11:07 demo2-2.tar
-rw-r--r--  1 albert  bfactory  10181 Aug 11 11:07 demo2.log
-rw-r--r--  1 albert  bfactory   5915 Aug 11 11:07 demo2.rec
-rw-r--r--  1 objysrv bfactory 109101056 Aug 11 11:08 evs_g_isPhysicsEvents_tag004D81.bdb
```

To recover, just executes the same command:

```
% BdbCollExport -export demo2 -wait -label MY0001 -tape Redwood \
  -workdir /nfs/objyserv1/objy/databases/users/albert/export \
  /groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/allevnts \
  -nocomp -archive 128 -factor 45
SLAC local parameters
Export demo2 starting at 11-Aug-1999 11:08:41
Recovering from a previous attempt
Starting to export the databases...
Starting to export the databases...
Creating tape MY0001 (88835.078125 MB)
Creating archive demo2-3.tar (104.046875 MB)
Extracting evs_g_isPhysicsEvents_tag004D81: 104.0 MB (wait mode)
11:08: Trying to lock evs_g_isPhysicsEvents_tag004D81 (Wait mode)
Archiving to demo2-3.tar
Removing files evs_g_isPhysicsEvents_tag004D81.bdb
Creating archive demo2-4.xdb (619.015625 MB)
Extracting evs_g_isPhysicsEvents_evshdr004D51: 619.0 MB (wait mode)
```

This time, the extract operations begin with the `evs_g_isPhysicsEvents_tag004D81` database. The tool has seen that the archives containing the previously exported databases are still in the working directory. It also knows that the last exported database does not have the right size (it would have 110575616 bytes). Therefore, it restarts the export with this database and continues the operations from the previous crash.

2.2 Advanced Examples

2.2.1 Massive Exports

The major problem with the export is the time wasted to get the list of the databases and of the collections to export. This is due to some limitations in the AMS Server used to access the data in the databases.

To work around this problem, the solution is to scan the federation on the nodes where the databases are stored then to export the files on the computer hosting the tapes. The following sequence is used:

- Scanning the databases and collections on the `datamove2` and `shire01` disk servers
- Creating a valid TDF file using *BdbDistTdfFilter*
- Selecting the data to export to fit the space available on the disks of `datamove3`
- Exporting the databases on its disks
- Copying the exported databases to DLT and/or Redwood for the external centers

We will see these commands step by step.

Scanning the Federation

The Physics Analysis federation uses 2 disk servers: `datamove2` and `shire01`. Similar commands are executed on the both computers:

- the databases to export are selected in the *isPhysicsEvents* group;
- the scan is restricted to the local host;
- only the databases modified since the last massive export are selected;
- and only the databases smaller than 2 GB are selected.

Datamove2 scan:

```
% ssh datamove2
% analboot
% BdbDistBrowser -verb \
    Events \
    -since '21 Jul' \
    -group isPhysicsEvents \
    -node datamove2 \
    -maxsize 2047 |& tee isPhysics-dm2.scn
```

Shire scan:

```
% ssh shire01
% analboot
% BdbDistBrowser -verb \
    Events \
    -since '21 Jul' \
    -group isPhysicsEvents \
    -local \
    -maxsize 2047 |& tee isPhysics-shire.scn
```

Both commands use *BdbDistBrowser*. The tool iterates over all the databases of a specified domain and checks if they match the specified criteria. Here, we want to export the databases containing the events, therefore we are asking for the **Events** domain.

When *BdbDistBrowser* encounters a database containing the definition of events collections, it scans this database and adds the description of the collections to the output.

The `-verbose` option prints many information to the output. It is useful to try to understand what happens when something is wrong. Piping the output to the *tee* command allows the user to see the progression of the scanning.

The `-local` option can be used to restrict the selection to the databases stored on the *local* node only, whatever can be this node.

Organizing the TDF

2 files are created: `isPhysics-dm2.scn` and `isPhysics-shire.scn`. These 2 files are jointed in an unique file, converted to a valid TDF file using *BdbDistTdfFilter*.

```
% cat isPhysics-dm2.scn isPhysics-shire.scn > isPhysics-incr.scn
% BdbDistTdfFilter isPhysics-incr.scn -ignore > isPhysics-incr.tdf
```

The `-ignore` option is used to inform the tool that the input file is not really a valid TDF file, and it can avoid complaining about missing checksums.

Selecting the Databases to Export

Using *BdbDistSummary*, we will search how to reduce the export to fit with the space available in the working directory.

For this example, we will suppose the space available is around 100 GB.

```
% BdbDistSummary isPhysics-incr.tdf
SLAC local parameters
Export summary
Bootfile: /nfs/objyserv1/objy/databases/users/albert/tst820/BaBar.BOOT
Number of collections: 35827
No pre-compression
Number of archives: 0
Number of databases: 459
Data size: 636516.8 MB
```

The first try shows too much data to export. We will reduce the size to the files modified since August 1st.

```
% BdbDistTdfFilter isPhysics-incr.scn -ignore -since '1 Aug' > isPhysics-incr-Aug1.tdf
% BdbDistSummary isPhysics-incr-Aug1.tdf
SLAC local parameters
Export summary
Bootfile: /nfs/objyserv1/objy/databases/users/albert/tst820/BaBar.BOOT
Number of collections: 35827
No pre-compression
Number of archives: 0
Number of databases: 17
Data size: 5641.5 MB
```

If there are still too many data, we can generate 2 or 3 exports: one for the fast data (tag, aod, esd), one for the rec, and one for the raw.

```
% BdbDistSummary isPhysics-incr-Aug1.tdf -include aod:esd:tag
```

The export will be done using the same **-include** option.

Exporting the Databases on Disk

The selected databases are exported with *BdbCollExport*. No tapes are specified and the files will stay on disk.

```
% BdbDistCenter -wait -archive 250 -nocomp -lftp \
    -export isPhysics-incr-Aug1 \
    -work /objy/databases/test/
    -use isPhysics-incr-Aug1.tdf
SLAC local parameters
Please enter your password:
Export isPhysics-incr-Aug1 starting at 9-Aug-1999 17:46:30
Starting to export the databases...
(...)
17:58: Trying to lock evs_g_isPhysicsEvents_evt00805E (Wait mode)
Local directory now /objy/databases/test/workdir
Archiving to isPhysics-incr-Aug1-13.tar
Removing files evs_g_isPhysicsEvents_col0052B1.bdb evs_g_isPhysicsEvents_evt00805E.bdb
Export isPhysics-incr-Aug1 completed at 9-Aug-1999 17:59:53
36.78u 200.90s 13:25.62 29.5%
```

Because the disks used to store the databases are not NFS exported, you can't simply copy them to the local disk of datamove3. FTP is the only way (at this time) to copy the databases from one datamover to another. You are asked at the beginning of the export for your password. Then, the tool is able to copy the files to the local disk and create the archives.

The usage of the **-lftp** option is restricted to the node where the archives have to be created. In this example, the command must be issued on datamove3. Of course, you have to be allowed to login to this computer.

Saving the Export on Tape

The *BdbDistWriteTape* tool is used to save this export on a Redwood then on a DLT.

```

% BdbDistWriteTape -export BH0025 \
  -tape Redwood -label BH0025 \
  -tdf /objy/databases/test/workdir/isPhysics-BH0025-Aug1.tdf
Aug  9 19:07:48 tpwrite[43022]: selecting tape server ...
Aug  9 19:07:48 tpwrite[43022]: * datamove3 is a possible tape ♦
Aug  9 19:07:48 tpwrite[43022]: ! selected tape server is datamove3.

CPDSKTP - TAPE MOUNTED ON UNIT tpdd32
CPDSKTP - RECORD FORMAT: F
CPDSKTP - BLOCK SIZE: 262144
CPDSKTP - RECORD LENGTH: 262144
CPDSKTP - END OF TRANSFER

CPDSKTP - 193085440 BYTES COPIED
CPDSKTP - 737 RECORDS COPIED
CPDSKTP - DATA TRANSFER BANDWIDTH (188560 Kbytes through unknown) ♦
(...)

% BdbDistWriteTape -export RM0002 \
  -tape DLTIV -label RM0002 \
  -tdf /objy/databases/test/workdir/isPhysics-RM0002-Aug1.tdf
(...)

```

2.2.2 Import

The import is done using *BdbCollImport*. There is no difference with a standard import.

2.3 Mastering MetaData.

2.3.1 What are MetaData?

Scanning a collection can be a very time-consuming task. This is because every persistent object somehow accessible from a collection (events, headers, components...) has to be accessed to get its DbID.

To save time, it is possible to store the list of DbIDs inside a persistent object which can be associated to each collection. Such objects are called metadata. Using metadata, the first time a collection is scanned, its DbID list can be stored inside a persistent object associated with the collection. As long as the metadata remain valid (more details later), subsequent scans can take advantage of the existing lists, thus being much faster.

2.3.2 Automatic MetaData handling.

There are operations which can alter the list of databases referenced by a collection, such as inserting or removing events or other collections. It's not yet possible to dynamically update the metadata, to keep them consistent after every kind of collection update. The simple strategy of invalidating them automatically has been developed. Basically, whenever an event is added/detached from a collection X, or another collection is inserted or removed from X, then the metadata informations associated with X are marked as *invalid*. This is needed to inform the collections scanner that something has changed inside the collection, so it is necessary to check it from scratch.

2.3.3 Basic examples.

Suppose you have just created some collections (in this example, with *BdbEvsTest*):

```

/system/
  SysColl: 20 events

/system/SubDir/
  SysColl2: 0 events

/users/

/users/marzolla/
  UserColl: 20 events

/users/marzolla/MyDir/
  UserColl2: 40 events

```

First, we check the status of the metadata:

```

% BdbDistScan -printmd /
MESSAGE 'Scanning tree node / ...'
/system/SubDir/SysColl2
9218-4-1-6 0 { }
/system/SysColl
/users/marzolla/MyDir/UserColl2
/users/marzolla/UserColl

```

From this we can see that the collection /system/SubDir/SysColl2 has a metadata object, which is reported as invalid (the “0” before the braces). Other collections don’t have metadata at all.

To update the metadata, we can use *BdbDistScan*:

```

%BdbDistScan -updatemd /
MESSAGE 'Scanning tree node / ...'
MESSAGE 'Scanning collection 9218-4-1-6 ...'
MESSAGE 'Scanning collection 9218-3-1-2 ...'
MESSAGE 'Scanning collection 9223-3-1-5 ...'
MESSAGE 'Scanning collection 9223-3-1-2 ...'
evs_s_bb_col002402 9218
evs_s_bb_evt002403 9219
evs_s_bb_tag002404 9220
evs_s_bb_rawhdr002406 9222
evs_s_bb_simhdr002405 9221
evs_u_marzolla_col002407 9223
evs_u_marzolla_evt002408 9224
evs_u_marzolla_tag002409 9225
evs_u_marzolla_rawhdr00240B 9227
evs_u_marzolla_simhdr00240A 9226

```

Now, checking again, we get:

```

%BdbDistScan -printmd /
MESSAGE 'Scanning tree node / ...'
/system/SubDir/SysColl2
9218-4-1-6 1 { }
/system/SysColl
9218-4-1-2 1 { 9219 9220 9221 9222 }
9218-4-1-4 1 { 9219 9220 9221 9222 }
/users/marzolla/MyDir/UserColl2
9223-4-1-6 1 { 9224 9225 9226 9227 }
9223-4-1-8 1 { 9224 9225 9226 9227 }
9218-4-1-2 1 { 9219 9220 9221 9222 }
9218-4-1-4 1 { 9219 9220 9221 9222 }
/users/marzolla/UserColl
9223-4-1-2 1 { 9224 9225 9226 9227 }
9223-4-1-4 1 { 9224 9225 9226 9227 }

```

This listing should be read as follows: after each collection name, there is one line for each contained *vector collection* (currently, only vector collections can have metadata associated; however, this is an implementation

detail and is subject to change). Each line contains the collection ID in Objectivity format, the validity flag (1=metadata are valid, 0=metadata are invalid), and the list of the Ids of databases associated with that collection. In the example above, note that for the collection /system/SubDir/SysColl2, the DbID list is empty, but the metadata object is reported as being valid. This means that the collection is empty (as may be checked with *BdbInspector*).

Now, suppose we delete all the events from /users/marzolla/MyDir/UserColl2. We will end up in the following situation:

```
% BdbDistScan -printmd /
MESSAGE 'Scanning tree node / ...'
/system/SubDir/SysColl2
9218-4-1-6 1 { }
/system/SysColl1
9218-4-1-2 0 { }
9218-4-1-4 0 { }
/users/marzolla/MyDir/UserColl2
9223-4-1-6 0 { }
9223-4-1-8 0 { }
9218-4-1-2 0 { }
9218-4-1-4 0 { }
/users/marzolla/UserColl1
9223-4-1-2 1 { 9224 9225 9226 9227 }
9223-4-1-4 1 { 9224 9225 9226 9227 }
```

The metadata associated with each vector collection of /users/marzolla/MyDir/UserColl2 appears now as invalid. At this point, it is possible to update the metadata with the following command:

```
% BdbDistScan -updatemd -trustmd /
MESSAGE 'Scanning tree node / ...'
MESSAGE 'Scanning collection 9218-4-1-6 ...'
MESSAGE 'Scanning collection 9218-3-1-2 ...'
MESSAGE 'Scanning collection 9223-3-1-5 ...'
MESSAGE 'Scanning collection 9223-3-1-2 ...'
evs_s_bb_col002402 9218
evs_u_marzolla_col002407 9223
evs_u_marzolla_evt002408 9224
evs_u_marzolla_tag002409 9225
evs_u_marzolla_simhdr00240A 9226
evs_u_marzolla_rawhdr00240B 9227
```

In this case, the **-updatemd** is used together with the **-trustmd** flag. This means that only collections whose metadata are invalid will be inspected. Using the **-updatemd** flag alone implies a complete scan of each collection, regardless of the validity state of its metadata.

2.3.4 Using MetaData with BdbCollExport.

BdbCollExport understands both the **-updatemd** and **-trustmd** flag. So, it is possible to write something like this:

```
% BdbCollExport -export myTest -updatemd -trustmd -work ./tmp /users
```

In this way, *BdbCollExport* will use valid metadata to speed up the scan phase, and will also update invalid metadata. Please note that the default behaviour of both *BdbDistScan* and *BdbCollExport* is to *not* use nor update the metadata.

2.4 Notes

Format of the Dates

Dates follow the TCL convention: a date can be given using an absolute or a relative form. Absolute dates can specify the day, month (year), date and hour. Relative dates are in the form of a number of “common” keywords, like minute, hour, day, week, month...

Examples:

```
-since '14 July'  
-before '2 hours'  
-since 'Aug 6'
```

Format of the Lists

A list of values is specified using one of these two forms: values concatenated by “:” or inside of a string enclosed by quotes.

Examples:

```
-include tag:aod:esd  
-exclude "raw rec"  
-label TP0001:TP0002:TP0003
```

Names of the components

The components specified using **-include** or **-exclude** can be the standard components defined by BaBar or some alias defined in the export tools.

The known standard components are the following: **tag, aod, esd, rec, raw, sim, tru**.

The following aliases are defined:

- **nanodst** = tag
- **fast** = tag + aod
- **microdst** = aod
- **minidst** = esd
- **dst** = rec

Note: even if not explicitly specified, the **col** and the **evt** databases are always exported. The header databases associated to a specified component are also copied (e.g. the **rawhdr** databases are selected when the **raw** component is specified).

3 BdbCollExport

3.1 Abstract

Export the databases containing the data related to a list of collections.

3.2 Usage

```
BdbCollExport [options] collection...
```

3.3 Options

- [no]archivesize **MB**: maximum size of the archives to create, in MB, or disable the creation of archives, if the **-noarchive** form is used;
the default value is 1800 (something smaller than 2 GB);
- before **date**: select only the databases modified before the specified date (see *Format of the Dates* page 2-13);
- bootfile **file**: name of the boot file of the federation;
the default value is given by the environment variable **\$OO_FD_BOOT**;
- checkonly: just print a report describing the selected databases, but don't export anything;
- [no]compress: compress or not compress the created archives;
the default behavior is to compress the archives;
- configure **file.cfg**: private configuration file;
by default, the standard configuration file is read (see *Chapter Site Specific Customizations* page 23-1);
- delay **minutes**: time to wait for a read lock on the base;
by default, there is no delay;
- diskspace **MB**: maximum space available in the working directory for the export operation;
by default, the working directory is supposed large enough for the operation;
- exclude **components-list**: list of type of data to exclude from the export (see *Format of the Lists* page 2-13);
by default, all the types of data are exported;
- export **name**: name of the export – this name is used to build the name of Export Description File (TDF);
the default name is the name of the first collection;
- factor **rate**: estimation of the factor of compression of the data (see *Tape Transfer* page 2-4);
by default, a factor of 40 % is assumed;

- from *bootfile***: alias for **-bootfile**;
- help**: description of the available options;
- include *components-list***: alias for **-items**;
- items *components-list***: selection of types of data to export (see *Format of the Lists* page 2-13);
by default, all the type data are exported;
- labels *label-list***: labels of the tapes to use to export the databases (see *Format of the Lists* page 2-13);
by default, the exported databases stay on disk and are not copied on tape
- lftp**: use FTP to copy the databases from the federation to the working directory;
by default, the files are copied using *ooattachdb* or the UNIX *cp* command, depending of the usage or note of the **-wait** option;
- local *name***: specify the name of a pre-define configuration – actually, only *slac* and *ccin2p3* are defined;
the default is based on the value of the environment variable **\$BFSITE**;
if the corresponding configuration does not exist, a default one is used (see Chapter *Site Specific Customizations* page 23-1)
- [no]log *logfile-name***: define the name of the log file to create, or if the form **-nolog** is used, explicitly disable the creation of such a log file;
the default name is the name of the export (see **-export**) with the extension: *.log*
- maxsize *MB***: restrict the export to the databases smaller than the given value, expressed in Megabytes;
by default, there is no restriction;
- minsize *MB***: restrict the export to the databases larger than the given value, expressed in Megabytes;
by default, there is no restriction;
- [no]recover *recfile***: give the name of a recovery file to read at the beginning of the export operation, or if the form **-norecover** is used, explicitly disable the loading of such a recovery file;
by default, the recovery file is the name of the export export (see **-export**) with the extension: *.rec*;
if such a file exists in the working directory, it is automatically loaded (see *Recovering from a Crash* page 2-6);
- since *date***: select only the databases modified since the given date (see *Format of the Dates* page 2-13);
- summary**: print a summary of the export operation;
- tapemodel *name***: specify the model of tapes used to export the database – this option must be specified if the option **-labels** is used;
- [no]trustmd**: tell *BdbDistScan* to scan the database using the metadata informations associated with the collections; this greatly speeds up this operation.
by default, metadata are not used;
- [no]updatemd**: tell *BdbDistScan* to update invalid metadata informations during the scan. If used in conjunction with **-trustmd**, collections with valid metadata will not be updated. Otherwise, each collection will be completely checked, and each metadata object will be updated, regardless of the validity status.
by default, metadata are not updated;
- use *input-tdf***: use an existing TDF file describing the candidate databases to export – the selection options can be used to reduce this list – the federation is not scanned;
by default, the databases are searched in the federation using *BdbDistScan*;

- verbose**: print additional information about the export operations;
 - by default, only a minimal amount of information are printed;
 - the “verbose” information are automatically saved in the log file, even if the **-verbose** option is not used (but they don't appeared on the screen);
- [no]wait**: ask the program used to extract the database from the federation to proceed under the protection of a read-lock;
 - by default, the database is extracted using *ooattachdb* and is not locked;
- workdir dir**: name of a directory to use as work area to copy the databases extracted from the federation, create the archives and compressed them (depending of the used options);
 - if the export doesn't use tapes, the files will stay in this directory at the end of the export, else, they will be copied on the tapes and removed from this directory;
 - the space on the disk must be large enough to receive the exported files, or the **-diskspace** option must be used to specify how many Megabytes can be used

3.4 Description

This section is missing

3.5 Restrictions

If the **-lftp** is used, the export operation must be executed on a computer where the *working directory* is mounted.

4 BdbCollImport

4.1 Abstract

Export the databases of a list of domains

4.2 Usage

```
BdbCollImport [options] description-file
```

4.3 Options

- bootfile** *file*: name of the boot file of the federation;
the default value is given by the environment variable `$OO_FD_BOOT`;
- checkonly**: just print a report describing the selected databases, but don't export anything;
- continue**: ignore the errors and try to continue the import;
by default, the import tool will stop if an error is encountered;
- delay** *minutes*: time to wait for an update lock on the base to replace, or for the collection databases;
by default, there is no delay;
- exclude** *components-list*: list of type of data to ignore during the import (see *Format of the Lists* page 2-13);
by default, all the types of data are imported;
- help**: description of the available options;
- include** *components-list*: alias for –**items**;
- items** *components-list*: restrict the import to the types of data specified (see *Format of the Lists* page 2-13);
by default, all the data are imported;
- lftp**: use FTP to copy the databases on disks of the federation;
by default, the files are copied the UNIX *cp* command (see *Restrictions* page 4-5);
- local** *name*: specify the name of a pre-define configuration – actually, only `slac` and `ccin2p3` are defined;
the default is based on the value of the environment variable `$BFSITE`;
if the corresponding configuration does not exist, a default one is used (see Chapter *Site Specific Customizations* page 23-1)
- [no]log** *logfile-name*: define the name of the log file to create, or if the form –**nolog** is used, explicitly disable the creation of such a log file;
the default name is the name of the export (see –**export**) with the extension: *.log*

- nostaging**: don't use the staging system to access the files and search them in the working directory
- [no]recover *recfile***: give the name of a recovery file to read at the beginning of the export operation, or if the form **-norecover** is used, explicitly disable the loading of such a recovery file;
 - by default, the recovery file is the name of the export export (see **-export**) with the extension: **.rec**;
 - if such a file exists in the working directory, it is automatically loaded (see *Recovering from a Crash* page 2-6);
- replace**: replace any existing databases or collections by the imported ones;
 - by default, the existing databases or collections are not replaced and an error is signaled;
- stagesize *MB***: maximum space available in the disks used to stage-in the databases to import;
 - by default, the staging system is supposed to have enough space for the operation;
- to *bootfile***: alias for **-bootfile**;
- verbose**: print additional information about the export operations;
 - by default, only a minimal amount of information are printed;
 - the “verbose” information are automatically saved in the log file, even if the **-verbose** option is not used (but they don't appeared on the screen);
- [no]wait**: wait for an update lock before trying to import a database or to attach a collection;
 - by default, the import program does not wait;
- workdir *dir***: name of a directory to use as work area to uncompress and expand the archives;
 - the imported files are automatically removed at the end of the import;

4.4 Description

This section is missing

4.5 Restrictions

–**lftp** must be used on the computer where the working directory is mounted, or the user must use the same account than the PUD daemon and AMS server.

5 BdbCollTransfer

5.1 Abstract

Copy the databases associated to one or more collections from one federation to another.

5.2 Usage

```
BdbCollTransfer [options] collection... [to]
```

5.3 Options

- continue**: ignore the errors and try to continue the import;
by default, the import tool will stop if an error is encountered;
- delay minutes**: time to wait for an update lock on the base to replace, or for the collection databases;
by default, there is no delay;
- diskspace MB**: maximum space available in the working directory for the export operation;
by default, the working directory is supposed large enough for the operation;
- exclude components-list**: list of type of data to exclude from the export (see *Format of the Lists* page 2-13);
by default, all the types of data are exported;
- from boot-filefile**: boot file of the source federation;
the default value is given by the environment variable `$OO_FD_BOOT`;
- help**: description of the available options;
- items components-list**: selection of types of data to transfer (see *Format of the Lists* page 2-13);
by default, all the type data are transferred;
- [no]log logfile-name**: define the name of the log file to create, or if the form **–nolog** is used, explicitly disable the creation of such a log file;
the default name is the name of the export (see **–export**) with the extension: *.log*;
- [no]recover recfile**: give the name of a recovery file to read at the beginning of the export operation, or if the form **–norecover** is used, explicitly disable the loading of such a recovery file;
by default, the recovery file is the name of the export export (see **–export**) with the extension: *.rec*;
if such a file exists in the working directory, it is automatically loaded (see *Recovering from a Crash* page 2-6);
- replace**: replace any existing databases or collections by the imported ones;

- by default, the existing databases or collections are not replaced and an error is signaled;
- to boot-file**: boot file of the target federation;
 - by default, the last parameter is used;
- verbose**: print additional information about the export operations;
 - by default, only a minimal amount of information are printed;
 - the “verbose” information are automatically saved in the log file, even if the –**verbose** option is not used (but they don't appeared on the screen);
- [no]wait**: export the databases from the source federation under a read lock protection, and wait for an update lock during the import of the databases and collections;
 - by default, the export and import operations don't wait;
- workdir dir**: a directory to use as work area to copy the databases extracted from the source federation;
 - the files are removed after their copy to the target federation;
 - the space on the disk must be large enough to receive the exported files, or the –**diskspace** option must be used to specify how many Megabytes can be used

5.4 Description

This section is missing

6 BdbDbImport

6.1 Abstract

Copy and install in the target federation the imported databases

6.2 Usage

`BdbDbImport` [*options*]

6.3 Options

- [no]checksize**: enable or disable the control of the file of the imported database;
by default, the size of the files is checked after the copy to their directory;
- delay *minutes***: time to wait for an update lock on the base to replace, or for the collection databases;
by default, there is no delay;
- group *name***: name of the UNIX group to use for the files;
- ignore**: ignore the errors and try to continue the import;
by default, the import tool will stop if an error is encountered;
- lftp**: use FTP to copy the databases on disks of the federation;
by default, the files are copied the UNIX *cp* command;
- quiet**: don't print any information about the executed operations;
- [no]read**: turn the files in read-only mode, or, if the form **-noread** is used, in write mode (this is the default);
- replace**: replace any existing databases or collections by the imported ones;
by default, the existing databases or collections are not replaced and an error is signaled;
- verbose**: print additional information about the export operations;
by default, only a minimal amount of information are printed;
- [no]wait**: wait for an update lock before trying to import a database or to attach a collection;
by default, the import program does not wait;

6.4 Commands

BdbDbImport reads the commands to execute from the standard input

The supported commands are:

- **IMPORT** *dbsysname dbid filename subdir size*

Parameters of a database to import:

- **dbsysname:** system name of the database;
- **dbid:** database identification number;
- **filename:** name of the file to import;
- **subdir:** subdirectory of the database in the original federation
- **size:** size of the database, in Megabytes;

- **COLLECTION** *collection oid*

Parameters of a collection to define:

- **collection:** complete path name of the collection;
- **oid:** object identification of the collection;

6.5 Description

BdbDbImport is a C++ program used by the import tools to import the database files to their right location, to attach the databases to the federation and to define the imported collections.

Before starting to import a database, *BdbDbImport* checks if a database with the same name already exists. If such a database exists and the **–replace** option is not used, it stops with an error message, if the **–ignore** option is not used. If the **–replace** option is used, the existing database will be replaced by the new one.

To know where the database has to be installed, *BdbDbImport* asks the clustering system. This system gives a base directory and a host name. The database is installed in the same subdirectory than in the original federation under this directory specified by the clustering system on indicated host. This will maintain the same structure of directories in the local federation than in the original one.

If an existing database has to be replaced by a new one, the new file is copied to its location with the extension **bdbimp**, then the existing database is replaced by the new one under the protection of an update lock.

The collections are define by creating the tree node corresponding to the collection and associating this tree node to the persistent object designed by its object id. If the corresponding object id. is not valid (for example, because the database has not been installed), an error message is printed and the collection is ignored.

6.6 See Also

BdbCollImport, BdbDmnImport, BdbCollTransfer

7 BdbDistBrowser

7.1 Abstract

Display information about the databases of a domain

7.2 Usage

```
BdbDistBrowser [options] domain...
```

7.3 Options

- before date**: select only the databases modified before the specified date (see *Format of the Dates* page 2-13);
- [no]collections**: scan the collection databases during the scanning of the federation to be able to recreate the collections during the import operation, or skip this operation (to save time) if the **-nocollections** form is used;
- exclude items-list**: list of type of data to exclude from the export (see *Format of the Lists* page 2-13);
by default, all the types of data are exported;
- [no]filled**: include, or explicitly exclude the databases marked “filled” by the clustering system;
by default, all the databases are selected;
- groups groups-list**: restrict the export to the databases defined in the specified groups (see *Format of the Lists* page 2-13);
- help**: description of the available options;
- include items-list**: alias for **-items** (see *Format of the Lists* page 2-13);
- items items-list**: select only the databases containing the types of data specified (see *Format of the Lists* page 2-13);
by default, all the type data are selected;
- local**: select only the databases stored on a disk of the computer where the command is running;
by default, all the databases are selected;
- maxsize MB**: restrict the export to the databases smaller than the given value, expressed in Megabytes;
by default, there is no restriction;
- minsize MB** : restrict the export to the databases larger than the given value, expressed in Megabytes;
by default, there is no restriction;
- missing**: select only the databases not present on the disks;

- nodes *nodes-list***: select only the databases stored on disks owned by the specified nodes;
- partial**: alias for **-nofilled**;
- since *date***: select only the databases modified since the given date (see *Format of the Dates* page 2-13);
- tdf**: display the information about the databases and the collections in a TDF form;
- verbose**: print additional information about the export operations;
by default, only a minimal amount of information are printed;

7.4 Description

BdbDistBrowser is a TCL script using the *BdbDistManager* extended commands. Its is used to find the databases associated to one or more domains. Miscellaneous criteria can be used to select databases: attributes of the files (date, size); logical organization of the data (components, group...)

The information is printed to the standard output in a TDF format. This format is is generally understandable only by Perl tools. If the **-tdf** option is used, a real TDF file is produced, and checksums are generated.

The selected collection databases are scanned and the attributes of the collections they contains are printed in TDF form.

7.5 See Also

BdbDistCenter, *BdbDmnExport*, *BdbDistMirror*, *collstagein*

BdbDistCollLoader

8 BdbDistCenter

8.1 Abstract

Export the databases of the Events Store based on file criteria, like date of modification

8.2 Usage

```
BdbDistCenter [options]
```

8.3 Options

- [no]archivesize *MB*: maximum size of the archives to create, in MB, or disable the creation of archives, if the –noarchive form is used;
the default value is 1800 (something smaller than 2 GB);
- before *date*: select only the databases modified before the specified date (see *Format of the Dates* page 2-13);
- bootfile *file*: boot file of the federation;
the default value is given by the environment variable \$OO_FD_BOOT;
- checkonly: just print a report describing the selected databases, but don't export anything;
- [no]collections: scan the collection databases during the scanning of the federation to be able to recreate the collections during the import operation, or skip this operation (to save time) if the -nocollections form is used;
- [no]compress: compress or not compress the created archives;
the default behavior is to compress the archives;
- configure *file.cfg*: private configuration file;
by default, the standard configuration file is read (see Chapter *Site Specific Customizations* page 23-1);
- delay *minutes*: time to wait for a read lock on the base;
by default, there is no delay;
- diskspace *MB*: maximum space available in the working directory for the export operation;
by default, the working directory is supposed large enough for the operation;
- exclude *components-list*: list of type of data to exclude from the export (see *Format of the Lists* page 2-13);
by default, all the types of data are exported;
- export *name*: name of the export – this name is used to build the name of Export Description File (TDF);

- the default name is the name of the first collection;
- factor rate**: estimation of the factor of compression of the data (see *Tape Transfer* page 2-4);
by default, a factor of 40 % is assumed;
 - [no]**filled**: include, or explicitly exclude the databases marked “filled” by the clustering system;
by default, all the databases are selected;
 - from bootfile**: alias for **-bootfile**;
 - groups groups-list**: restrict the export to the databases defined in the specified groups (see *Format of the Lists* page 2-13);
 - help**: description of the available options;
 - include components-list**: alias for **-items**;
 - items components-list**: selection of types of data to export (see *Format of the Lists* page 2-13);
by default, all the type data are exported;
 - labels label-list**: labels of the tapes to use to export the databases (see *Format of the Lists* page 2-13);
by default, the exported databases stay on disk and are not copied on tape;
 - lftp**: use FTP to copy the databases from the federation to the working directory;
by default, the files are copied using *ooattachdb* or the UNIX *cp* command, depending of the usage or note of the **-wait** option;
 - local name**: specify the name of a pre-define configuration – actually, only *slac* and *ccin2p3* are defined;
the default is based on the value of the environment variable **\$BFSITE**;
if the corresponding configuration does not exist, a default one is used (see Chapter *Site Specific Customizations* page 23-1)
 - [no]**log logfile-name**: define the name of the log file to create, or if the form **-nolog** is used, explicitly disable the creation of such a log file;
the default name is the name of the export (see **-export**) with the extension: *.log*;
 - maxsize MB**: restrict the export to the databases smaller than the given value, expressed in Megabytes;
by default, there is no restriction;
 - minsize MB**: restrict the export to the databases larger than the given value, expressed in Megabytes;
by default, there is no restriction;
 - nodes nodes-list**: restrict the export to the databases stored on the specified nodes;
 - partial**: alias for **-nofilled**;
 - [no]**recover recfile**: give the name of a recovery file to read at the beginning of the export operation, or if the form **-norecover** is used, explicitly disable the loading of such a recovery file;
by default, the recovery file is the name of the export (see **-export**) with the extension: *.rec*;
if such a file exists in the working directory, it is automatically loaded (see *Recovering from a Crash* page 2-6);
 - since date**: select only the databases modified since the given date (see *Format of the Dates* page 2-13);
 - summary**: print a summary of the export operation;
 - tapemodel name**: specify the model of tapes used to export the database – this option must be specified if the option **-labels** is used;

- use *input-tdf***: use an existing TDF file describing the candidate databases to export – the selection options can be used to reduce this list – the federation is not scanned;
by default, the databases are searched in the federation using ***BdbDistScan***;
- verbose**: print additional information about the export operations;
by default, only a minimal amount of information are printed;
the “verbose” information are automatically saved in the log file, even if the –**verbose** option is not used (but they don't appeared on the screen);
- [no]wait**: ask the program used to extract the database from the federation to proceed under the protection of a read-lock;
by default, the database is extracted using ***ooattachdb*** and is not locked;
- workdir *dir***: a directory to use as work area to copy the databases extracted from the federation, create the archives and compressed them (depending of the used options);
if the export doesn't use tapes, the files will stay in this directory at the end of the export, else, they will be copied on the tapes and removed from this directory;
the space on the disk must be large enough to receive the exported files, or the –**diskspace** option must be used to specify how many Megabytes can be used

8.4 Description

This section is missing

8.5 Restrictions

If the –**lftp** is used, the export operation must be executed on a computer where the *working directory* is mounted.

9 BdbDistCollLoader

9.1 Abstract

Define in the federation the collections contains in collection databases.

9.2 Usage

```
BdbDistCollLoader [option] dbsysname ...
```

9.3 Options

- checkonly**: just print a report describing the selected databases, but don't export anything;
- ignore**: ignore the errors and try to continue the import;
 - by default, the import tool will stop if an error is encountered;
- replace**: replace any existing collections by the new ones;
 - by default, the existing collections are not replaced;
- [no]tdf file**: give a name to the generated TDF file; or disable the creation of such file, if the **-notdf** form is used;
 - by default, a TDF file is created with the name of the first selected database;
- verbose**: print additional information about the export operations;
 - by default, only a minimal amount of information are printed;

9.4 Description

BdbDistCenter is a tool to restore the link between the persistent objects stored in a database, representing event collections, and the catalog of collections of the federation.

This tool can be used to recover situations where collection databases have been added to a federation but the collections have not been created.

10 BdbDistCopy

10.1 Abstract

Copy a database file from one place to another.

10.2 Usage

```
BdbDistCopy [options] from to
```

10.3 Options

- [no]checksize: enable or disable the control of the file of the imported database;
by default, the size of the files is checked after the copy;
- dbname: alias for -dbsysname;
- dbsysname: specify that the file to copy is designated by the system name of a database; this database is owned by the federation pointed by the value of the environment variable `$OO_FD_BOOT`;
- delay *mn*: time to wait for a read lock on the database to copy;
by default, there is no delay;
this option is valid only if a database is specified using -dbsysname;
- lftp: use FTP to copy the databases from the federation to the working directory;
by default, the files are copied the UNIX *cp* command;
- [no]lock: try to lock the database in read lock to be able to safely export it using UNIX *cp*;
by default, the database is exported is *oocopydb*;
this option is valid only if a database is specified using -dbsysname;
- [no]replace: allow *BdbDistCopy* to replace any existing file with the same name by the copied one;
by default, the program stops with an error message;
- verbose: print additional information about the export operations;
by default, only a minimal amount of information are printed;
- [no]wait: wait to obtain a read lock on the database to copy; if the -delay option is not specified, the program wait for ever;
by default, there is no delay;
this option is valid only if a database is specified using -dbsysname;

10.4 Description

BdbDistCopy is a script used by the export and import tools to copy the database files from a location to another. It can work with standard files, like any copy program, and with databases.

In file mode, the source file is just copied to the designated destination. In this case, the only interested of *BdbDistCopy* is it know how to talk with the PUD daemon to copy the file, or is able to establish a FTP transfer if the **-lftp** option is used.

In database mode, the source file is designated by the system name of the database. From this name, *BdbDistCopy* gets the name of the file, and the name of its disk server. In such a case, it is able to lock the database the time of the transfer to prevent an alteration of the database. The **-lock** option must be used. The copy itself is executed using the standard UNIX *cp* or using FTP, depending of the usage or not of the **-lftp** option.

The **-wait** option can be used to instruct the tool to wait up to get the lock. How many time the tool has to wait can be specified using **-delay** (the value specifies a number of minutes). Those both options allow the tool to wait until the processes currently modifying data in the databases finish their transactions.

If the **-lock** option is not used, the database is copied using the Objectivity program *oocopydb*. Its principal restriction is to not support any concurrent access to the database.

10.5 See Also

BdbCollExport, *BdbCollImport*, *BdbCollTransfer*, *BdbDistCenter*, *BdbDistMirror*, *BdbDmnExport*, *BdbDmnImport*, *BbdbImport*

11 BdbDistInstall

11.1 Abstract

Install a database in a federation

11.2 Usage

```
BdbDistInstall [options]
```

11.3 Options

- delay *mn***: time to wait for a lock on the database to install, if the database exists and has to be replaced;
by default, there is no delay;
- db *dbname***: system name of the database to install;
this option is *mandatory*;
- filepath *filepath***: complete path of the database file;
this option is *mandatory*;
- forced**: force the installation of the database by bypassing the Objectivity *ooattachdb* program, and all the checks it is doing;
- group *groupname***: name of a UNIX group;
- host *host***: host of the node serving the disk where the database file is copied;
this option is *mandatory*;
- id *dbid***: identification number of the database;
this option is *mandatory*;
- readonly**: turn the database file in read-only mode;
by default, the file is installed in write mode;
- verbose**: print additional information about the export operations;
by default, only a minimal amount of information are printed;
- [no]wait**: wait to obtain an update lock on the database to install, if this database has to be replaced;
by default, there is no delay;

11.4 Description

BdbDistInstall is a script used by the import tools to attach an imported database to the federation. The file must have been copied to its destination. *BdbDistInstall* will not copy it. This operation is usually done by *BdbDbImport* by using *BdbDistCopy*.

By default, the Objectivity program: *ooattachdb* is used. The main advantage of this program is to check the database before attaching it. It is able to detect corrupted databases, preventing the introduction of wrong data in the federation.

Unfortunately, the current version of Objectivity for Sun is not able to handle files larger than 2 Gigabytes. When such “big databases” are encountered, the installation is done using a tricky set of operations. In this mode, the database can't be controlled. Such situation also occurs when a new database replaces an existing one. To prevent conflicts with the users, the definition of the database in the catalog of the federation is changed under an update lock.

11.5 See Also

BdbCollImport, *BdbDmnImport*, *BdbDistMirror*, *BdbDbImport*

12 BdbDistManager

12.1 Abstract

TCL shell providing wrappers around the C++ classes used in the import/export operations

12.2 Usage

```
BdbDistManager [options] [command or script] [arguments]
```

12.3 Options

- execute**: execute rest of the line as a TCL program;
- readonly**: disable the update mode;
- timeout** *seconds*: PUD daemon default timeout;
- verbose**: print additional information about the export operations;
by default, only a minimal amount of information are printed;

12.4 Commands

A TCL command to immediately execute, when the –**execute** option is used.

12.5 Script

A TCL script to read and execute if the –**execute** option is not used.

12.6 Description

BdbDistManager is a TCL shell with some building commands wrapping the C++ classes used during the import and export operations. The low-level scripts used to handle the databases are based on this program.

As the TCL shells, if it is called without argument, *BdbDistManager* enters in a interactive mode and ask the user for the TCL instructions to execute. Unknown instructions are considered as UNIX commands and automatically send to the shell. Standard TCL extensions can be loaded using the **load** TCL instruction.

Arguments on the command lines are considered as the name of a script to execute and arguments to send to this script, as for any TCL shell, if the –**execute** option is not present at the beginning of the line.

In those both modes, there is no automatic connection to the Objectivity federation.

With the `-execute` option, the end of the command line is considered as a *BdbDistManager* instruction immediately executed. The result is printed on the user terminal. In this case, an connection to the federation is automatically through a read-only transaction.

An exit handler is installed to abort any open transaction before the program exits.

The supported commands added by *BdbDistManager* to TCL are described in the chapter *Extended TCL Commands*.

12.6.1 Examples

In this first example, *BdbDistManager* will execute the TCL script `checkdbs.tcl`.

```
% BdbDistManager checkdbs.tcl
```

In the second example, the *BdbDatabase* instruction creates a *wrapper* to access the database `evs_g_bulk_col0120AB`. This wrapper is called to check if the corresponding file is present on the disk. As the file exists, the *exist* subcommand returns “1”. This value is printed on the terminal.

```
% BdbDistManager -execute '[BdbDatabase new evs_g_bulk_col0120AB] exist'  
1
```

12.7 See Also

Chapter *BdbDistManager
Commands*

BdbDistCopy, *BdbDistInstall*, *BdbDistBrowser*, *BdbDistScan*, *BdbDistMirror*, *bdbstagein*, *collstagein*

13 BdbDistMirror

13.1 Abstract

Copy the databases from a production federation to an analysis one

13.2 Usage

```
BdbDistMirror [context] [options]
```

13.3 Context Options

- physics**: work with the OPR federation as source federation and the Physics Analysis one as target federation; the `OO_FD_BOOT` variable must point to the Physics Analysis boot file;
- simulation**: work with the SP1 production federation as source federation and the Simulation Analysis one as target federation; the `OO_FD_BOOT` variable must point to the Simulation Analysis boot file;

13.4 Options

- before date**: select only the databases modified before the specified date (see *Format of the Dates* page 2-13);
- checkonly**: just print a report describing the selected databases, but don't export anything;
- [no]collections**: scan the collection databases during the scanning of the federation to be able to recreate the collections during the import operation, or skip this operation (to save time) if the **-nocollections** form is used;
- [no]continue**: ignore the errors and try to continue the import, or immediately stop with an error message, if the form **-nocontinue** is used (this is the default);
- copy**: physically transfer the files to “mirror” to a disk of the destination federation – this is the default (see **-shadow**);
- delay mn**: time to wait for an update lock on the databases to replace;
by default, the tool waits during 15 minutes, then skip the database and try to continue with the next one;
- exclude items-list**: list of type of data to ignore (see *Format of the Lists* page 2-13);
- [no]filled**: include, or explicitly exclude the databases marked “filled” by the clustering system;
- groups groups-list**: restrict the export to the databases defined in the specified groups (see *Format of the Lists* page 2-13);
- include items-list**: types of data to select (see *Format of the Lists* page 2-13);
- items items-list**: alias for **-include**;

- local**: select only the databases stored on a disk served by the computer where the command is executed;
- missing**: select only the databases defined in the catalog of the source federation but not present on the disks (assuming those databases are available in the HPSS);
this option must be used only with the –**shadow** option;
- nodes *nodes-list***: select only the databases stored on disks served by the specified nodes (see *Format of the Lists* page 2-13);
- partial**: alias for –**nofilled**;
- replace**: replace in the target federation the files selected from the source one, even if they are up-to-date;
a target file is considered as up-to-date if its modification date is newer than the modification date of the source file;
by default, the up-to-date files are not replaced;
- shadow**: create an entry for the database in the catalog of the target federation, but don't copy the file – this option is intended to declare to Objectivity a database saved in the HPSS;
- since *date***: select only the databases modified since the given date (see *Format of the Dates* page 2-13);
- [no]stop**: stop the operations if an error is encountered, or try to continue (if the –**nostop** form is used);
this option has the same behavior than the –**continue** one, with the opposite convention;
- summary**: print a summary of the databases and collections to transfer;
- [no]tdf *tdf-file***: give an explicit name of the TDF file generated at the end of the operations, or disable the creation of such a file if the –**notdf** form is used;
by default, a TDF file is created and its name is built from the current date (*yearmonthday.tdf*);
- to *host***: name of the host owning the disks where the files have to be copied;
by default, the files are copied on the disks of the target federation server;
- use *tdf-file***: read the list of candidate files to transfer from the indicated TDF file – this option avoids scanning the source federation to find the files to copy;
Caution: the filter options don't work in this mode (–**include**, –**exclude**, –**groups**, ...).
- verbose**: print additional information about the export operations;
by default, only a minimal amount of information are printed;
- [no]wait**: wait for an update lock on the databases to replace;
by default, the tool waits during 15 minutes, then skip the database and try to continue with the next one;

13.5 Description

This section is missing

13.6 Restriction

Actually, this tool is specific to the SLAC configuration.

14 BdbDistScan

14.1 Abstract

Print attributes of the databases associated to a list of collections.

14.2 Usage

```
BdbDistScan [options] collection-treepath...
```

14.3 Options

- components *comp-list***: alias for **-include**;
- exclude *comp-list***: list of type of databases to ignore (see *Format of the Lists* page 2-13);
by default, no databases are ignored;
- include *comp-list***: selection of types of databases to report (see *Format of the Lists* page 2-13);
by default, all the databases are reported;
- full**: display the information about the selected databases in a TDF form – this option excludes the **-summary** one;
by default, only the name and the id. of the databases are printed;
- help**: description of the available options;
- invalidatemd**: mark the metadata objects of the selected collections as invalid. *No collection will be scanned.*
- printmd**: dump the content of metadata objects associated with each collection. *No collection will be scanned.*
- since *date***: select only the databases modified since the given date (see *Format of the Dates* page 2-13);
- summary**: signal if the database is present on the disks, is not present on a disk, or is not defined in the federation catalog – this option excludes the **-full** option;
- [no]trustmd**: scan the database using the metadata informations associated with the collections; this greatly speeds up this operation.
by default, metadata are not used;
- [no]updatemd**: update invalid metadata informations during the scan. If used in conjunction with **-trustmd**, collections with valid metadata will not be updated. Otherwise, each collection will be completely checked, and each metadata object will be updated, regardless of the validity status.
by default, metadata are not updated;
- verbose**: print additional information about the export operations;

by default, only a minimal amount of information are printed;

14.4 Description

BdbDistScan is a script used by the export tools to search for the databases containing the data pointed by collections. It iterates over all the events of the specified collections, searching the databases where they are stored.

The information printed about the selected database is depending of the specified options. By default, only the system name and the identification number of the databases are displayed. With the **-full** option, *BdbDistScan* prints other database attributes requested by the export tools. The TDF format is used. The **-summary** option is useful to check if the files are present on the disks or if they have been migrated in the HPSS.

14.4.1 Disclaims

BdbDistScan has to traverse the entire data tree of the events contains in the specified collections. This operation can take time for large collection. The performances can worse if the federation is heavily used and the data are scanned through the AMS server. The situation may be improved by taking advantage of possible valid metadata, using the **-trustmd** and **-updatemd** flags.

14.5 See Also

BdbCollExport, *BdbCollTransfer*

15 BdbDistSummary

15.1 Abstract

Prints a report on the databases and tapes selected by an export or an import.

15.2 Usage

```
BdbDistSummary [options] [-collections] collections ...
```

```
BdbDistSummary [options] [-domains] domains ...
```

```
BdbDistSummary [options] [-tdf] export-file ...
```

15.3 Context Options

- collections**: indicate that the arguments are the names of collections;
- domains**: indicate that the arguments are the names of domains;
- tdf**: indicate that the argument is a TDF file;

15.3.1 Recognized Contexts

If the argument is the name of a file and if the extension of this file is **tdf**, the TDF context is assumed, else, if no *context option* is used, the collection context is assumed.

15.4 Options

- archivesize** *MB*: maximum size of the archives to create, in MB, or disable the creation of archives, if the –**noarchive** form is used;
the default value is 1800 (something smaller than 2 GB);
- bootfile** *file*: boot file of the federation;
the default value is given by the environment variable `$OO_FD_BOOT`;
- [no]compress**: compress or not compress the created archives;
the default behavior is to compress the archives;
- configure** *file.cfg*: private configuration file;

- by default, the standard configuration file is read (see Chapter *Site Specific Customizations* page 23-1);
- diskspace MB**: maximum space available in the working directory for the export operation;
 - by default, the working directory is supposed large enough for the operation;
- exclude components-list**: list of type of data to exclude from the export (see *Format of the Lists* page 2-13);
 - by default, all the types of data are exported;
- export name**: name of the export – this name is used to build the name of Export Description File (TDF);
 - the default name is the name of the first collection or the name of the first domain;
- factor rate**: estimation of the factor of compression of the data (see *Tape Transfer* page 2-4);
 - by default, a factor of 40 % is assumed;
- from bootfile**: alias for –**bootfile**;
- help**: description of the available options;
- include components-list**: alias for –**items**;
- items components-list**: selection of types of data to export (see *Format of the Lists* page 2-13);
 - by default, all the type data are exported;
- labels label-list**: labels of the tapes to use to export the databases (see *Format of the Lists* page 2-13);
 - by default, the exported databases stay on disk and are not copied on tape;
- tapemodel name**: specify the model of tapes used to export the database – this option must be specified if the option –**labels** is used;
- verbose**: print additional information about the export operations;
 - by default, only a minimal amount of information are printed;

15.5 Description

BdbDistSummary is useful to estimate the size of an export: how many databases are selected, how many Mega or Gigabytes will be copied, how many tapes are needed... It is also useful to check an existing TDF file and know what disk space will be necessary for the import, the name of the tapes, and soon. It accepts most of the options of the export and import tools.

BdbDistSummary is able to recognize a TDF file if its extension is "**tdf**". In the other case, collections are assumed. The –**collections**, –**domains** and –**tdf** options can be used to specify how the arguments must be interpreted.

15.6 See Also

BdbCollExport, *BdbDmnExport*, *BdbCollImport*, *BdbDmnImport*, *BdbCollTransfer*, *BdbDistCenter*, *BdbDistScan*, *BdbDistBrowser*

16 BdbDistTdfFilter

16.1 Abstract

Create a new TDF file by applying selection criteria to an existing one.

16.2 Usage

```
BdbDistTdfFilter [options] input-tdf
```

16.3 Options

- before *date*: select only the databases modified before the date
- checkonly: print a report - don't move the databases
- collectiononly: keep only the definitions of the collections
- exclude *components-list*: exclude the specified components
- groups *groups-list*: select only the specified groups
- help: description of the available options
- ignore: ignore the TDF checksums
- include *components-list*: alias for "-items"
- items *components-list*: export only the specified components
- maxsize *MB*: maximum size of the selected databases
- minsize *MB*: minimum size of the selected databases
- nocollections: remove the definitions of the collections
- since <date>: select only the databases modified since the date
- summary: print a summary instead of the filter TDF
- verbose: verbose mode - maximum of traces

16.4 Description

The section is missing...

17 BdbDistWriteTape

17.1 Abstract

Write to tapes databases previously exported to disk.

17.2 Usage

```
BdbWriteTape [-tdf] tdf-file -tapemodel model -labels labels
```

17.3 Options

- export** *export-name*: name of the export, used to name the new TDF file;
- help**: print the available options;
- label** *label-list*: list of the tapes to use;
- tapemodel** *model*: model of tape to use;
- tdf** *input-tdf-file*: input TDF file describing the files to write on tape;

17.4 Description

This section is missing...

18 BdbDmnExport

18.1 Abstract

Export the databases of a list of domains

18.2 Usage

```
BdbDmnExport [options] domain...
```

18.3 Options

- [no]archivesize **MB**: maximum size of the archives to create, in MB, or disable the creation of archives, if the –noarchive form is used;
the default value is 1800 (something smaller than 2 GB);
- before **date**: select only the databases modified before the specified date (see *Format of the Dates* page 2-13);
- bootfile **file**: name of the boot file of the federation;
the default value is given by the environment variable **\$OO_FD_BOOT**;
- checkonly: just print a report describing the selected databases, but don't export anything;
- [no]compress: compress or not compress the created archives;
the default behavior is to compress the archives;
- configure **file.cfg**: private configuration file;
by default, the standard configuration file is read (see *Chapter Site Specific Customizations* page 23-1);
- delay **minutes**: time to wait for a read lock on the base;
by default, there is no delay;
- diskspace **MB**: maximum space available in the working directory for the export operation;
by default, the working directory is supposed large enough for the operation;
- exclude **components-list**: list of type of data to exclude from the export (see *Format of the Lists* page 2-13);
by default, all the types of data are exported;
- export **name**: name of the export – this name is used to build the name of Export Description File (TDF);
the default name is the name of the first collection;
- factor **rate**: estimation of the factor of compression of the data (see *Tape Transfer* page 2-4);
by default, a factor of 40 % is assumed;

- from bootfile:** alias for **-bootfile**;
- help:** print the available options;
- include components-list:** alias for **-items** (see *Format of the Lists* page 2-13);
- items components-list:** selection of types of data to export (see *Format of the Lists* page 2-13);
by default, all the type data are exported;
- labels label-list:** labels of the tapes to use to export the databases (see *Format of the Lists* page 2-13);
by default, the exported databases stay on disk and are not copied on tape
- lftp:** use FTP to copy the databases from the federation to the working directory;
by default, the files are copied using *ooattachdb* or the UNIX *cp* command, depending of the usage or note of the **-wait** option;
- local name:** specify the name of a pre-define configuration – actually, only *slac* and *ccin2p3* are defined;
the default is based on the value of the environment variable **\$BFSITE**;
if the corresponding configuration does not exist, a default one is used (see Chapter *Site Specific Customizations* page 23-1)
- [no]log logfile-name:** define the name of the log file to create, or if the form **-nolog** is used, explicitly disable the creation of such a log file;
the default name is the name of the export (see **-export**) with the extension: *.log*
- maxsize MB:** restrict the export to the databases smaller than the given value, expressed in Megabytes;
by default, there is no restriction;
- minsize MB:** restrict the export to the databases larger than the given value, expressed in Megabytes;
by default, there is no restriction;
- [no]recover recfile:** give the name of a recovery file to read at the beginning of the export operation, or if the form **-norecover** is used, explicitly disable the loading of such a recovery file;
by default, the recovery file is the name of the export export (see **-export**) with the extension: *.rec*;
if such a file exists in the working directory, it is automatically loaded (see *Recovering from a Crash* page 2-6);
- since date:** select only the databases modified since the given date (see *Format of the Dates* page 2-13);
- summary:** print a summary of the export operation;
- tapemodel name:** specify the model of tapes used to export the database – this option must be specified if the option **-labels** is used;
- use input-tdf:** use an existing TDF file describing the candidate databases to export – the selection options can be used to reduce this list – the federation is not scanned;
by default, the databases are searched in the federation using *BdbDistScan*;
- verbose:** print additional information about the export operations;
by default, only a minimal amount of information are printed;
the “verbose” information are automatically saved in the log file, even if the **-verbose** option is not used (but they don't appeared on the screen);
- [no]wait:** ask the program used to extract the database from the federation to proceed under the protection of a read-lock;
by default, the database is extracted using *ooattachdb* and is not locked;

-workdir dir: name of a directory to use as work area to copy the databases extracted from the federation, create the archives and compressed them (depending of the used options);
if the export doesn't use tapes, the files will stay in this directory at the end of the export, else, they will be copied on the tapes and removed from this directory;
the space on the disk must be large enough to receive the exported files, or the **-diskspace** option must be used to specify how many Megabytes can be used

18.4 Description

This section is missing

18.5 Restrictions

-lftp must be used on the computer where the working directory is mounted.

19 BdbDmnImport

19.1 Abstract

Import domain databases exported using *BdbDmnExport*

19.2 Usage

```
BdbDmnImport [options] description-file
```

19.3 Options

- bootfile file**: name of the boot file of the federation;
the default value is given by the environment variable `$OO_FD_BOOT`;
- checkonly**: just print a report describing the selected databases, but don't export anything;
- continue**: ignore the errors and try to continue the import;
by default, the import tool will stop if an error is encountered;
- delay minutes**: time to wait for an update lock on the base to replace, or for the collection databases;
by default, there is no delay;
- exclude components-list**: list of type of data to ignore during the import (see *Format of the Lists* page 2-13);
by default, all the types of data are imported;
- help**: description of the available options;
- include components-list**: alias for **–items**;
- items components-list**: restrict the import to the types of data specified (see *Format of the Lists* page 2-13);
by default, all the data are imported;
- lftp**: use FTP to copy the databases on disks of the federation;
by default, the files are copied the UNIX *cp* command (see *Restrictions* page 19-34);
- local name**: specify the name of a pre-define configuration – actually, only `slac` and `ccin2p3` are defined;
the default is based on the value of the environment variable `$BFSITE`;
if the corresponding configuration does not exist, a default one is used (see Chapter *Site Specific Customizations* page 23-1)
- [no]log logfile-name**: define the name of the log file to create, or if the form **–nolog** is used, explicitly disable the creation of such a log file;
the default name is the name of the export (see **–export**) with the extension: *.log*

- nostaging:** don't use the staging system to access the files and search them in the working directory
- [no]recover *recfile*:** give the name of a recovery file to read at the beginning of the export operation, or if the form **-norecover** is used, explicitly disable the loading of such a recovery file;
 - by default, the recovery file is the name of the export export (see **-export**) with the extension: **.rec**;
 - if such a file exists in the working directory, it is automatically loaded (see *Recovering from a Crash* page 2-6);
- replace:** replace any existing databases by the imported ones;
 - by default, the existing databases are not replaced and an error is signaled;
- stagesize *MB*:** maximum space available in the disks used to stage-in the databases to import;
 - by default, the staging system is supposed to have enough space for the operation;
- to *bootfile*:** alias for **-bootfile**;
- verbose:** print additional information about the export operations;
 - by default, only a minimal amount of information are printed;
 - the “verbose” information are automatically saved in the log file, even if the **-verbose** option is not used (but they don't appeared on the screen);
- [no]wait:** wait for an update lock before trying to import a database or to attach a collection;
 - by default, the import program does not wait;
- workdir *dir*:** name of a directory to use as work area to uncompress and expand the archives;
 - the imported files are automatically removed at the end of the import;

19.4 Description

This section is missing

19.5 Restrictions

- lftp** must be used on the computer where the working directory is mounted.

20 bdbstagein

20.1 Abstract

Download from HPSS tapes databases designed by their system names

20.2 Usage

```
bdbstagein [options] database ...
```

20.3 Options

- checkonly**: just print a report describing the selected databases, but don't export anything;
- [no]**message** *text*: give the text of title to give to the e-mail sent at the end of the stage-in – if the form –**nomessage** is used, no e-mail are sent;
the default is to send a message defined by the stage-in system;
- read**: turn the staged files in read only mode (this is the default);
- summary**: print a report on the files to stage;
- time** *mn*: interval of time between to check operation, in minutes; the default value is 15 mn, if the –**wait** option is used;
this value can't be smaller than 5 minutes;
- verbose**: print additional information about the export operations;
by default, only a minimal amount of information are printed;
- [no]**wait**: wait for the end of the stage-in operation, by periodically checking if the files are available;
by default, the tool doesn't wait;
- write**: turn the files in write mode; the user must be allowed to manage the files to use this option (ie., he or she must have the system privileges for the corresponding files);
by default, the files are turned in read-only mode by the stage-in system;

20.4 Description

bdbstagein is a script to download from HPSS tapes the databases defined in the federation catalog but not present on disk. Such a situation can occurred when the database files have been purged by the HPSS system. It allows the manager of the federation to easily restore those files.

20.4.1 Disclaims

*Caution: **bdbstagein** is closely related to the HPSS support software used at SLAC.*

20.5 See Also

collstagein

21 collstagein

21.1 Abstract

Download from HPSS tapes databases associated to a list of collections which are not on disk

21.2 Usage

```
collstagein [options] collection ...
```

21.3 Options

- checkonly**: just print a report describing the selected databases, but don't export anything;
- exclude *components***: explicitly rejected those type of databases from the list of databases to stage-in (see *Format of the Lists* page 2-13);
by default, all the types of databases are selected;
- include *components***: select only of those types of databases to stage-in (see *Format of the Lists* page 2-13);
by default, all the type databases are selected;
- items *components***: alias for –**include**;
- [no]message *text***: give the text of title to give to the e-mail sent at the end of the stage-in – if the form –**nomessage** is used, no e-mail are sent;
the default is to send a message defined by the stage-in system;
- read**: turn the staged files in read only mode (this is the default);
- summary**: print a report on the files to stage;
- time *mn***: interval of time between to check operation, in minutes; the default value is 15 mn, if the –**wait** option is used;
this value can't be smaller than 5 minutes;
- verbose**: print additional information about the export operations;
by default, only a minimal amount of information are printed;
- [no]wait**: wait for the end of the stage-in operation, by periodically checking if the files are available;
by default, the tool doesn't wait;
- write**: turn the files in write mode; the user must be allowed to manage the files to use this option (ie., he or she must have the system privileges for the corresponding files);
by default, the files are turned in read-only mode by the stage-in system;

21.4 Description

collstagein is a script to download from HPSS tapes the databases related to collections but not present on disk. Such a situation can occur when the database files have been purged by the HPSS system. It allows a user to restore those files, for example before starting a job.

21.4.1 Disclaims

Caution: collstagein is closely related to the HPSS support software used at SLAC.

21.5 See Also

bdbstagein, BdbDistScan

22 BdbDistManager Commands

22.1 Overview

22.1.1 Class Methods

This section is missing...

22.1.2 Object Methods

This section is missing...

22.2 BdbDistManager Built-in Commands

22.2.1 BdbApplication

Parent	BdbObject
Description	Manage the transaction services and the application set-up.

Functions

abort	BdbApplication abort <i>Class method</i> Abort the current transaction and cancel the modifications made since the last <i>start</i> .
bootfile	BdbApplication bootfile <i>Class method</i> Return the name of the boot file used to access the federation.
commit	BdbApplication commit <i>Class method</i>

Close the current transaction and save the modifications.

- host** `BdbApplication host`
Class method
Return the name of the host where the catalog of the federation is stored.
- mode** `BdbApplication mode -access`
`BdbApplication mode -verbose`
`BdbApplication mode -wait`
Class method
-**access**: return the access mode to the data (the possible returned values are **none**: no transaction open, **read**: read only transaction, and **update**: update transaction);
-**verbose**: return 1 or 0, depending if the application is set in verbose mode or not;
-**wait**: return 1 or 0, depending if the wait mode is enabled or not.
- nowait** `BdbApplication nowait`
Class method
Disable the wait mode. Any following requests to lock an object while returned immediately if it failed.
- quiet** `BdbApplication quiet`
Class method
Disable the verbose mode.
- root** `BdbApplication root`
Class method
Return the name of the directory where the federation catalog is stored.
- start** `BdbApplication start ?read|update?`
Class method
Start a new transaction. By default, the transaction is set in read-only mode (**read**). Using the option **update**, the transaction is initialized in update mode.
- update** `BdbApplication update`
Class method
Raise an existing read-only transaction to the update mode (if the transaction was already in update mode, nothing happens).

verbose `BdbApplication verbose ?mode?`
Class method
 Without argument, verbose mode is enabled (this is equivalent to give the value **1** as argument). With the **0** as argument, the verbose mode is disabled.

wait `BdbApplication wait ?delay?`
Class method
 Enable the wait mode: the following lock requests will wait until succeeded. If an argument is used, it specifies how many seconds to wait. Accordingly to the Objectivity documentation, this value must be upper than **0** and smaller than **14400**.
0 is equivalent to “no wait”. Another value is seen as an infinite wait.

22.2.2 BdbAuthorization

Parent `BdbObject`
Description `Manage the authorization services.`

Functions

check `BdbApplication check system`
 `BdbApplication check group groupname`
 `BdbApplication check user username`
Class method
 Check if the user is authorized for one of the different authorization level defined by the BaBar kernel, for the specified group or user name.
 Return **1** is the user is authorized, else **0**.

set `BdbApplication set system`
 `BdbApplication set group groupname`
 `BdbApplication set user username`
Class method
 Set the access mode to the specified authorization level. Raise a TCL error if the user is not authorized.

22.2.3 BdbCollection

Parent `BdbObject`
Description `Interface to the events collections.`

Functions

exist `collection exist ?name?`

Object method

Return **1** if the specified collection exists, else **0**.

If the collection exists, the object wrapper is set up to this collection object. Else, the existing definition of the wrapper is unchanged.

id

collection id

Object method

Return the object id. of the persistent object associated to the collection.

Metadata

collection metadata -print|-invalidate

Object method

Print or invalidate the metadata associated with the given collection. The dump of the metadata is a list, whose components are triplets (lists) with the following structure:

CollectionID validityFlag {DBID1 DBID2 ... DBIDn}

Where CollectionID is the ID of the collection in Objectivity format (DB-Cont-Page-Slot), validityFlag is 0 if the metadata are reported as being invalid (or missing altogether), and {DBID1 ... DBIDn} is the (possibly empty) list of DBIDs. An example is the following:

```
9218-4-1-2 1 { 9219 9220 9221 9222 }
```

```
9218-4-1-4 1 { 9219 9220 9221 9222 }
```

This means that the selected collection was indeed a tree collection, containing two vector collection 9218-4-1-2 and 9218-4-1-4. Both have valid metadata, and the DBIDs of the referenced files are 9219, 9220, 9221 and 9222.

name

collection name

Object method

Return the short name of the collection, ie.: the last part of the path, after the last separator (“/”).

new

BdbCollection new

BdbCollection new ?-name? name

BdbCollection new ?-id? id

Class method

Create a new collection wrapper. If an argument is specified, the wrapper is associated to the corresponding collection. This collection can be specified by its name, or by its object id.

Most of the time, the name and the object id. can be automatically recognized by the wrapper. The **-name** and **-id** options are usable to remove any ambiguity.

If the specified collection is not valid, an TCL error is raised.

path	<i>collection path</i> <i>Object method</i> Return the full path name of the collection.
set	<i>collection set ?-name? name</i> <i>collection set ?-id? id</i> <i>Object method</i> Initialize an existing wrapper with the collection specified by its name or its object id. Most of the time, the name and the object id. can be automatically recognized by the wrapper. The -name and -id options are usable to remove any ambiguity. If the specified collection is not valid, an TCL error is raised.
size	<i>collection size</i> <i>Object method</i> Return the number of events associated to the collection.
tdf	<i>collection tdf</i> <i>Object method</i> Print the attributes of the collection in a format compatible understandable by the export tools.

22.2.4 BdbCollectionInspector

Parent	BdbObject
Description	Iterate over the collection paths and return the name or object id. of the collections found.

Functions

clear	<i>inspector clear</i> <i>Object method</i> Clear the list of the collections collected during the last scan.
exist	<i>inspector exist path</i> <i>Object method</i> Check if the specified collection path is valid. Return 1 if the path is valid, or 0 .
next	<i>inspector next ?-name -id?</i> <i>Object method</i>

Return the next collection from the list created during the last scan. Return an empty string at the end of the list.

By default, the full path of the collection is returned. The **-id** option can be used to request the object id. of the collection.

new `BdbCollectionInspector new`

Class method

Create a new instance of the collection inspector wrapper.

reset `inspector reset`

Object method

Restart the list of the collections collected during the last scan to its beginning.

scan `inspector scan`

`inspector scan path`

`inspector scan -system`

`inspector scan -group group`

`inspector scan -user user`

`inspector scan -database dbname`

Object method

Start the scanning of the collections tree. The collections are received by repetitively calling **next**.

Without any argument, the entire collection tree is scanned.

The **-system**, **-group** or **-user** options can be used to restrict the scanning to the corresponding authorization level.

If a *path* is specified, only the collections with a name starting with this argument are selected.

If the **-database** option is used, it specifies the system name of a collection database. All the persistent collection objects found in this database are collected.

22.2.5 BdbDaemon

Parent `BdbObject`

Description `Interface to the PUD daemon.`

The daemon is identified by the name of the host where it is running. The default port number used is 3333. All the functions of this wrapper accept as optional argument a port number and a host name.

Functions

chread `daemon chread file ?-host host? ?-port port?`

- Object method*
Turn the specified file in read-only mode.
- chwrite** *daemon chwrite file ?-host host? ?-port port?*
Object method
Turn the specified file in read and write mode.
- copy** *daemon copy from to ?-host host? ?-port port?*
Object method
Copy the file specified by *from* to the location specified by *to*.
- erase** *daemon erase file ?-host host? ?-port port?*
Object method
Delete the specified file.
- exist** *daemon exist file ?-host host? ?-port port?*
Object method
Check if the file exists. Return **1** or **0**.
- mkdir** *daemon mkdir path ?-host host? ?-port port?*
Object method
Create the specified directory.
- msize** *daemon msize file ?-host host? ?-port port?*
Object method
Return the size of the file, in Megabytes.
- mtime** *daemon mtime file ?-host host? ?-port port?*
Object method
Return the date of the last modification of the file, in an integer number of seconds, following the UNIX convention.
- new** *BdbDaemon new ?host ?port??*
Class method
Create a new instance of the PUD daemon wrapper. 2 optional arguments can be specified: the name of the host where the daemon is running, and the IP port number where it is listening. The BaBar custom is to use the port 3333.

rename	<i>daemon rename from to ?-host host? ?-port port?</i> <i>Object method</i> Rename the file specified by <i>from</i> to the name and path specified by <i>to</i> .
set	<i>daemon set host port</i> <i>Object method</i> Change the host name and port number attributes of the wrapper instance.
size	<i>daemon size file ?-host host? ?-port port?</i> <i>Object method</i> Return the size of the specified file in Megabytes.
timeout	<i>BdbDaemon timeout ?seconds?</i> <i>Class method</i> Define how many time the wrapper will wait to receive an answer to its connection request.

22.2.6 BdbDatabase

Parent	BdbFile
Description	Interface to the services need to manage the database files. This wrapper inherits the properties of the <i>BdbFile</i> wrapper.

Functions

containers	<i>database containers</i> <i>Object method</i> Return the number of containers existing in the database.
dbexist	<i>database dbexist</i> <i>Object method</i> Check if the database associated to the wrapper exists. Return 1 or 0 .
dbid	<i>database dbid</i> <i>Object method</i> Return the identification number of the database, as a simple integer.
dbname	<i>database dbname</i>

Object method

Return the system name of the database.

isvalid

`database isvalid`

Object method

Check if the database is valid or not. Return **1** or **0**.

lock

`database lock ?read|update?`

Object method

Try to lock the database. Without argument, a read lock is requested. An update lock can be requested using the **update** argument.

How many time the application will wait for this lock is depending of the current setting (see for example ***BdbApplication wait***).

new

`BdbDatabase new`

`BdbDatabase new ?-name? dbname`

`BdbDatabase new ?-id? dbid`

Class method

Create a new instance of the database wrapper. A database can be associated to the new instance wrapper by specifying its system name or its database id.

Most of the time, the wrapper is able to automatically recognize if the argument is a system name or a number. The **-name** or **-id** options can be used to remove any ambiguity.

set

`database set ?-name? dbname`

`database set ?-id? dbid`

Object method

Associate a new database to the instance wrapper. The database can be identified by its system name or its number. Most of the time, the wrapper is able to automatically recognize if the argument is a system name or a number. The **-name** or **-id** options can be used to remove any ambiguity.

tdf

`database tdf`

Object method

Print the characteristics of the database using a format understandable by the export tools.

22.2.7 BdbDistribution

Parent

BdbObject

Description

Wrapper for the database distribution services.

Functions

attach	<code>BdbDistribution attach <i>filepath id</i></code> <code> ?-dbname <i>dbname?</i></code> <code> ?-host <i>host?</i> ?-port <i>port?</i></code> <code> ?-read?</code> <i>Class method</i>
autocommit	<code>BdbDistribution autocommit ?0 1?</code> <i>Class method</i>
change	<code>BdbDistribution change <i>dbname filepath</i></code> <code> ?-host <i>host?</i> ?-port <i>port?</i></code> <code> ?-read? ?-force?</code> <i>Class method</i>
collection	<code>BdbDistribution collection <i>pathname id</i> ?-force?</code> <i>Class method</i>
detach	<code>BdbDistribution detach <i>dbname</i></code> <i>Class method</i>
extract	<code>BdbDistribution extract <i>dbname outfile</i></code> <code> ?-host <i>outhost?</i> ?-port <i>port?</i></code> <code> ?-force?</code> <i>Class method</i>
flushFullDbs	<code>BdbDistribution flushFullDbs <i>dbname...</i></code> <i>Class method</i>
fullDbs	<code>BdbDistribution fullDbs ?-id?</code> <i>Class method</i>
recordDb	<code>BdbDistribution recordDb <i>dbname ?host filepath?</i></code> <i>Class method</i>
remove	<code>BdbDistribution remove <i>dbname</i> ?-force?</code> <i>Class method</i>

replace	BdbDistribution replace <i>dbname</i> <i>impfile</i> ?-host <i>host?</i> ?-port <i>port?</i> ?-read? <i>Class method</i>
shadow	BdbDistribution shadow <i>filepath</i> <i>id</i> ?-host <i>host?</i> ?-port <i>port?</i> <i>Class method</i>
stagein	BdbDistribution stagein <i>filepath...</i> -host <i>host</i> ?-port <i>port?</i> ?-[no]message <i>message?</i> ?-priority <i>level?</i> ?-read -write <i>Class method</i>
unrecordDb	BdbDistribution unrecordDb <i>dbname</i> <i>Class method</i>

22.2.8 BdbFSMgr

Parent	BdbObject
Description	Interface to the File System Manager.

Functions

exist	<i>fsmgr</i> exist <i>file</i> <i>Object method</i> Check if the specified file exists. Return 1 or 0 .
hread	<i>fsmgr</i> hread <i>file</i> <i>Object method</i> Turn the specified file in read-only mode.
hwrite	<i>fsmgr</i> hwrite <i>file</i> <i>Object method</i> Turn the specified file in write mode.

copy	<i>fsmgr copy from to</i> <i>Object method</i> Copy the file specified by <i>from</i> to the path specified by <i>to</i> .
erase	<i>fsmgr erase file</i> <i>Object method</i> Delete the specified file.
mkdir	<i>fsmgr mkdir path</i> <i>Object method</i> Create the directory specified by <i>path</i> .
msize	<i>fsmgr msize file</i> <i>Object method</i> Return the size of the file in Megabytes.
mtime	<i>fsmgr mtime file</i> <i>Object method</i> Return the date of last modification of the file in an integer number of seconds, following the UNIX convention.
new	BdbFSMgr new <i>Class method</i> Create a new instance of the wrapper.
rebuild	<i>fsmgr rebuild domain authorization usergroupname component</i> <i>Object method</i> Rebuild the access parameters for the specified database domain, authorization level, and component.
rename	<i>fsmgr rename from to</i> <i>Object method</i> Rename the file specified by <i>from</i> to the name and the path specified by <i>to</i> .
size	<i>fsmgr size file</i> <i>Object method</i>

Return the full size of the file, in bytes.

22.2.9 BdbFile

Parent BdbObject

Description Wrapper to parse the name of a file following the BaBar database name conventions, and access the characteristics of that file, locally or through the PUD daemon.

Functions

authorization *file* authorization

Object method

Return the authorization level name corresponding to the file.

hread *file* chread

Object method

Turn the file in read-only mode.

hwrite *file* chwrite

Object method

Turn the file in write mode.

copy *file* copy tofile

Object method

Copy the file described by the instance wrapper to the location specified by *tofile*.

daemon *file* daemon *?*-host *host?* *?*-port *port?* *?*-enable|-disable?

Object method

Explicitly define a PUD daemon to use to access the file associated to the wrapper instance, or disable any call to a daemon (assuming the file is locally accessible) if the **disable** option is used.

directory *file* directory

Object method

Return the name of the directory of the file.

domain *file* domain

Object method

Return the BaBar domain of the file.

exist	<i>file</i> exist <i>Object method</i> Check if the file exists. Return 1 or 0 .
filename	<i>file</i> filename <i>Object method</i> Return the name of the file. This name is the file part of the full path, after the last directory separator (“/”).
fullpath	<i>file</i> fullpath <i>Object method</i> Return the full path name of the file, including the name of the host, following the syntax: <i>host : /path /name .extention</i> .
host	<i>file</i> host <i>Object method</i> Return the name of the host serving the disk where the file is stored.
hostequal	<i>file</i> hostequal <i>host</i> <i>Object method</i> Check if the specified host is the same than the host of the file. The comparison is done using the long name form of both hosts, without case. Return 1 or 0 .
hostequal	BdbFile hostequal <i>host1 host2</i> <i>Class method</i> Check if the both hosts are equal. The comparison is done using the long name form of both hosts, without case. Return 1 or 0 .
localhost	BdbFile localhost <i>Class method</i> Return the long form name of the local host (the host where the application is running).
mkdir	<i>file</i> mkdir <i>Object method</i> Create the directory associated to the file.
msize	<i>file</i> msize

	<i>Object method</i>
	Return the size of the file in Megabytes.
mtime	<i>file mtime</i> <i>Object method</i>
	Return the date of the last modification of the file, in an integer number of seconds, following the UNIX convention.
name	<i>file name</i> <i>Object method</i>
	Return the short name of the file, without the directory part or its extension.
new	<code>BdbFile new ?filename ?-host host? -port port?</code> <i>Class method</i>
	Create a new instance of the wrapper, and associate it to the specified file name, if a <i>filename</i> argument is specified. The PUD daemon to use to manage the file can be forced using the -host and -port options.
	By default, the configuration file loaded in the federation is used to determine how to access the file.
number	<i>file number</i> <i>Object method</i>
	Return the number associated to the name of the database file, if any.
path	<i>file path</i> <i>Object method</i>
	Return the local name of the file, ie.: the directory name, the short name and the extension, but not the host name.
port	<i>file port</i> <i>Object method</i>
	Return the port number used to access the PUD daemon, if any.
remove	<i>file remove</i> <i>Object method</i>
	Delete the file associated to the wrapper.
rename	<i>file rename tofile</i>

Object method

Rename the file associated to the wrapper to the name and path specified by *tofile*.

Caution: *the current characteristics of the wrapper are not modified. A new wrapper has to be created, or the characteristics of the current instance have to be modified using the **set** service to access the new file.*

root	<i>file root</i> <i>Object method</i> Return the directory of <i>the File System</i> associated to the file, accordingly to the configuration definitions loaded in the federation.
set	<i>file set ?filename? ?-host host? ?-port port?</i> <i>Object method</i> Change the characteristics of the wrapper instance. A new file can be associated to the instance, or the characteristics of the PUD daemon to access the file can be modified, or both.
size	<i>file size</i> <i>Object method</i> Return the full size of the file, in bytes.
tdf	<i>file tdf</i> <i>Object method</i> Print the characteristics of the file in a format understandable by the export tools.

22.2.10 BdbLocateDb

Parent	BdbObject
Description	Search the databases associated to a collection or a domain and return them.

Functions

catalog	<i>locator catalog</i> <i>Object method</i> Scan the entire federation catalog.
clear	<i>locator clear</i> <i>Object method</i> Clear the list of databases collected during the last scan.

- collection** `locator collection ?-trustmd|-notrustmd? ?-updatemd|-noupdatemd? name...`
- Object method*
- Scan the specified collections, optionally updating the associated metadata. By default, metadata are not used nor updated.
- If the **-trustmd** flag is used, and the collection's metadata are reported as being valid, then they are used to get the file list instead of a full scan (thus saving time).
- If the **-updatemd** flag is used, then the collection is scanned completely, and the associated metadata are update. If you use this flag, make sure the current transaction is in update mode.
- Note: you can combine the **-trustmd** and **-updatemd** flags: if both are used, then only collections whose metadata are reported as being invalid (or non-existent) are completely scanned and their metadata are updated. So, use both flags if you want to scan the specified collection using the metadata, where available, and with a full scan + metadata update elsewhere.
- domain** `locator domain name...`
- Object method*
- Scan the specified domain.
- exclude** `locator exclude ?-event|-domain|-pattern? component...`
- Object method*
- Specify a list of patterns defining the databases to exclude from the list of the collected databases. The comparison is done on the database system name. The databases matching the specified exclude criteria are rejected. The include criteria, if any, are verified after the exclude ones.
- The **-event** option is used to specify the name of event components to exclude. Components currently defined in the BaBar Events Store are: *tru*, *sim*, *raw*, *rec*, *esd*, *aod* and *tag*.
- The **-domain** option is used to specify the name of a detector to exclude.
- The **-pattern** option can be used to specify any regular expression to match with the selected database names.
- get** `locator get`
- Object method*
- Return the entire list of the selected database names.
- include** `locator include ?-event|-domain|-pattern? component...`
- Object method*
- Specify a list of patterns restricting the list of the collected databases. The comparison is done on the database system name. Only the databases matching the specified include criteria are kept. The exclude criteria, if any, are verified first,

The **-event** option is used to specify the name of event components to exclude. Components currently defined in the BaBar Events Store are: *tru*, *sim*, *raw*, *rec*, *esd*, *aod* and *tag*.

The **-domain** option is used to specify the name of a detector to exclude.

The **-pattern** option can be used to specify any regular expression to match with the selected database names.

next	<code>locator next ?-database -tdf?</code> <i>Object method</i> Return the next database from the collected list. By default, the name of the database is returned. If the -tdf option is used, a list of the database characteristics is built by the wrapper and returned as a string. The used format is understandable by the export tools. Return an empty string at the end of the list of the collected databases. Caution: because multiple scans can be done with the same wrapper instance, a call to reset must be done before any call to next .
nextInvalid	<code>locator nextInvalid</code> <i>Object method</i> Return the next database seen as invalid during the scan operation. A database can be considered as invalid if it is not defined in the federation catalog at all, or if it is missing from the disk. Caution: because multiple scans can be done with the same wrapper instance, a call to reset must be done before any call to nextInvalid .
new	<code>BdbLocateDb new</code> <i>Class method</i> Create a new instance of the wrapper.
reset	<code>locator reset</code> <i>Object method</i> Set the iterator used to get the collected databases to the beginning of the list.
tdf	<code>locator tdf</code> <i>Object method</i> Return the characteristics of the collected databases as a list of TDF lines.

22.2.11 BdbMessage

Parent BdbObject

Description Output messages to the terminal and logfiles.

Functions

message `BdbMessage message <strings>`

Class method

Output message string.

prefix `BdbMessage message ?<mode>?`

Class method

Turn on or off prefix mode. In prefix mode, the any output to stdout is prefixed with `MESSAGE` and enclosed in single quotes. This allows the calling Perl script to identify it.

verbose `BdbMessage verbose <strings>`

Class method

Output verbose message string. Not displayed unless `BdbApplication verbose` is set

warning `BdbMessage message <strings>`

Class method

Output warning/error message string. Output also goes to error logfile.

22.3 Library facilities

The instructions describe in this section are implemented as TCL procedures. They are automatically loaded using the TCL auto-load mechanism.

abort `abort message`

module bdbDistLibrary.tcl

Stop the application with an error message.

dump `dump variable...`

module bdbDistLibrary.tcl

Print to the user terminal the value of the specified TCL variables.

getPassword `getPassword ?user ""? ?host "remote"?`

module bdbDistFtp.tcl

Ask the user for its password.

If the environment variable `BDBFTP_PASSWORD` is defined, its value is returned and the user is not asked. The procedure does not define this value.

The *user* and *host* arguments are used to build the prompt. If *user* is not specified, the name of the user is searched using *whoami*.

help	<code>help</code> <i>module bdbDistLibrary.tcl</i> Print on the user terminal the syntax of the known <i>BdbDistManager</i> extensions.
message	<code>message message</code> <i>module bdbDistLibrary.tcl</i> Output a message to the terminal and the logfile.
severe	<code>severe stop message</code> <i>module bdbDistLibrary.tcl</i> Output warning/error message string and stop if <code>stop=1</code> .
verbose	<code>verbose message</code> <i>module bdbDistLibrary.tcl</i> Output a message to the logfile and, if the application is running in verbose mode, the terminal.
warning	<code>warning message</code> <i>module bdbDistLibrary.tcl</i> Output warning/error message string. Output also goes to error logfile.

22.4 General Procedures

bdbCollectionScan	<code>bdbCollectionScan dbsysname...</code> <i>module bdbDistCollScan.tcl</i> Print in TDF format the description of the collections found in collection databases specified as argument. Caution: to reduce the number of locks, the transaction is automatically committed and restarted in read-only mode after each 64 collections found.
bdbDbInstall	<code>bdbDbInstall dbname dbid filepath dbhost readonly groupname forced</code> <i>module bdbDistDbInstall.tcl</i> Install a database in the federation.

The *groupname* option specifies the UNIX group of the AMS server and is used to change the group name of the file to this value. This feature is executed only if the program runs on a node mounting the disk where the file is stored.

The *forced* option forces the installation of the database by using catalog operations, bypassing the Objectivity *ooattachdb* tool. This argument can be used to force the installation of a database rejected by *ooattachdb*.

The other arguments are similar to the arguments of *ooattachdb*.

This procedure takes care of the “large files” (databases larger than 2 GB) and attaches those databases using catalog operations (to work around an *ooattachdb* limitation on Sun).

bdbDistFtp

`bdbDistFtp fromhost fromuser frompath tohost touser topath
module bdbDistFtp.tcl`

Copy a file using FTP.

If the environment variable **BDBFTP_PASSWORD** is defined, it contains the user password, else, the user is asked (see *getPassword*).

bdbDistSelect

`bdbDistSelect callerOptions
bdbDistSelect.tcl`

Extended databases scanner.

The selection criteria are specified using a TCL associative array. The recognized keys are the following (those keys are *case sensitive*):

- **INCLUDE**: databases components to accept;
- **EXCLUDE**: database component to reject;
- **NODES**: nodes serving the disks where databases to accept are stored;
- **ROOTS**: root directories of the databases to accept;
- **GROUPS**: names of the groups of the databases to accept;
- **MINSIZE**: minimal size (in MB) of the databases to accept;
- **MAXSIZE**: maximal size (in MB) of the databases to accept;
- **SINCE**: the databases are accepted if they have a date of modification following this date (specified as a valid TCL time or an UNIX number of seconds);
- **BEFORE**: the databases are accepted if they have been modified before this date (specified as a valid TCL time or an UNIX number of seconds);
- **FILLED**: only the databases marked entirely filled/not filled in the Clustering registry (0/1);
- **MISSING**: only the databases missing from the disks (0/1).

If none of **FILLED** or **MISSING** keys are specified, no **MISSING** is assumed.

If the databases are not on disks, the date and size criteria are not used.

The databases must satisfy all the specified criteria to be selected.

Example:

```
set options(FILLED) 1
set options(INCLUDE) "raw rec"
while {[set dbwrp [distEventsScanner options]] != ""} {
    puts "DB Name: [${dbwrp name}]"
}
```

Reserved Keys:

The following keys are internally used by *bdbDistSelect*:

- **dbWrp:** instance of BdbDatabase used to check the database attributes;
- **filledList:** list of database ids received from the Clustering System;
- **filledIdx:** current position in the *filledList* list;
- **locator:** instance of BdbLocateDb used to scan the federation.

23 Site Specific Customizations

23.1 Overview

The support of the local tape system uses the 3 functions *stagein*, *stageout* and *stageclear*. Those functions are loaded from a Perl module named **bdbDistLocal.pm**. This module is searched in the subdirectory **site/\$BFSITE/** of the distribution package (where *BFSITE* is the environment variable defined for each site, following the BaBar convention)

Currently, 3 sites are supported: SLAC, CCIN2P3 and RAL. Their *bdbDistLocal* modules can be used as example to write the support for other tape systems. The local BaBar correspondent has also to enter this new subdirectory and the corresponding Perl module in CVS, then e-mail the description of the change to the package coordinator (David R. Quarrie: quarrie@SLAC.Stanford.edu).

If such a module does not exist for the site, a default version is loaded from the package directory. This default version does not do anything very useful, and is mostly for debug tests.

*Note: the access to the tapes can be disabled using the **-nostaging** option of the import tools. In such a case, the files must be downloaded from the tapes by the user and copied to the directory where the TDF file is located (or symbolic links can be created in the directory to those files).*

23.2 Tape Support Functions

The 3 functions to provide to access the tapes are the following.

23.2.1 *stagein*

Read files from a tape. The files are designated by their names, their position on the tapes and their size.

The list of positions can be converted to something understandable by a staging system using the function **bdbDistLib::indexSequence** (see for example **site/slac/bdbDistLocal.pm**).

Arguments

- **label**: label of the tape to read;
- **model**: type of tape (Redwood, DLTIV, DLTIII);
- **positions**: position of the files on the tapes, counted from 1 (“:” separated list);
- **filelist**: list of the files to read (space separated);
- **filesize**: size of the files, rounded up to an integer number of MB (“,” separated list).

23.2.2 stageout

Write files on a tape. The files are written sequentially from the last written files.

Caution: the first write starts at the beginning of the tape, without checking if something was already saved on the tape.

Arguments

- **label:** label of the tape to write;
- **model:** type of tape (Redwood, DLTIV, DLTIII);
- **files:** list of the files to write (space separated);
- **nbfiles:** number of files to write;
- **position:** position on the tape of the first file to write – the files are written sequentially. The start of the tape is position 1.

23.2.3 stageclear

Remove the staged-in files

Arguments

- **filelist:** list of the files to remove (space separated list).

24 Transfer Description Format

24.1 Overview

The collections, databases, archives and tapes exported by the Distribution tools are described in a TDF file (Transfer Description File). All the information needed by the import tools are contained in this file.

Each line describes a particular element. This description is a sequence of keywords followed by a corresponding value. The first keyword of each line (primary key) identifies the nature of the element ³.

The most useful keywords are the following:

- **DATABASE**: describes a database;
- **ARCHIVE**: describes an archive;
- **LABEL**: describes a tape;
- **COLLECTION**: describes a collection;
- **DBS**: simple checksum - number of DATABASE lines in the TDF;
- **ARCHS**: simple checksum - number of ARCHIVE lines in the TDF;
- **TAPES**: simple checksum - number of LABEL lines in the TDF;
- **COLLS**: simple checksum - number of COLLECTION lines in the TDF;
- **EXPORT**: describes the export.

Notes:

- The term “archive” is used to specify “an individual file” of the export. This file can be a tar file or a single database. It can be compressed or not. Its name is based on the export name, followed by a number. Its extension is `tar` for a tar file and `xdb` for a single database.
- If the file is compressed, the `gz` extension is added at the end of its name.
- Even if a database is exported as a single file, there is a **DATABASE** entry for the database **and** an **ARCHIVE** entry for the file.
- The **LABEL** and **TAPES** keywords can be missing if the exported files are not saved on tape.
- The **COLLECTION** and **COLLS** keywords can be missing for a non Events Store domain export, an incremental export without collection databases, or if the definition the collections has been explicitly disabled.

³ Such a file is easy to convert to associative arrays using a Perl function.

24.2 Simplified Example

- LABEL ZZ0001 ARCHS 0:1:2 MODEL DLTIV SIZE 7952.796875 NB 0

Description of the tape **ZZ0001**.

<u>Keys</u>	<u>Value</u>
LABEL	the label of the tape
MODEL	the model of the tape (here, a DLT 4)
SIZE	amount of data on the tape, in MB <i>Note: this value can be larger than the normal size of the tape, depending on the compression factor option specified to take care of the hardware compressor efficiency.</i>
ARCHS	index of the archives written of that tape
NB	index of the tape

- ARCHIVE hadron-1.tar.gz DBS 0:1:2:3:4:8 FORMAT tar SIZE 2943.421875 COMPRESS 1 NB 0

Description of the first archive.

The index of this archive is seen in the field ARCHS of the previous LABEL entry.

<u>Keys</u>	<u>Value</u>
ARCHIVE	the name of the file
FORMAT	type of the file (here, a tar file)
SIZE	size of the file (after <i>gzipping</i> , if the compress option was used)
COMPRESS	boolean value indicating if the file has been compressed using <i>gzip</i> (like in this example)
DBS	index of the databases packed in the archive <i>Note: even if the file is a single database ie, is not a tar file, there is a different DATABASE entry for it, and this field contains its index</i>
NB	index of the archive, as seen in the LABEL . ARCHS field

- DATABASE evs_g_isPhysicsEvents_col008019 FILE evs_g_isPhysicsEvents_col008019.bdb DBID 32793 SIZE 81.359375 HOST datamove2.slac.stanford.edu TIME 933226464 NB 0

Description of the first exported database (simplified)

<u>Keys</u>	<u>Value</u>
DATABASE	system name of the database, as it appears in the Objectivity catalog
FILE	name of the database file (by convention, the system name followed by .bdb)
DBID	the Database Identification number
SIZE	size of the file, in MB (before <i>gzipping</i> if the compress option was used)

HOST	name of the disk server where the file was stored (for information only)
TIME	date of the last modification of the file, as a number of seconds, following the UNIX convention
NB	index of the database, as seen in ARCHIVE.DBS

24.2.1 TDF Example

```
EXPORT hadron FILE /afs/slac.stanford.edu/u/ec/albert/BaBar/tst820/hadron.tdf SECONDS 933276920 DATE 29-Jul-1999
12:35:20 USER albert HOST datamove2.SLAC.Stanford.EDU
COLLECTION /groups/isPhysicsEvents/0000/5700/P8.1.10eV00fb/00005794/cb001/Skim8.1.10eV01/hadron EVENTS 248 COLLID 32
176-1-2 NB 0
LABEL ZZ0001 ARCHS 0:1:2 MODEL DLTIV SIZE 7952.796875 NB 0
ARCHIVE hadron-1.tar.gz DBS 0:1:2:3:4:8 FORMAT tar SIZE 2943.421875 COMPRESS 1 NB 0
ARCHIVE hadron-2.tar.gz DBS 5:6:7:9:12 FORMAT tar SIZE 2842.875 COMPRESS 1 NB 1
ARCHIVE hadron-3.tar.gz DBS 10:11 FORMAT tar SIZE 2166.5 COMPRESS 1 NB 2
DATABASE evs_g_isPhysicsEvents_col008019 FILE evs_g_isPhysicsEvents_col008019.bdb DBID 32793 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_evt00801A FILE evs_g_isPhysicsEvents_evt00801A.bdb DBID 32794 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_tag0049A5 FILE evs_g_isPhysicsEvents_tag0049A5.bdb DBID 18853 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_evshdr004A05 FILE evs_g_isPhysicsEvents_evshdr004A05.bdb DBID 18949 ROOT /objy/databa
DATABASE evs_g_isPhysicsEvents_rec004A3C FILE evs_g_isPhysicsEvents_rec004A3C.bdb DBID 19004 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_aod004A15 FILE evs_g_isPhysicsEvents_aod004A15.bdb DBID 18965 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_rawhdr0049DE FILE evs_g_isPhysicsEvents_rawhdr0049DE.bdb DBID 18910 ROOT /objy/databa
DATABASE evs_g_isPhysicsEvents_raw004A33 FILE evs_g_isPhysicsEvents_raw004A33.bdb DBID 18995 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_esd004A18 FILE evs_g_isPhysicsEvents_esd004A18.bdb DBID 18968 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_evt0049E0 FILE evs_g_isPhysicsEvents_evt0049E0.bdb DBID 18912 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_rec004A50 FILE evs_g_isPhysicsEvents_rec004A50.bdb DBID 19024 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_raw004A4E FILE evs_g_isPhysicsEvents_raw004A4E.bdb DBID 19022 ROOT /objy/databases/pr
DATABASE evs_g_isPhysicsEvents_tag004A53 FILE evs_g_isPhysicsEvents_tag004A53.bdb DBID 19027 ROOT /objy/databases/pr
TAPES 1
COLLS 1
ARCHS 3
DBS 13
```

24.3 Reading a TDF File

The **bdbDistLib.pm** module contains functions to handle TDF files. The **loadTdf** routine allows one to load a file to Perl associative lists. This function creates 4 Perl associative arrays, indexed by the entry number:

- **db**s: list of the databases;
- **archs**: list of the archives;
- **tapes**: list of the tapes;
- **colls**: list of the collections.

Example

A simple example to read a TDF, and iterates over it:

```
#!/usr/local/bin/perl5
# Description: demonstration of the TDF usage
# Usage: scantdf.pl <tdf-file>

# Hack to load packages from a relative path
$BDBDIST_PACKAGE = "BdbDistTools";
do "BdbDistTools/bdbDistLoader.pl";

# Load the bdbDistLib library
require "BdbDistTools/bdbDistLib.pm";

# Load the TDF file given as argument
```

```

if (! -e $ARGV[0]) {
    die "No such file";
}

&bdbDistLib::loadTdf($ARGV[0]);

# Iterate over the TDF info
for $t (0..$#tapes) {
    print "Tape: $tapes[$t]{LABEL}, size: $tapes[$t]{SIZE}\n";

    $pos = 1;
    for $a (split(":", $tapes[$t]{ARCHS})) {
        print "    Archive: $archs[$a]{ARCHIVE}, size: $archs[$a]{SIZE},",
            " tape location: $pos\n";
        $pos++;

        for $d (split(":", $archs[$d]{DBS})) {
            print "        Database: $dbs[$d]{DATABASE}, Id: $dbs[$d]{DBID}\n";
        }

        print "\n";
    }
}

```

And the result is:

```

% perl scantdf.pl hadron.rec
SLAC local parameters
Tape: ZZ0001, size: 7952.796875
  Archive: hadron-1.tar.gz, size: 2943.421875, tape location: 1
    Database: evs_g_isPhysicsEvents_col008019, Id: 32793
    Database: evs_g_isPhysicsEvents_evt00801A, Id: 32794
    Database: evs_g_isPhysicsEvents_tag0049A5, Id: 18853
    Database: evs_g_isPhysicsEvents_evshdr004A05, Id: 18949
    Database: evs_g_isPhysicsEvents_rec004A3C, Id: 19004
    Database: evs_g_isPhysicsEvents_esd004A18, Id: 18968

  Archive: hadron-2.tar.gz, size: 2842.875, tape location: 2
    Database: evs_g_isPhysicsEvents_col008019, Id: 32793
    Database: evs_g_isPhysicsEvents_evt00801A, Id: 32794
    Database: evs_g_isPhysicsEvents_tag0049A5, Id: 18853
    Database: evs_g_isPhysicsEvents_evshdr004A05, Id: 18949
    Database: evs_g_isPhysicsEvents_rec004A3C, Id: 19004
    Database: evs_g_isPhysicsEvents_esd004A18, Id: 18968

  Archive: hadron-3.tar.gz, size: 2166.5, tape location: 3
    Database: evs_g_isPhysicsEvents_col008019, Id: 32793
    Database: evs_g_isPhysicsEvents_evt00801A, Id: 32794
    Database: evs_g_isPhysicsEvents_tag0049A5, Id: 18853
    Database: evs_g_isPhysicsEvents_evshdr004A05, Id: 18949
    Database: evs_g_isPhysicsEvents_rec004A3C, Id: 19004
    Database: evs_g_isPhysicsEvents_esd004A18, Id: 18968

```

25 Known Problems and Traps

Crazy Size of the Export

> I have selected a 3-events collection, and I get Zigabytes of data.

The point here is to remember that an export is done at the **database** level.

The events of a collection are contained in databases. The export tool is only able to retrieve in which databases those events are stored, and copy them somewhere.

Consider the following example of a 3-events collection:

```
% BdbDistSummary -detail \  
  /groups/isPhysicsEvents/0000/6300/P8.1.10iV02fb/00006395/cb001/99-07-15.09:43:15-036-1  
  
SLAC local parameters  
Scanning collection /groups/isPhysicsEvents/0000/6300/P8.1.10iV02fb/00006395/cb001/99-07-  
15.09:43:15-036-1  
Organizing the export  
Export summary (estimation)  
Bootfile: /nfs/objyserv1/objy/databases/Production/physics/V1/production/analysis/BaBar.BOOT  
Number of collections: 1  
Pre-compression: yes  
Compression factor: 40 %  
Archive capacity: 1800 MB  
Number of archives: 3  
Number of databases: 10  
Data size: 7898.3 MB  
Compressed archives: 7898.3 MB  
  
Archive 1, Name: 99-07-15.09:43:15-036-1-1.tar.gz, Total Size: 2673.1 MB  
  Database 1, Name: evs_g_isPhysicsEvents_col004DC2, Size: 1507.3 MB  
  Database 2, Name: evs_g_isPhysicsEvents_col004F79, Size: 47.8 MB  
  Database 3, Name: evs_g_isPhysicsEvents_evt004F67, Size: 128.5 MB  
  Database 4, Name: evs_g_isPhysicsEvents_tag004F8F, Size: 126.4 MB  
  Database 8, Name: evs_g_isPhysicsEvents_rawhdr004FCA, Size: 591.0 MB  
  Database 10, Name: evs_g_isPhysicsEvents_esd004FD3, Size: 272.1 MB  
  
Archive 2, Name: 99-07-15.09:43:15-036-1-2.tar.gz, Total Size: 2747.2 MB  
  Database 5, Name: evs_g_isPhysicsEvents_evshdr004FC2, Size: 1437.6 MB  
  Database 6, Name: evs_g_isPhysicsEvents_rec004FD2, Size: 1309.6 MB  
  
Archive 3, Name: 99-07-15.09:43:15-036-1-3.tar.gz, Total Size: 2477.9 MB  
  Database 7, Name: evs_g_isPhysicsEvents_aod004FA8, Size: 1303.3 MB  
  Database 9, Name: evs_g_isPhysicsEvents_raw004FCD, Size: 1174.6 MB  
  
Collections:  
  /groups/isPhysicsEvents/0000/6300/P8.1.10iV02fb/00006395/cb001/99-07-15.09:43:15-036-1: 3 events
```

The data of the 3 events are stored in 10 databases. Unfortunately, 4 of these databases are between 1 and 2.5 GB. The result is ~ 7.5 GB of data to export.

The PUD Daemon can't copy the file

> When I tried to export events from analboot, I got this error:

```
% BdbCollExport -wait -exc raw:rec
/groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/99-07-08.06:10:11-021-1 -workdir
/nfs/objyserv1/objy/databases/users/albert/export
SLAC local parameters
Export 99-07-08.06:10:11-021-1 starting at 6-Aug-1999 15:46:20
Scanning the collections (can be long...)
Scanning collection /groups/isPhysicsEvents/0000/6300/P8.1.10hV00fb/00006388/cb001/99-07-
08.06:10:11-021-1
Organizing the export
Starting to export the databases...
Starting to export the databases...
Creating archive 99-07-08.06:10:11-021-1.tar.gz (2204.1875 MB)
Extracting evs_g_isPhysicsEvents_col004D80: 47.4 MB (wait mode)
15:48: Trying to lock evs_g_isPhysicsEvents_col004D80 (Wait mode)
Objectivity/DB Error #2007111: BdbDaemon::sendRequest(): The pud daemon returned an error. Error nr:
2, message: pudc: Unable to copy a file from
/objy/databases/production/resident/physics/event/physics/V1/events/groups/isPhysicsEvents/col/col100
4D00-004E00/evs_g_isPhysicsEvents_col004D80.bdb to
/nfs/objyserv1/objy/databases/users/albert/export/evs_g_isPhysicsEvents_col004D80.bdb
Objectivity/DB Error #2007023: BdbDaemon::copyFile(): Function failed:
/objy/databases/production/resident/physics/event/physics/V1/events/groups/isPhysicsEvents/col/col100
4D00-004E00/evs_g_isPhysicsEvents_col004D80.bdb
/nfs/objyserv1/objy/databases/users/albert/export/evs_g_isPhysicsEvents_col004D80.bdb
15:48: Failed to copy evs_g_isPhysicsEvents_col004D80 to
/nfs/objyserv1/objy/databases/users/albert/export/evs_g_isPhysicsEvents_col004D80.bdb (Copy failed)
./bin/SunOS5/BdbDistCopy "evs_g_isPhysicsEvents_col004D80" failed: status 1
command: ./bin/SunOS5/BdbDistCopy -replace -wait -db evs_g_isPhysicsEvents_col004D80
/nfs/objyserv1/objy/databases/users/albert/export/evs_g_isPhysicsEvents_col004D80.bdb
```

> Do you know what went wrong?

The PUD daemon is used to manage the databases. It runs on each disk server and copies the files to your working directory. If it cannot write to this directory, the export fails.

You have to change the protections of this directory using:

```
chmod g+w /nfs/objyserv1/objy/databases/users/albert/export
```

Can't Grant a Lock

The export failed with a lock error. The final part of the log is the following:

```
> Extracting evs_u_young_evt002017 (4.81 MB)
> oocopydb -external -exists delete -outfile \
> /nfs/objyserv2/u7/databases/albert/workdir//evs_u_young_evt002017.bdb \
> -db evs_u_young_evt002017 \
> /nfs/objyserv2/u2/databases/qa/7.10.3/test/BaBar.BOOT
>
> Objectivity/DB (TM) Copy Database Utility, Version 5.0 Build: 58
> Copyright (c) Objectivity, Inc 1992, 1998. All rights reserved.
>
> Checking the Database for external associations...
> The Database has external associations.
> Now copying the Database File...
> ** System Error #3001: Cannot grant the requested lock; conflict with an
> existing lock.
> - context = #?--?--?
> ** System Error #2530: Object Manager was unable to lock the Database
> ** Error #2828: DC: Unable to copy the Database File to
> "/nfs/objyserv2/u7/databases/albert/workdir//evs_u_young_evt002017.bdb".
> ** Error #2928: oocopydb : An error has occurred. Processing terminated.
> oocopydb "evs_u_young_evt002017" failed: status 1
```

By default, the export tools use the standard export program from Objectivity, named *oocopydb*. This program does not accept any concurrent access to the database. I don't know why. Perhaps this is to ensure the database is really in a safe state?

The error listed above clearly shows the problem: the tool cannot get the lock because of a conflict with an existing lock (though the message could be clearer):

```
> ** System Error #3001: Cannot grant the requested lock; conflict with an
> existing lock.
> - context = #?-?-?-?
> ** System Error #2530: Object Manager was unable to lock the Database
```

The work-around we are using is to put a read lock on the database and copy it using an UNIX tool (like *cp*). This read lock guaranties that there are no write access to the database. This is *supposed* to ensure that the database is in a consistent state and can be safely copied.

This special mode is activated by using the **-wait** option. To avoid to wait infinitely, you can add the **-delay** option. If after the specified number of minutes, the read lock can't be granted, the system will return an error.

Import Lock Conflict

The same problem can occur during an import. If users are accessing a database that needs to be replaced, the import will fail on a lock conflict error. The work around is the same: use the **-wait**, and optionally, **-delay** options. In such a case, the import tool will try to lock the database to be replaced in *update* mode, to be sure that no user is accessing it (even in read mode).

Unfortunately, very frequently-accessed databases, like the databases of the Conditions and Configuration domains can't be swept so easily, because the operator trying an import will never have a chance to be granted the lock. The only solution seems to be to freeze the federation in order to perform such an import. This becomes an administration issue.