BART Reference Guide

Release 2.5.3

BART Reference Guide

May 28, 2020

Contents

This guide acts as a quick reference for BART subcommands and provides comprehensive examples of the following BART operations:

- Performing a full backup of database servers
- Performing a point-in-time recovery (PITR) on a remote PostgreSQL database server
- Restoring an incremental backup
- Restoring a database cluster with tablespaces
- Evaluating, marking, and deleting backups and incremental backups
- Local and remote database server configuration and operation

For detailed information about BART subcommands and operations, see the EDB Postgres Backup and Recovery User Guide.

The document is organized as follows:

- See *Subcommands* for information and examples related to BART subcommands.
- See *Examples* for sample BART operations.
- See *Sample BART System* to view examples of both local and remote database server configuration and operation.

CHAPTER 1

BART Subcommand Syntax and Examples

This section briefly describes each BART subcommand and provides an example.

Invoking BART

BART subcommands are invoked at the Linux command line. You can invoke the `bart` program (located in the `<BART_HOME>/bin` directory) with the desired options to manage your BART installation.

The following examples demonstrate ways of invoking BART. In these examples, the BART user account is named `bartuser`.

```
$ su bartuser
Password:
$ export
LD_LIBRARY_PATH=/opt/PostgresPlus/9.5AS/lib/:$LD_LIBRARY_PATH
$ ./bart SHOW-SERVERS
```

To run BART from any current working directory:

```
$ su bartuser
Password:
$ export
LD_LIBRARY_PATH=/opt/PostgresPlus/9.5AS/lib/:$LD_LIBRARY_PATH
$ bart SHOW-SERVERS
```

Syntax for invoking BART

```
bart [ <general_option> ]... [ <subcommand> ] [<subcommand_option>]...
```

You can use either abbreviated or long option forms on the command line (for example `-h` or `--help`).

General Options

You can specify the following general options with `bart`.

`-h` or (`--help`)

- Displays general syntax and information about BART usage.
- All subcommands support a help option (`-h, --help`). If the help option is specified, information is displayed regarding that particular subcommand. The subcommand, itself, is not executed.

The following code sample displays information about the result of invoking the `--help` option for the `BACKUP` subcommand:

```
-bash-4.2$ bart BACKUP --help
bart: backup and recovery tool

Usage:
bart BACKUP [OPTION]...

Options:
-h, --help Show this help message and exit
-s, --server Name of the server or 'all' (full backups only) to specify all servers
-F, --format=p|t Backup output format (tar (default) or plain)
-z, --gzip Enables gzip compression of tar files
-c, --compress-level Specifies the compression level (1 through 9, 9 being
     best compression)
--backup-name Specify a friendly name for the current backup
--parent Specify parent backup for incremental backup
--check Verify checksum of required mbm files
```

`-v` (or `--version`)

The following code sample displays information about version while executing the BART version subcommand.

```
[edb@localhost bin]$ bart --version
bart (EnterpriseDB) 2.5.2
[edb@localhost bin]$
```

`-d` (or `--debug`)

The following code sample displays information about debugging output while executing the BART manage subcommand.

```
-bash-4.1$ bart -d MANAGE -n
DEBUG: Server: acctg, Now: 2015-04-17 16:34:03 EDT, RetentionWindow:
259200 (secs) ==> 72 hour(s)
DEBUG: Server: dev, Now: 2015-04-17 16:34:03 EDT, RetentionWindow:
1814400 (secs) ==> 504 hour(s)
DEBUG: Server: hr, Now: 2015-04-17 16:34:03 EDT, RetentionWindow:
7776000 (secs) ==> 2160 hour(s)
```

`-c` (or `--config-path`) `<config_file_path>`

The following code sample displays information about including the `-c` option with the configuration file name and path. This option is used if you do not want to use the default BART configuration file `BART_HOME/etc/bart.cfg`.

```
$ su bartuser
Password:
$ export
LD_LIBRARY_PATH=/opt/PostgresPlus/9.5AS/lib/:$LD_LIBRARY_PATH
$ bart -c /home/bartuser/bart.cfg SHOW-SERVERS
```

The following section describes the BART subcommands. The option `help` is omitted from the syntax diagrams in the following sections for the purpose of providing clarity for the subcommand options.

1.1 BACKUP

Use the `BACKUP` subcommand to create a full or incremental backup.

Syntax for a Full Backup:

```
bart BACKUP -s { <server_name> | all } [ -F { p | t } ]

[ -z ] [ -c <compression_level> ]

[ --backup-name <backup_name> ]

[ --thread-count <number_of_threads> ]

[ { --with-pg_basebackup | --no-pg_basebackup } ]
```

Syntax for an Incremental Backup:

```
bart BACKUP -s <server_name> [-Fp]

[ --parent { <backup_id> | <backup_name> } ]

[ --backup-name <backup_name> ]

[ --thread-count <number_of_threads> ]
```

Please note that before performing an incremental backup, you must take a full backup. For more details about incremental backup, refer to *Block-Level Incremental Backup* in the EDB Postgres Backup and Recovery User Guide.

The following table describes the `BACKUP` options:

Options	Description
`-s` or `--server { <server_name> | all }`	Use this option to specify the database server to be backed up. Specify `<server_name>` to take a backup of the database server (as specified in the BART configuration file). Specify `all` to take a backup of all servers.
`-F` or `--format { p | t }`	Use this option to specify the backup file format. Specify `p` option to take backup in plain text format and specify `t` option to take backup in tar format. If the `p` or `t` option is omitted, the default is tar format. Use `p` option with the `BACKUP` subcommand when streaming is used as a backup method. An incremental backup can only be taken in plain text format (`p`).

Continued on next page

Table 1.1 – continued from previous page

Options	Description
`-z` or `--gzip` (applicable only for full backup and `tar` format)	Use this option to enable gzip compression of tar files using the default compression level (typically 6).
`-c` or `--compress-level <compression_level>` (applicable only for full backup and tar format)	Use this option to specify the gzip compression level on the tar file output. `<compression_level>` is a digit from 1 through 9, with 9 being the best compression.
`--backup-name <backup_name>`	Use this option to assign a user-defined, alphanumeric friendly name to the backup. The maximum permitted length of backup name is 49 characters. For detailed information about this parameter, see the EDB Postgres Backup and Recovery User Guide. If the option `--backup-name` is not specified and the `backup_name` parameter is not set for this database server in the BART configuration file, then the backup can only be referenced in other BART subcommands by the BART assigned backup identifier.
`--thread-count <number_of_threads>`	Use this option to specify the number of worker threads to run in parallel to copy blocks for a backup. For detailed information about the `--thread-count` parameter, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide.
`--with-pg_basebackup` (applicable only for full backup)	Use this option to specify the use of `pg_basebackup` to take a full backup. The number of thread counts in effect is ignored as given by the `thread_count` parameter in the BART configuration file. When taking a full backup, if the thread count in effect is greater than `1`, then the `pg_basebackup` utility is not used to take the full backup (parallel worker threads are used) unless the `--with-pg_basebackup` option is specified with the `BACKUP` subcommand.
`--no-pg_basebackup` (applicable only for full backup)	Use this option to specify that `pg_basebackup` is not to be used to take a full backup. When taking a full backup, if the thread count in effect is only `1`, then the `pg_basebackup` utility is used to take the full backup unless the `--no-pg_basebackup` option is specified with the `BACKUP` subcommand.
`--parent { <backup_id> | <backup_name> }`	Use this option to take an incremental backup. The parent backup is a backup taken prior to the incremental backup; it can be either a full backup or an incremental backup. `<backup_id>` is the backup identifier of a parent backup and `<backup_name>` is the user-defined alphanumeric name of a parent backup.

Continued on next page

Table 1.1 – continued from previous page

Options	Description
`--check` (applicable only for incremental backup)	Use this option to verify if the required MBM files are present in the BART backup catalog before taking an incremental backup. However, an actual incremental backup is not taken when the `--check` option is specified. The `--parent` option must be used along with the `--check` option.

Examples

The following code sample demonstrates using variables with the `BACKUP` subcommand:

```
./bart backup -s ppas12 -Ft --backup-name "YEAR = %year MONTH =
%month DAY = %day"
```

```
./bart backup -s ppas12 -Ft --backup-name "YEAR = %year MONTH =
%month DAY = %day %%"
```

```
./bart show-backups -s ppas12 -i "test backup"
```

The following code sample displays the result of creating a full backup in the default tar format with gzip compression when the `BACKUP` subcommand was invoked. Note that checksums are generated for the full backup and user-defined tablespaces for the tar format backup:

```
[edb@localhost bin]$ ./bart BACKUP -s hr -z
INFO:  DebugTarget - getVar(checkDiskSpace.bytesAvailable)
INFO:  new backup identifier generated 1567591909098
INFO:  creating 5 harvester threads
NOTICE:  all required WAL segments have been archived
INFO:  backup completed successfully
INFO:
BART VERSION: 2.5
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1567591909098
BACKUP NAME: none
BACKUP PARENT: none
BACKUP LOCATION: /home/edb/bkup_new/hr/1567591909098
BACKUP SIZE: 13.91 MB
BACKUP FORMAT: tar.gz
BACKUP TIMEZONE: America/New_York
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 0
TABLESPACE(s): 3
Oid     Name    Location
16387   test1   /home/edb/tbl1
16388   test2   /home/edb/tbl2
16389   test3   /home/edb/tbl3

START WAL LOCATION: 000000010000000000000025
STOP WAL LOCATION: 000000010000000000000026
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2019-09-04 06:11:49 EDT
```

```
STOP TIME: 2019-09-04 06:11:53 EDT
TOTAL DURATION: 4 sec(s)
```

The following code sample displays information about the directory containing the full backup:

```
[edb@localhost bin]$number_of_threads>
[edb@localhost bin]$ ls -l /home/edb/bkup_new/hr/
total 8
drwxrwxr-x. 3 edb edb   34 Aug 27 05:57 1566899819709
drwxrwxr-x. 3 edb edb   58 Aug 27 05:57 1566899827751
drwxrwxr-x. 3 edb edb 4096 Sep  4 06:11 1567591909098
drwxrwxr-x. 2 edb edb 4096 Sep  4 06:11 archived_wals
[edb@localhost bin]$
```

The following code sample displays information about the creation of a full backup while streaming the transaction log. Note that the `-Fp` option must be specified with the `BACKUP` subcommand when streaming is used as a backup method.

```
[edb@localhost bin]$ ./bart BACKUP -s ACCTG -Fp
INFO: DebugTarget - getVar(checkDiskSpace.bytesAvailable)
INFO: new backup identifier generated 1566898964200
INFO: creating 5 harvester threads
NOTICE: pg_stop_backup complete, all required WAL segments have been archived
INFO: backup completed successfully
INFO:
BART VERSION: 2.5
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1566898964200
BACKUP NAME: none
BACKUP PARENT: none
BACKUP LOCATION: /home/edb/bkup_new/acctg/1566898964200
BACKUP SIZE: 46.03 MB
BACKUP FORMAT: plain
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 0
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000017
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2019-08-27 05:42:44 EDT
STOP TIME: 2019-08-27 05:42:46 EDT
TOTAL DURATION: 2 sec(s)
```

The following code sample displays the assignment of a user-defined backup name with the `--backup-name` option:

```
[edb@localhost bin]$ ./bart BACKUP -s acctg --backup-name acctg_%year-%month-%day
INFO: DebugTarget - getVar(checkDiskSpace.bytesAvailable)
INFO: new backup identifier generated 1566899004804
INFO: creating 5 harvester threads
NOTICE: pg_stop_backup complete, all required WAL segments have been archived
INFO: backup completed successfully
INFO:
BART VERSION: 2.5
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1566899004804
```

```
BACKUP NAME: acctg_2019-08-27
BACKUP PARENT: none
BACKUP LOCATION: /home/edb/bkup_new/acctg/1566899004804
BACKUP SIZE: 46.86 MB
BACKUP FORMAT: tar
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 0
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000001A
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2019-08-27 05:43:24 EDT
STOP TIME: 2019-08-27 05:43:24 EDT
TOTAL DURATION: 0 sec(s)
```

The following code sample displays an incremental backup taken by specifying the `--parent` option. The option `-Fp` must be specified while taking an incremental backup as incremental backup can be taken only in plain text format.

```
[edb@localhost bin]$ ./bart BACKUP -s hr -Fp --parent hr_full_1 --backup-name
hr_incr_1
INFO: DebugTarget - getVar(checkDiskSpace.bytesAvailable)
INFO: checking /home/edb/bkup_new/hr/archived_wals for MBM files from 0/20000028 to
0/22000000
INFO: new backup identifier generated 1566899827751
INFO: creating 5 harvester threads
NOTICE: all required WAL segments have been archived
INFO: backup completed successfully
INFO:
BART VERSION: 2.5
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1566899827751
BACKUP NAME: hr_incr_1
BACKUP PARENT: 1566899819709
BACKUP LOCATION: /home/edb/bkup_new/hr/1566899827751
BACKUP SIZE: 7.19 MB
BACKUP FORMAT: plain
BACKUP TIMEZONE: America/New_York
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 0
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000022
STOP WAL LOCATION: 000000010000000000000023
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2019-08-27 05:57:07 EDT
STOP TIME: 2019-08-27 05:57:08 EDT
TOTAL DURATION: 1 sec(s)
```

1.2 CHECK-CONFIG

The CHECK-CONFIG subcommand checks the parameter settings in the BART configuration file as well as the database server configuration for which the -s option is specified.

Syntax:

The following syntax is used to check the BART configuration file global section settings.

```
bart CHECK-CONFIG
```

The following syntax is used to check the database server configuration settings.

```
bart CHECK-CONFIG [ -s <server_name> ]
```

The following table describes the CHECK-CONFIG option:

Option	Description
-s (or --server) <server_name>	<server_name> is the name of the database server whose configuration parameter settings are to be checked.

Example

The following code sample demonstrates successfully checking the BART configuration file global parameters with the bart CHECK-CONFIG command:

```
bash-4.1$ bart CHECK-CONFIG
INFO: Verifying that pg_basebackup is executable
INFO: success -
INFO: success - pg_basebackup(/usr/edb/as11/bin/pg_basebackup) returns
version 11.400000
```

The following code sample demonstrates successfully checking the BART configuration file database server parameters with the bart CHECK-CONFIG command with the -s option:

```
[edb@localhost bin]$ ./bart check-config -s hr
INFO: Checking server hr
INFO: Verifying cluster_owner and ssh/scp connectivity
INFO: success
INFO: Verifying user, host, and replication connectivity
INFO: success
INFO: Verifying that user is a database superuser
INFO: success
INFO: Verifying that cluster_owner can read cluster data files
INFO: success
INFO: Verifying that you have permission to write to vault
INFO: success
INFO: /home/edb/bkup_new/hr
INFO: Verifying database server configuration
INFO: success
INFO: Verifying that WAL archiving is working
INFO: waiting 30 seconds for
/home/edb/bkup_new/hr/archived_wals/000000010000000000000001E
INFO: success
INFO: Verifying that bart-scanner is configured and running
INFO: success
```

1.3 DELETE

The DELETE subcommand removes the subdirectory and data files from the BART backup catalog for the specified backups along with archived WAL files.

Syntax:

```
bart DELETE -s <server_name>
-i { all | ']{ <backup_id> | <backup_name> },... }['] }
[ -n ]
```

Note that when invoking the DELETE subcommand, you must specify a database server.

For database servers under a retention policy, there are conditions where certain backups may not be deleted. For more information, see the EDB Postgres Backup and Recovery User Guide.

The following table describes the DELETE options:

Options	Description
-s (or --server) <server_name>	<server_name> is the name of the database server whose backups are to be deleted.
-i (or --backupid) { all | ']{ <backup_id> | <backup_name> }',... }[`] }	<backup_id> is the backup identifier of the backup to be deleted. <backup_name> is the user-defined alphanumeric name for the backup. Multiple backup identifiers and backup names may be specified in a comma-separated list. The list must be enclosed within single quotes if there is any white space appearing before or after each comma (see *Example*). If all is specified, all backups and their archived WAL files for the specified database server are deleted.
-n or --dry-run	Performs the test run and displays the results prior to physically removing files; no files are actually deleted.

Example

The following code sample demonstrates deleting a backup from the specified database server:

```
[edb@localhost bin]$ ./bart DELETE -s acctg -i acctg_2019-08-27
INFO: deleting backup 'acctg_2019-08-27' of server 'acctg'
INFO: deleting backup '1566900093665'
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or will
be marked unused
WARNING: not marking any WALs as unused WALs, the WAL file
'/home/edb/bkup_new/acctg/archived_wals/000000010000000000000025'
is required, yet not available in archived_wals directory
INFO: backup(s) deleted
[edb@localhost bin]$
```

After the deletion, the BART backup catalog for the database server no longer contains the corresponding directory for the deleted backup ID. The following code sample displays information about archived_wals subdirectory that no longer contains the backup WAL files:

```
[edb@localhost acctg]$ ls -l
total 16
```

```
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:03 1566900199604
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:03 1566900204377
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:03 1566900209087
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:05 1566900321228
drwxrwxr-x. 2 edb edb 6 Aug 27 06:01 archived_wals
```

The following code sample demonstrates deleting multiple backups from the database server.

```
[edb@localhost bin]$ ./bart DELETE -s acctg -i `1566988095633,1566988100760,
acctg_2019-08-28`
INFO: deleting backup `1566988095633` of server `acctg`
INFO: deleting backup `1566988095633`
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or will
be marked unused
WARNING: not marking any WALs as unused WALs, the WAL file
`/home/edb/bkup_new/acctg/archived_wals/000000010000000000000037` is required,
yet not available in archived_wals directory
INFO: backup(s) deleted
INFO: deleting backup `1566988100760` of server `acctg`
INFO: deleting backup `1566988100760`
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or will
be marked unused
WARNING: not marking any WALs as unused WALs, the WAL file
`/home/edb/bkup_new/acctg/archived_wals/000000010000000000000039` is
required, yet not available in archived_wals directory
INFO: backup(s) deleted
INFO: deleting backup `acctg_2019-08-28` of server `acctg`
INFO: deleting backup `1566988115512`
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or will
be marked unused
WARNING: not marking any WALs as unused WALs, the WAL file
`/home/edb/bkup_new/acctg/archived_wals/00000001000000000000003C` is required,
yet not available in archived_wals directory
INFO: backup(s) deleted
[edb@localhost bin]$
[edb@localhost bin]$
[edb@localhost bin]$
[edb@localhost acctg]$
[edb@localhost acctg]$ ls -l
total 8
drwxrwxr-x. 3 edb edb 4096 Aug 28 06:28 1566988105086
drwxrwxr-x. 3 edb edb 4096 Aug 28 06:28 1566988109477
drwxrwxr-x. 2 edb edb 6 Aug 28 06:09 archived_wals
[edb@localhost acctg]$
```

Deleting Multiple Backups with Space Characters

The following code sample also demonstrates deleting multiple backups; since there are space characters in the comma-separated list, the entire list must be enclosed within single quotes:

```
[edb@localhost bin]$ ./bart DELETE -s acctg -i
`1566900199604,1566900204377,1566900209087`;
INFO: deleting backup `1566900199604` of server `acctg`
INFO: deleting backup `1566900199604`
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or will
be marked unused
WARNING: not marking any WALs as unused WALs, the WAL file
`/home/edb/bkup_new/acctg/archived_wals/000000010000000000000028` is required,
```

```
yet not available in archived_wals directory
INFO: backup(s) deleted
INFO: deleting backup `1566900204377` of server `acctg`
INFO: deleting backup `1566900204377`
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or will
be marked unused
WARNING: not marking any WALs as unused WALs, the WAL file
`/home/edb/bkup_new/acctg/archived_wals/000000010000000000000002A` is required,
yet not available in archived_wals directory
INFO: backup(s) deleted
INFO: deleting backup `1566900209087` of server `acctg`
INFO: deleting backup `1566900209087`
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or will
be marked unused
WARNING: not marking any WALs as unused WALs, the WAL file
`/home/edb/bkup_new/acctg/archived_wals/000000010000000000000002C` is required,
yet not available in archived_wals directory
INFO: backup(s) deleted
[edb@localhost bin]$
[edb@localhost bin]$
[edb@localhost acctg]$ ls -l
total 4
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:05 1566900321228
drwxrwxr-x. 2 edb edb 6 Aug 27 06:01 archived_wals
[edb@localhost acctg]$
```

1.4 INIT

The INIT subcommand is used to create the BART backup catalog directory, rebuild the BART backupinfo file, and set the archive_command in the server based on the archive_command setting in the bart.cfg file.

Syntax:

```
bart INIT [ -s { <server_name> | all } ] [ -o ]

[ -r [ -i { <backup_id> | <backup_name> | all } ] ]

[-- no-configure]
```

The following table describes the INIT options:

Options	Description
-s or --server { <server_name> | all }	<server_name> is the name of the database server to which the INIT actions are to be applied. If all is specified or if the option is omitted, actions are applied to all servers.
-o or -override	Overrides the existing Postgres archive_command configuration parameter setting in the postgresql.conf file or the postgresql.auto.conf file using the BART archive_command parameter in the BART configuration file. The INIT generated archive command string is written to the postgresql.auto.conf file.
-r or -rebuild	Rebuilds the backupinfo file located in each backup subdirectory. If all is specified or if the option is omitted, the backupinfo files of all backups for the database servers specified by the -s option are recreated. This option is only intended for recovering from a situation where the backupinfo file has become corrupt. If the backup was initially created with a user-defined backup name, and then the INIT -r option is invoked to rebuild that backupinfo file, the user-defined backup name is no longer available. Thus, future references to the backup must use the backup identifier.
-i or --backupid { <backup_id> | <backup_name> | all }	<backup_id> is an integer, backup identifier and <backup_name> is the user-defined alphanumeric name for the backup. The -i option can only be used with the -r option.
--no-configure	Prevents the archive_command from being set in the PostgreSQL server.

Examples

In the following code sample, you can see that archive_mode = off and archive_command is not set. After invoking the BART INIT subcommand, archive_mode is set to on and archive_command is set:

```
archive_mode = off # enables archiving; off, on, or always
# (change requires restart)
archive_command = ''
# command to use to archive a logfile segment
[edb@localhost bin]$ ./bart init -s ppas11
INFO: setting archive_mode/archive_command for server 'ppas11'
```

```
WARNING: archive_mode/archive_command is set. Restart the PostgreSQL
server using 'pg_ctl restart'
[edb@localhost bin]$
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
archive_mode = 'on'
archive_command = 'scp %p
edb@127.0.0.1:/home/edb/bkup/ppas11/archived_wals/%f'
```

In the following code sample, you can see that `archive_mode = on`, and `archive_command` is not set. After invoking the `INIT` subcommand, `archive_command` is set:

```
archive_mode = on # enables archiving; off, on, or always
# (change requires restart)
archive_command = '' # command to use to archive a logfile segment
[edb@localhost bin]$ ./bart init -s ppas11
INFO: setting archive_mode/archive_command for server 'ppas11'
WARNING: archive_command is set. Reload the configuration in the
PostgreSQL server using pg_reload_conf() or 'pg_ctl reload'
[edb@localhost bin]$
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
archive_command = 'scp %p
edb@127.0.0.1:/home/edb/bkup/ppas11/archived_wals/%f'
```

In the following code sample, you can see that `archive_mode = on` and `archive_command` are already set. After invoking the `INIT` subcommand, there is no change in their settings. Note that to override the existing `archive_command`, you must include the `-o` option.

```
archive_mode = on # enables archiving; off, on, or always
# (change requires restart)
archive_command = 'scp %p
edb@127.0.0.1:/home/edb/bkup/ppas11/archived_wals/%f' # command to use
to archive a logfile segment
# placeholders: %p = path of file to archive
[edb@localhost bin]$ ./bart init -s ppas11
INFO: setting archive_mode/archive_command for server 'ppas11'
WARNING: archive_command is not set for server 'ppas11'
[edb@localhost bin]$
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
```

In the following code sample, you can see that `archive_mode = off` and `archive_command` is already set. After invoking the `INIT` subcommand `archive_mode` is set to `on`:

```
archive_mode = off # enables archiving; off, on, or always
# (change requires restart)
archive_command = 'scp %p
edb@127.0.0.1:/home/edb/bkup/ppas11/archived_wals/%f' # command to use
to archive a log file segment
[edb@localhost bin]$ ./bart init -s ppas11
INFO: setting archive_mode/archive_command for server 'ppas11'
WARNING: archive_mode/archive_command is set. Restart the PostgreSQL
server using 'pg_ctl restart'
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
archive_mode = 'on'
```

```
archive_command = 'scp %p
edb@127.0.0.1:/home/edb/bkup/ppas11/archived_wals/%f'
```

In the following code sample an existing `archive command` setting is overridden by resetting the `archive_command` in the PostgreSQL server with the `archive_command = 'cp %p %a/%f'` parameter from the `bart.cfg` file:

The following parameters are set in the `bart.cfg` file:

```
[BART]

bart_host= enterprisedb@192.168.2.22
backup_path = /opt/backup_edb
pg_basebackup_path = /usr/edb/as11/bin/pg_basebackup
logfile = /tmp/bart.log
scanner_logfile = /tmp/bart_scanner.log

[ACCTG]

host = 127.0.0.1
port = 5444
user = repuser
cluster_owner = enterprisedb
archive_command = 'cp %p %a/%f'
description = "Accounting"
```

The `archive_mode` and `archive_command` parameters in the database server are set as follows:

```
edb=# SHOW archive_mode;
archive_mode
--------------
on
(1 row)
edb=# SHOW archive_command;
archive_command
-----------------------------------------------------------------
scp %p bartuser@192.168.2.22:/opt/backup/acctg/archived_wals/%f

(1 row)
```

Invoke the `INIT` subcommand with the `-o` option to override the current `archive_command` setting in the PostgreSQL server:

```
-bash-4.1$ bart INIT -s acctg -o
INFO: setting archive_mode/archive_command for server 'acctg'
WARNING: archive_command is set. Reload the configuration in the
PostgreSQL server using pg_reload_conf() or 'pg_ctl reload'
```

Reload the database server configuration; a restart of the database server is not necessary to reset only the `archive_command` parameter.

```
[root@localhost tmp]# service ppas11 reload
```

The `archive_command` in the PostgreSQL server is now set as follows:

```
edb=# SHOW archive_command;
               archive_command
----------------------------------------------
```

```
cp %p /opt/backup_edb/acctg/archived_wals/%f
(1 row)
```

The new command string is written to the `postgresql.auto.conf` file:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'cp %p /opt/backup_edb/acctg/archived_wals/%f'
```

When you invoke the BART `INIT` command with the `-r` option, BART rebuilds the `backupinfo` file using the content of the backup directory for the server specified or for all servers. The BART `backupinfo` file is initially created by the `BACKUP` subcommand and contains the backup information used by BART.

Note: If the backup was initially created with a user-defined backup name, and then the `INIT -r` option is invoked to rebuild that `backupinfo` file, the user-defined backup name is no longer available. Thus, future references to the backup must use the backup identifier.

The following code sample shows the `backupinfo` file location in a backup subdirectory:

```
[root@localhost acctg]# pwd
/opt/backup/acctg
[root@localhost acctg]# ls -l
total 4
drwx------ 2 enterprisedb enterprisedb 38 Oct 26 10:21 1477491569966
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Oct 26 10:19 archived_wals
[root@localhost acctg]# ls -l 1477491569966
total 61144
-rw-rw-r-- 1 enterprisedb enterprisedb 703 Oct 26 10:19 backupinfo
-rw-rw-r-- 1 enterprisedb enterprisedb 62603776 Oct 26 10:19 base.tar
```

The following code sample displays the `backupinfo` file content:

```
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1477491569966
BACKUP NAME: none
BACKUP PARENT: none
BACKUP LOCATION: /opt/backup/acctg/1477491569966
BACKUP SIZE: 59.70 MB
BACKUP FORMAT: tar
BACKUP TIMEZONE:
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
ChkSum File
84b3eeb1e3f7b3e75c2f689570d04f10 base.tar
TABLESPACE(s): 0
START WAL LOCATION: 2/A5000028 (file 000000010000000200000A5)
STOP WAL LOCATION: 2/A50000C0 (file 000000010000000200000A5)
CHECKPOINT LOCATION: 2/A5000028
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2016-10-26 10:19:30 EDT
LABEL: pg_basebackup base backup
STOP TIME: 2016-10-26 10:19:30 EDT
TOTAL DURATION: 0 sec(s)
```

The following code sample displays an error message if the backupinfo file is missing when invoking a BART subcommand:

```
-bash-4.2$ bart SHOW-BACKUPS
ERROR: 'backupinfo' file does not exist for backup '1477491569966'
please use 'INIT -r' to generate the file
```

The backupinfo file may be missing if the BACKUP subcommand did not complete successfully.

The following code sample displays information about rebuilding the backupinfo file of the specified backup for database server acctg:

```
-bash-4.1$ bart INIT -s acctg -r -i 1428346620427
INFO: rebuilding BACKUPINFO for backup '1428346620427' of server 'acctg'
INFO: backup checksum: ced59b72a7846ff8fb8afb6922c70649 of base.tar
```

The following code sample displays information about how the backupinfo files of all backups are rebuilt for all database servers:

```
-bash-4.1$ bart INIT -r

INFO: rebuilding BACKUPINFO for backup '1428347191544' of server 'acctg'
INFO: backup checksum: 1ac5c61f055c910db314783212f2544f of base.tar
INFO: rebuilding BACKUPINFO for backup '1428346620427' of server 'acctg'
INFO: backup checksum: ced59b72a7846ff8fb8afb6922c70649 of base.tar
INFO: rebuilding BACKUPINFO for backup '1428347198335' of server 'dev'
INFO: backup checksum: a8890dd8ab7e6be5d5bc0f38028a237b of base.tar
INFO: rebuilding BACKUPINFO for backup '1428346957515' of server 'dev'
INFO: backup checksum: ea62549cf090573625d4adeb7d919700 of base.tar
```

The following code sample displays information about invoking BART INIT with the -r - i option:

```
edb@localhost bin]$ ./bart init -s ppas11 -i 1551778898392 -r
INFO: rebuilding BACKUPINFO for backup '1551778898392' of server
'ppas11'
[edb@localhost bin]$ ls /home/edb/bkup/ppas11/1551778898392/
backupinfo backup_label base base-1.tar base-2.tar base-3.tar
base-4.tar base-5.tar base.tar
```

The following code sample displays information about invoking the BART INIT command with the --no-configure option. You can use the --no-configure option with the INIT subcommand to prevent the archive_command option from being set in the PostgreSQL server.

```
[edb@localhost bin]$ ./bart init -s ppas11 -o --no-configure
[edb@localhost bin]$
# Do not edit this file manually!
# It will be overwritten by the ALTER SYSTEM command.
```

1.5 MANAGE

The MANAGE subcommand can be invoked to:

- Evaluate backups, mark their status, and delete obsolete backups based on the retention_policy parameter in the BART configuration file.
- Compress the archived WAL files based on the wal_compression parameter in the BART configuration file.

Syntax:

```
bart MANAGE [ -s { <server_name> | all} ]
[ -l ] [ -d ]
[ -c { keep | nokeep }
-i { <backup_id> | <backup_name> | all } ]
[ -n ]
```

To view detailed information about the MANAGE subcommand and retention policy management, see *the EDB Postgres Backup and Recovery User Guide*. For information about setting the wal_compression parameter, see the *EDB Postgres Backup and Recovery Installation and Upgrade Guide*. These guides are available at the EnterpriseDB documentation web page.

The following table describes the MANAGE options:

Options	Description
-s or --server [<server_name> | all]	<server_name> is the name of the database server to which the MANAGE actions are to be applied. If all is specified or if the -s option is omitted, actions are applied to all database servers.
-l or --list-obsolete	Lists the backups marked as obsolete.
-d or --delete-obsolete	Deletes the backups marked as obsolete. This action physically deletes the backup along with its archived WAL files and any MBM files for incremental backups.
-c or --change-status { keep | nokeep }	Specify keep to change the backup status to keep to retain the backup indefinitely. Specify nokeep to change the backup status back to active. You can then re-evaluate and possibly mark the backup as obsolete (according to the retention policy) using the MANAGE subcommand. The -c option can only be used with the -i option.
-i or --backupid {<backup_id> | <backup_name> | all }	<backup_id> is a backup identifier and <backup_name> is the user-defined alphanumeric name for the backup. If all is specified, actions are applied to all backups. The -i option can only be used with the -c option.

Continued on next page

Table 1.5 – continued from previous page

Options	Description
`-n` or `--dry-run`	Performs the test run and displays the results prior to actually implementing the actions as if the operation was performed, however, no changes are actually made. If you specify `-n` with the `-d` option, it displays which backups would be deleted, but does not actually delete the backups. If you specify `-n` with the `-c` option, it displays the keep or nokeep action, but does not actually change the backup status. If you specify `-n` alone with no other options or if you specify `-n` with only the `-s` option, it displays which active backups would be marked as obsolete, but does not actually change the backup status. In addition, no compression is performed on uncompressed, archived WAL files even if WAL compression is enabled for the database server.

Example

The following code sample performs a dry run for the specified database server displaying which active backups are evaluated as obsolete according to the retention policy, but does not actually change the backup status:

```
-bash-4.2$ bart MANAGE -s acctg -n
INFO: processing server 'acctg', backup '1482770807519'
INFO: processing server 'acctg', backup '1482770803000'
INFO: marking backup '1482770803000' as obsolete
INFO: 1 WAL file(s) marked obsolete
INFO: processing server 'acctg', backup '1482770735155'
INFO: marking backup '1482770735155' as obsolete
INFO: 2 incremental(s) of backup '1482770735155' will be marked obsolete
INFO: marking incremental backup '1482770780423' as obsolete
INFO: marking incremental backup '1482770763227' as obsolete
INFO: 3 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: 2 Unused file(s) (WALs included) present, use 'MANAGE -l' for the
list
```

The following code sample marks active backups as obsolete according to the retention policy for the specified database server:

```
-bash-4.2$ bart MANAGE -s acctg
INFO: processing server 'acctg', backup '1482770807519'
INFO: processing server 'acctg', backup '1482770803000'
INFO: marking backup '1482770803000' as obsolete
INFO: 1 WAL file(s) marked obsolete
INFO: processing server 'acctg', backup '1482770735155'
INFO: marking backup '1482770735155' as obsolete
INFO: 2 incremental(s) of backup '1482770735155' will be marked obsolete
INFO: marking incremental backup '1482770780423' as obsolete
INFO: marking incremental backup '1482770763227' as obsolete
INFO: 3 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: 2 Unused file(s) (WALs included) present, use 'MANAGE -l' for the
list
```

The following code sample lists backups marked as obsolete for the specified database server:

```
-bash-4.2$ bart MANAGE -s acctg -l
SERVER NAME: acctg
BACKUP ID: 1482770803000
BACKUP STATUS: obsolete
BACKUP TIME: 2016-12-26 11:46:43 EST
BACKUP SIZE: 59.52 MB
WAL FILE(s): 1
WAL FILE: 000000010000000100000055
SERVER NAME: acctg
BACKUP ID: 1482770735155
BACKUP STATUS: obsolete
BACKUP TIME: 2016-12-26 11:45:35 EST
BACKUP SIZE: 59.52 MB
INCREMENTAL BACKUP(s): 2
BACKUP ID: 1482770780423
BACKUP PARENT: 1482770735155
BACKUP STATUS: obsolete
BACKUP TIME: 2016-12-26 11:45:35 EST
BACKUP SIZE: 59.52 MB
BACKUP ID: 1482770763227
BACKUP PARENT: 1482770735155
BACKUP STATUS: obsolete
BACKUP TIME: 2016-12-26 11:45:35 EST
BACKUP SIZE: 59.52 MB
WAL FILE(s): 3
WAL FILE: 000000010000000100000054
WAL FILE: 000000010000000100000053
WAL FILE: 000000010000000100000052
UNUSED FILE(s): 2
UNUSED FILE: 000000010000000100000051
UNUSED FILE: 000000010000000151000028000000152000000.mbm
```

The following code sample deletes the obsolete backups for the specified database server:

```
-bash-4.2$ bart MANAGE -s acctg -d
INFO: removing all obsolete backups of server 'acctg'
INFO: removing obsolete backup '1482770803000'
INFO: 1 WAL file(s) will be removed
INFO: removing WAL file '000000010000000100000055'
INFO: removing obsolete backup '1482770735155'
INFO: 3 WAL file(s) will be removed
INFO: 2 incremental(s) of backup '1482770735155' will be removed
INFO: removing obsolete incremental backup '1482770780423'
INFO: removing obsolete incremental backup '1482770763227'
INFO: removing WAL file '000000010000000100000054'
INFO: removing WAL file '000000010000000100000053'
INFO: removing WAL file '000000010000000100000052'
INFO: 8 Unused file(s) will be removed
INFO: removing (unused) file '000000010000000100000056.00000028.backup'
INFO: removing (unused) file '000000010000000100000056'
INFO: removing (unused) file '000000010000000100000055.00000028.backup'
INFO: removing (unused) file '000000010000000100000054.00000028.backup'
INFO: removing (unused) file '000000010000000100000053.00000028.backup'
INFO: removing (unused) file '000000010000000100000052.00000028.backup'
INFO: removing (unused) file '000000010000000100000051'
INFO: removing (unused) file
'000000010000000151000028000000152000000.mbm'
```

The following code sample changes the specified backup to keep status to retain it indefinitely:

```
-bash-4.2$ bart MANAGE -s acctg -c keep -i 1482770807519
INFO: changing status of backup '1482770807519' of server 'acctg' from
'active' to 'keep'
INFO: 1 WAL file(s) changed
-bash-4.2$ bart SHOW-BACKUPS -s acctg -i 1482770807519 -t
SERVER NAME : acctg
BACKUP ID : 1482770807519
BACKUP NAME : none
BACKUP PARENT : none
BACKUP STATUS : keep
BACKUP TIME : 2016-12-26 11:46:47 EST
BACKUP SIZE : 59.52 MB
WAL(S) SIZE : 16.00 MB
NO. OF WALS : 1
FIRST WAL FILE : 000000010000000100000057
CREATION TIME : 2016-12-26 11:52:47 EST
LAST WAL FILE : 000000010000000100000057
CREATION TIME : 2016-12-26 11:52:47 EST
```

The following code sample resets the specified backup to active status:

```
-bash-4.2$ bart MANAGE -s acctg -c nokeep -i 1482770807519
INFO: changing status of backup '1482770807519' of server 'acctg' from
'keep' to 'active'
INFO: 1 WAL file(s) changed
-bash-4.2$ bart SHOW-BACKUPS -s acctg -i 1482770807519 -t
SERVER NAME : acctg
BACKUP ID : 1482770807519
BACKUP NAME : none
BACKUP PARENT : none
BACKUP STATUS : active
BACKUP TIME : 2016-12-26 11:46:47 EST
BACKUP SIZE : 59.52 MB
WAL(S) SIZE : 16.00 MB
NO. OF WALS : 1
FIRST WAL FILE : 000000010000000100000057
CREATION TIME : 2016-12-26 11:52:47 EST
LAST WAL FILE : 000000010000000100000057
CREATION TIME : 2016-12-26 11:52:47 EST
```

The following code sample uses the enabled `wal_compression` parameter in the BART configuration file as shown by the following:

```
[ACCTG]

host = 127.0.0.1
port = 5445
user = enterprisedb
cluster_owner = enterprisedb
allow_incremental_backups = disabled
wal_compression = enabled
description = "Accounting"
```

When the `MANAGE` subcommand is invoked, the following message is displayed indicating that WAL file compression is performed:

```
-bash-4.2$ bart MANAGE -s acctg
INFO: 4 WAL file(s) compressed
WARNING: 'retention_policy' is not set for server 'acctg'
```

The following code sample shows the archived WAL files in compressed format:

```
-bash-4.2$ pwd
/opt/backup/acctg
-bash-4.2$ ls -l archived_wals
total 160
-rw------- 1 enterprisedb enterprisedb 27089 Dec 26 12:16
000000010000000010000005B.gz
-rw------- 1 enterprisedb enterprisedb 305 Dec 26 12:17
000000010000000010000005C.00000028.backup
-rw------- 1 enterprisedb enterprisedb 27112 Dec 26 12:17
000000010000000010000005C.gz
-rw------- 1 enterprisedb enterprisedb 65995 Dec 26 12:18
000000010000000010000005D.gz
-rw------- 1 enterprisedb enterprisedb 305 Dec 26 12:18
000000010000000010000005E.00000028.backup
-rw------- 1 enterprisedb enterprisedb 27117 Dec 26 12:18
000000010000000010000005E.gz
```

1.6 RESTORE

The RESTORE subcommand restores a backup and its archived WAL files for the designated database server to the specified directory location.

Syntax for Restore:

```
bart RESTORE -s <server_name> -p <restore_path>
[ -i { <backup_id> | <backup_name> } ]
[ -r <remote_user>@<remote_host_address> ]
[ -w <number_of_workers> ]
[ -t <timeline_id> ]
[ { -x <target_xid> | -g <target_timestamp> } ]
[ -c ]
```

To view detailed information about the RESTORE subcommand, see the *EDB Postgres Backup and Recovery User Guide* available at this page.

If the backup is restored to a different database cluster directory than where the original database cluster resided, then some operations dependent upon the database cluster location may fail. This happens if the supporting service scripts are not updated to reflect the new directory location of restored backup.

For information about the use and modification of service scripts, see the EDB Postgres Advanced Server Installation Guide.

The following table describes the RESTORE options:

Options	Description
-s or --server <server_name>	<server_name> is the name of the database server to be restored.
-p or --restore-path <restore_path>	<restore_path> is the directory path where the backup of the database server is to be restored. The directory must be empty and have the proper ownership and privileges assigned to it.
-i or --backupid { <backup_id> | <backup_name>}	backup_id is the backup identifier of the backup to be used for the restoration and <backup_name> is the user-defined alphanumeric name for the backup. If the option is omitted, the latest backup is restored by default.

Continued on next page

Table 1.6 – continued from previous page

Options	Description
-r or --remote-host <remote_user@remote_host_address>	<remote_user> is the user account on the remote database server host that accepts a passwordless SSH/SCP login connection and is the owner of the directory where the backup is to be restored. <remote_host_address> is the IP address of the remote host to which the backup is to be restored. This option must be specified if the remote_host parameter for this database server is not set in the BART configuration file. For information about the remote_host parameter, see the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at this page.
-w or --workers <number_of_workers>	<number_of_workers> is the number of worker processes to run in parallel to stream the modified blocks of an incremental backup to the restore location. If the -w option is omitted, the default is 1 worker process. For example, if four worker processes are specified, four receiver processes on the restore host and four streamer processes on the BART host are used. The output of each streamer process is connected to the input of a receiver process. When the receiver gets to the point where it needs a modified block file, it obtains those modified blocks from its input. With this method, the modified block files are never written to the restore host disk.
-t or --target-tli <timeline_id>	<timeline_id> is the integer identifier of the timeline to be used for replaying the archived WAL files for point-in-time recovery.
-x or --target-xid <target_xid>	<target_xid> is the integer identifier of the transaction ID that determines the transaction up to and including, which point-in-time recovery encompasses.
-g or --target-timestamp <target_timestamp>	<target_timestamp> is the timestamp that determines the point in time up to and including, which point-in-time recovery encompasses.

Continued on next page

Table 1.6 – continued from previous page

Options	Description
`-c` or `--copy-wals`	Specify this option to copy archived WAL files from the BART backup catalog to `<restore_path>/archived_wals` directory. The `restore_command` retrieves the WAL files from `<restore_path>/archived_wals` for the database server archive recovery. If the `-c` option is omitted and the `copy_wals_during_restore` parameter in the BART configuration file is not enabled in a manner applicable to this database server, then the `restore_command` in the `postgresql.conf` retrieves the archived WAL files directly from the BART backup catalog. For information about the `copy_wals_during_restore` parameter, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide.

Examples

The following code sample restores a database server(named `mktg`) to the `/opt/restore` directory up to timestamp `2015-12-15 10:47:00`:

```
-bash-4.1$ bart RESTORE -s mktg -i 1450194208824 -p /opt/restore -t 1 -g
'2015-12-15 10:47:00'
INFO: restoring backup '1450194208824' of server 'mktg'
INFO: restoring backup to enterprisedb@192.168.2.24:/opt/restore
INFO: base backup restored
INFO: WAL file(s) will be streamed from the BART host
INFO: writing recovery settings to postgresql.auto.conf file
INFO: archiving is disabled
INFO: tablespace(s) restored
```

The following parameters are set in the `postgresql.auto.conf` file:

```
restore_command = 'scp -o BatchMode=yes -o PasswordAuthentication=no
enterprisedb@192.168.2.22:/opt/backup/mktg/archived_wals/%f %p'
recovery_target_time = '2015-12-15 10:47:00'
recovery_target_timeline = 1
```

The following is a list of the restored files and subdirectories:

```
[root@localhost restore]# pwd
/opt/restore
[root@localhost restore]# ls -l
total 108
-rw------- 1 enterprisedb enterprisedb 208 Dec 15 10:43 backup_label
drwx------ 6 enterprisedb enterprisedb 4096 Dec 2 10:38 base
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 10:42 dbms_pipe
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 11:00 global
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_clog\
-rw------- 1 enterprisedb enterprisedb 4438 Dec 2 10:38 pg_hba.conf
-rw------- 1 enterprisedb enterprisedb 1636 Nov 10 15:38 pg_ident.conf
```

```
drwxr-xr-x 2 enterprisedb enterprisedb 4096 Dec 15 10:42 pg_log
drwx------ 4 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_multixact
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 10:42 pg_notify
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_serial
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_snapshots
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 10:42 pg_stat
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 10:43 pg_stat_tmp
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_subtrans
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 11:00 pg_tblspc
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_twophase
-rw------- 1 enterprisedb enterprisedb 4 Nov 10 15:38 PG_VERSION
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 11:00 pg_xlog
-rw------- 1 enterprisedb enterprisedb 23906 Dec 15 11:00
postgresql.conf
-rw-r--r-- 1 enterprisedb enterprisedb 217 Dec 15 11:00
postgresql.auto.conf
```

Example

The following code sample performs a `RESTORE` operation with the `copy_wals_during_restore` parameter enabled to copy the archived WAL files to the local `<restore_path>/archived_wals` directory:

```
-bash-4.1$ bart RESTORE -s hr -i hr_2017-03-29T13:50 -p
/opt/restore_pg95 -t 1 -g '2017-03-29 14:01:00'
INFO: restoring backup 'hr_2017-03-29T13:50' of server 'hr'
INFO: base backup restored
INFO: copying WAL file(s) to
postgres@192.168.2.24:/opt/restore_pg95/archived_wals
INFO: writing recovery settings to postgresql.auto.conf file
INFO: archiving is disabled
INFO: permissions set on $PGDATA
INFO: restore completed successfully
```

The following parameters are set in the `postgresql.auto.conf` file:

```
restore_command = 'cp archived_wals/%f %p'
recovery_target_time = '2017-03-29 14:01:00'
recovery_target_timeline = 1
```

The following is a list of the restored files and subdirectories:

```
-bash-4.1$ pwd
/opt/restore_pg95
-bash-4.1$ ls -l
total 128
drwxr-xr-x 2 postgres postgres 4096 Mar 29 14:27 archived_wals
-rw------- 1 postgres postgres 206 Mar 29 13:50 backup_label
drwx------ 5 postgres postgres 4096 Mar 29 12:25 base
drwx------ 2 postgres postgres 4096 Mar 29 14:27 global
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_clog
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_commit_ts
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_dynshmem
-rw------- 1 postgres postgres 4212 Mar 29 13:18 pg_hba.conf
-rw------- 1 postgres postgres 1636 Mar 29 12:25 pg_ident.conf
drwxr-xr-x 2 postgres postgres 4096 Mar 29 13:45 pg_log
drwx------ 4 postgres postgres 4096 Mar 29 12:25 pg_logical
drwx------ 4 postgres postgres 4096 Mar 29 12:25 pg_multixact
drwx------ 2 postgres postgres 4096 Mar 29 13:43 pg_notify
```

```
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_replslot
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_serial
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_snapshots
drwx------ 2 postgres postgres 4096 Mar 29 13:43 pg_stat
drwx------ 2 postgres postgres 4096 Mar 29 13:50 pg_stat_tmp
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_subtrans
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_tblspc
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_twophase
-rw------- 1 postgres postgres 4 Mar 29 12:25 PG_VERSION
drwx------ 3 postgres postgres 4096 Mar 29 14:27 pg_xlog
-rw------- 1 postgres postgres 169 Mar 29 13:24 postgresql.auto.conf
-rw-r--r-- 1 postgres postgres 21458 Mar 29 14:27 postgresql.conf
-rw-r--r-- 1 postgres postgres 118 Mar 29 14:27 postgresql.auto.conf
```

1.7 SHOW-SERVERS

The SHOW-SERVERS subcommand displays information for the managed database servers listed in the BART configuration file.

Syntax:

```
bart SHOW-SERVERS [ -s { <server_name> | all } ]
```

The following table describes the SHOW-SERVERS option:

Option	Description
-s or --server { <server_name> | all }	<server_name> is the name of the database server to which the SHOW-SERVERS actions are to be applied. If all is specified or if the -s option is omitted, the actions are applied to all database servers.

Example

The following code sample shows all the database servers managed by BART as returned by the SHOW-SERVERS subcommand:

```
-bash-4.2$ bart SHOW-SERVERS
SERVER NAME : acctg
BACKUP FRIENDLY NAME: acctg_%year-%month-%dayT%hour:%minute
HOST NAME : 127.0.0.1
USER NAME : enterprisedb
PORT : 5444
REMOTE HOST :
RETENTION POLICY : 6 Backups
DISK UTILIZATION : 0.00 bytes
NUMBER OF ARCHIVES : 0
ARCHIVE PATH : /opt/backup/acctg/archived_wals
ARCHIVE COMMAND : (disabled)
XLOG METHOD : fetch
WAL COMPRESSION : disabled
TABLESPACE PATH(s) :
INCREMENTAL BACKUP : DISABLED
DESCRIPTION : "Accounting"
SERVER NAME : hr
BACKUP FRIENDLY NAME: hr_%year-%month-%dayT%hour:%minute
HOST NAME : 192.168.2.24
USER NAME : postgres
PORT : 5432
REMOTE HOST : postgres@192.168.2.24
RETENTION POLICY : 6 Backups
DISK UTILIZATION : 0.00 bytes
NUMBER OF ARCHIVES : 0
ARCHIVE PATH : /opt/backup/hr/archived_wals
ARCHIVE COMMAND : (disabled)
XLOG METHOD : fetch
WAL COMPRESSION : disabled
TABLESPACE PATH(s) :
INCREMENTAL BACKUP : DISABLED
DESCRIPTION : "Human Resources"
```

```
SERVER NAME : mktg
BACKUP FRIENDLY NAME: mktg_%year-%month-%dayT%hour:%minute
HOST NAME : 192.168.2.24
USER NAME : repuser
PORT : 5444
REMOTE HOST : enterprisedb@192.168.2.24
RETENTION POLICY : 6 Backups
DISK UTILIZATION : 0.00 bytes
NUMBER OF ARCHIVES : 0
ARCHIVE PATH : /opt/backup/mktg/archived_wals
ARCHIVE COMMAND : (disabled)
XLOG METHOD : fetch
WAL COMPRESSION : disabled
TABLESPACE PATH(s) :
INCREMENTAL BACKUP : DISABLED\
DESCRIPTION : "Marketing"
```

1.8 SHOW-BACKUPS

The SHOW-BACKUPS subcommand displays the backup information for the managed database servers.

Syntax:

```
bart SHOW-BACKUPS [ -s { <server_name> | all } ]
[ -i { <backup_id> | <backup_name> | all } ]
[ -t ]
```

The following table describes the SHOW-BACKUPS options:

Options	Description
-s or --server { <server_name> | all }	<server_name> is the name of the database server whose backup information is to be displayed. If all is specified or if the option is omitted, the backup information for all database servers is displayed.
-i or --backupid { <backup_id> | <backup_name> | all }	<backup_id> is a backup identifier and <backup_name> is the user-defined alphanumeric name for the backup. If all is specified or if the option is omitted, all backup information for the relevant database server is displayed.
-t or --toggle	Displays detailed backup information in list format. If the option is omitted, the default is a tabular format.

Example

The following code sample shows the backup from database server dev:

```
-bash-4.2$ bart SHOW-BACKUPS -s dev
SERVER NAME              BACKUP ID       BACKUP NAME               BACKUP PARENT
BACKUP TIME              BACKUP SIZE     WAL(s) SIZE               WAL FILES STATUS
dev                      1477579596637   dev_2016-10-27T10:46:36   none
2016-10-27 10:46:37 EDT  54.50 MB        96.00 MB                  6         active
```

The following code sample shows detailed information using the -t option:

```
-bash-4.2$ bart SHOW-BACKUPS -s dev -i 1477579596637 -t
SERVER NAME : dev
BACKUP ID : 1477579596637
BACKUP NAME : dev_2016-10-27T10:46:36
BACKUP PARENT : none
BACKUP STATUS : active
BACKUP TIME : 2016-10-27 10:46:37 EDT
BACKUP SIZE : 54.50 MB
WAL(S) SIZE : 80.00 MB
NO. OF WALS : 5
FIRST WAL FILE : 000000010000000010000000EC
CREATION TIME : 2016-10-27 10:46:37 EDT
LAST WAL FILE : 000000010000000010000000F0
CREATION TIME : 2016-10-27 11:22:01 EDT
```

The following code sample shows a listing of an incremental backup along with its parent backup:

```
-bash-4.2$ bart SHOW-BACKUPS
SERVER NAME              BACKUP ID         BACKUP NAME       BACKUP PARENT
BACKUP TIME              BACKUP SIZE       WAL(s) SIZE       WAL FILES       STATUS
acctg                    1477580293193     acctg_2016-10-27  none
2016-10-27 10:58:13 EDT  16.45 MB          16.00 MB          1               active
acctg 1477580111358 acctg_2016-10-27 none 2016-10-27 10:55:11 EDT 59.71
MB 16.00 MB 1 active
```

The following code sample shows the complete, detailed information of the incremental backup and the parent backup:

```
-bash-4.2$ bart SHOW-BACKUPS -t
SERVER NAME : acctg
BACKUP ID : 1477580293193
BACKUP NAME : none
BACKUP PARENT : acctg_2016-10-27
BACKUP STATUS : active
BACKUP TIME : 2016-10-27 10:58:13 EDT
BACKUP SIZE : 16.45 MB
WAL(S) SIZE : 16.00 MB
NO. OF WALS : 1
FIRST WAL FILE : 000000010000000200000D9
CREATION TIME : 2016-10-27 10:58:13 EDT
LAST WAL FILE : 000000010000000200000D9
CREATION TIME : 2016-10-27 10:58:13 EDT
SERVER NAME : acctg
BACKUP ID : 1477580111358
BACKUP NAME : acctg_2016-10-27
BACKUP PARENT : none
BACKUP STATUS : active
BACKUP TIME : 2016-10-27 10:55:11 EDT
BACKUP SIZE : 59.71 MB
WAL(S) SIZE : 16.00 MB
NO. OF WALS : 1
FIRST WAL FILE : 000000010000000200000D8
CREATION TIME : 2016-10-27 10:55:12 EDT
LAST WAL FILE : 000000010000000200000D8
CREATION TIME : 2016-10-27 10:55:12 EDT
```

1.9 VERIFY-CHKSUM

The VERIFY-CHKSUM subcommand verifies the MD5 checksums of the full backups and any user-defined tablespaces for the specified database server or for all database servers. The checksum is verified by comparing the current checksum of the backup against the checksum when the backup was taken.

Note: The VERIFY-CHKSUM subcommand is only used for tar format backups.

Syntax:

```
bart VERIFY-CHKSUM
[ -s { <server_name> | all } ]
[ -i { <backup_id> | <backup_name> | all } ]
```

The following table describes the VERIFY-CHKSUM options:

Options	Description
-s or --server { <server_name> | all }	<server_name> is the name of the database server whose tar backup checksums are to be verified. If all is specified or if the -s option is omitted, the checksums of all tar backups are verified for all database servers.
-i or --backupid {<backup_id> | <backup_name> | all }	<backup_id> is the backup identifier of a tar format full backup whose checksum is to be verified along with any user-defined tablespaces. <backup_name> is the user-defined alphanumeric name for the full backup. If all is specified or if the -i option is omitted, the checksums of all tar backups for the relevant database server are verified.

Example

The following code sample verifies the checksum of all tar format backups of the specified database server:

```
-bash-4.1$ bart VERIFY-CHKSUM -s acctg -i all
SERVER NAME    BACKUP ID     VERIFY
acctg          1430239348243 OK
acctg          1430232284202 OK
acctg          1430232016284 OK
acctg          1430231949065 OK
acctg          1429821844271 OK
```

1.10 Running the BART WAL Scanner

The BART WAL scanner is used to process each WAL file to find and record modified blocks in a corresponding MBM file. As a BART account user, use the BART WAL scanner to invoke the `bart-scanner` program located in the `<BART_HOME>/bin` directory.

For detailed information about the WAL scanner and its usage, see the EDB Postgres Backup and Recovery User Guide.

Syntax:

```
bart-scanner
[ -d ]
[ -c <config_file_path> ]
{ -h |
-v |
--daemon |
-p <mbm_file> |
<wal_file> |
RELOAD |
STOP }
```

When the `bart-scanner` program is invoked, it forks a separate process for each database server enabled with the `allow_incremental_backups` parameter.

The WAL scanner processes can run in either the foreground or background depending upon usage of the `--daemon` option:

- If the `--daemon` option is specified, the WAL scanner process runs in the background. All output messages can be viewed in the BART log file.
- If the `--daemon` option is omitted, the WAL scanner process runs in the foreground. All output messages can be viewed from the terminal running the program as well as in the BART log file.

The following table describes the `VERIFY-CHKSUM` options.

Options	Description
`-h` or `--help`	Displays general syntax and information on WAL scanner usage.
`-v` or `--version`	Displays the WAL scanner version information.
`-d` or `--debug`	Displays debugging output while executing the WAL scanner with any of its options.
`-c <config_file_path>` or `--config-path <config_file_path>`	Specifies `<config_file_path>` as the full directory path to a BART configuration file. Use this option if you do not want to use the default BART configuration file `<BART_HOME>/etc/bart.cfg`
`--daemon`	Runs the WAL scanner as a background process.
`-p <mbm_file>` or `--print <mbm_file>`	Specifies the full directory path to an MBM file whose content is to be printed. The `archived_wals` directory as specified in the the `archive_path` parameter in the `bart.cfg` file contains the MBM files.

Continued on next page

Table 1.10 – continued from previous page

Options	Description
wal_file	Specifies the full directory path to a WAL file to be scanned. The archive path directory contains the WAL files. Use it if a WAL file in the archive path is missing its MBM file. This option is to be used for assisting the EnterpriseDB support team for debugging problems that may have been encountered.
RELOAD	Reloads the BART configuration file. The keyword RELOAD is case-insensitive. The RELOAD option is useful if you make changes to the configuration file after the WAL scanner has been started. It will reload the configuration file and adjust the WAL scanners accordingly. For example, if a server section allowing incremental backups is removed from the BART configuration file, then the process attached to that server will stop. Similarly, if a server allowing incremental backups is added, a new WAL scanner process will be launched to scan the WAL files of that server.
STOP	Stops the WAL scanner. The keyword STOP is not case-sensitive.

Example

The following code sample shows the startup of the WAL scanner to run interactively. The WAL scanner begins scanning existing WAL files in the archive path that have not yet been scanned (that is, there is no corresponding MBM file for the WAL file):

```
-bash-4.2$ bart-scanner
INFO: process created for server 'acctg', pid = 5287
INFO: going to parse backlog of WALs, if any.
INFO: WAL file to be processed: 000000010000000000000ED
INFO: WAL file to be processed: 000000010000000000000EE
INFO: WAL file to be processed: 000000010000000000000EF
INFO: WAL file to be processed: 000000010000000000000F0
INFO: WAL file to be processed: 000000010000000000000F1
```

The following code sample is the content of the archive path showing the MBM files created for the WAL files. (The user name and group name of the files have been removed from the example to list the WAL files and MBM files in a more readable manner):

```
[root@localhost archived_wals]# pwd
/opt/backup/acctg/archived_wals
[root@localhost archived_wals]# ls -l
total 81944
-rw------- 1 ... ... 16777216 Dec 20 09:10 000000010000000000000ED
-rw------- 1 ... ... 16777216 Dec 20 09:06 000000010000000000000EE
-rw------- 1 ... ... 16777216 Dec 20 09:11 000000010000000000000EF
-rw------- 1 ... ... 16777216 Dec 20 09:15 000000010000000000000F0
-rw------- 1 ... ... 16777216 Dec 20 09:16 000000010000000000000F1
-rw------- 1 ... ... 305      Dec 20 09:16 000000010000000000000F1.00000028.backup
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
```

```
000000010000000ED00002800000000EE000000.mbm
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
000000010000000EE00002800000000EF000000.mbm
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
000000010000000EF00002800000000F0000000.mbm
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
000000010000000F00000280000000F1000000.mbm
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
000000010000000F10000280000000F2000000.mbm
```

To stop the interactively running WAL scanner, either enter `ctrl-C` at the terminal running the WAL scanner or invoke the `bart-scanner` program from another terminal with the `STOP` option:

```
-bash-4.2$ bart-scanner STOP
-bash-4.2$
```

The terminal on which the WAL scanner was running interactively now appears as follows after it has been stopped:

```
-bash-4.2$ bart-scanner
INFO: process created for server 'acctg', pid = 5287
INFO: going to parse backlog of WALs, if any.
INFO: WAL file to be processed: 0000000100000000000000ED
INFO: WAL file to be processed: 0000000100000000000000EE
INFO: WAL file to be processed: 0000000100000000000000EF
INFO: WAL file to be processed: 0000000100000000000000F0
INFO: WAL file to be processed: 0000000100000000000000F1
INFO: bart-scanner stopped
-bash-4.2$
```

The following code sample demonstrates invoking the WAL scanner to run as a background process with the `--daemon` option:

```
-bash-4.2$ bart-scanner --daemon
-bash-4.2$
```

The WAL scanner runs as a background process. There is also a separate background process for each database server that has been enabled for WAL scanning with the `allow_incremental_backups` parameter in the BART configuration file:

```
-bash-4.2$ ps -ef | grep bart
 enterpr+   4340  1   0   09:48   ? 00:00:00 bart-scanner --daemon
 enterpr+   4341 4340 0   09:48   ? 00:00:00 bart-scanner --daemon
 enterpr+   4415 3673 0   09:50     pts/0 00:00:00 grep   --color=auto bart
```

To stop the WAL scanner processes, invoke the WAL scanner with the `stop` option:

```
-bash-4.2$ bart-scanner STOP
-bash-4.2$
```

If it is necessary to individually scan a WAL file, this can be done as follows:

```
-bash-4.2$ bart-scanner /opt/backup/acctg/archived_wals/0000000100000000000000FF
-bash-4.2$
```

Should it be necessary to print the content of an MBM file for assisting the EnterpriseDB support team for debugging problems that may have been encountered, use the `-p` option to specify the file as shown in the following code sample:

```
-bash-4.2$ bart-scanner -p
/opt/backup/acctg/archived_wals/000000010000000FF000028000000100000000.mbm

Header:
Version: 1.0:90500:1.2.0
Scan Start: 2016-12-20 10:02:11 EST, Scan End: 2016-12-20 10:02:11 EST, Diff: 0 sec(s)
Start LSN: ff000028, End LSN: 100000000, TLI: 1
flags: 0, Check Sum: f9cfe66ae2569894d6746b61503a767d

Path: base/14845/16384
NodeTag: BLOCK_CHANGE
Relation: relPath base/14845/16384, isTSNode 0, Blocks
*.....................................................................
First modified block: 0
Total modified blocks: 1

Path: base/14845/16391
NodeTag: BLOCK_CHANGE
Relation: relPath base/14845/16391, isTSNode 0, Blocks
*.....................................................................
First modified block: 0
Total modified blocks: 1
```

CHAPTER 2

Additional Examples

This section lists examples of the following BART operations.

- Restoring a database cluster with tablespaces.
- Restoring an incremental backup.
- Managing backups.
- Managing incremental backups.

2.1 Restoring a Database Cluster with Tablespaces

The following code sample illustrates taking a backup and restoring a database cluster on a remote host containing tablespaces. For detailed information regarding using tablespaces, see the EDB Postgres Backup and Recovery User Guide.

On an Advanced Server database running on a remote host, the following tablespaces are created and used by two tables:

```
edb=# CREATE TABLESPACE tblspc_1 LOCATION '/mnt/tablespace_1';
CREATE TABLESPACE
edb=# CREATE TABLESPACE tblspc_2 LOCATION '/mnt/tablespace_2';
CREATE TABLESPACE
edb=# \db
             List of tablespaces
Name        | Owner           | Location
------------+-----------------+--------------------
pg_default  | enterprisedb    |
pg_global   | enterprisedb    |
tblspc_1    | enterprisedb    | /mnt/tablespace_1
tblspc_2    | enterprisedb    | /mnt/tablespace_2
(4 rows)

edb=# CREATE TABLE tbl_tblspc_1 (c1 TEXT) TABLESPACE tblspc_1;
```

```
CREATE TABLE
edb=# CREATE TABLE tbl_tblspc_2 (c1 TEXT) TABLESPACE tblspc_2;
CREATE TABLE
edb=# \d tbl_tblspc_1
Table "enterprisedb.tbl_tblspc_1"
Column | Type | Modifiers
-------+------+-----------
c1     | text |
Tablespace: "tblspc_1"

edb=# \d tbl_tblspc_2
Table "enterprisedb.tbl_tblspc_2"
Column | Type | Modifiers
-------+------+-----------
c1     | text |
Tablespace: "tblspc_2"
```

The following code sample shows the OIDs assigned to the tablespaces and the symbolic links to the tablespace directories:

```
-bash-4.1$ pwd
/opt/PostgresPlus/9.5AS/data/pg_tblspc
-bash-4.1$ ls -l
total 0
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Nov 16 16:17 16587 ->/mnt/tablespace_1
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Nov 16 16:17 16588 ->/mnt/tablespace_2
```

The BART configuration file contains the following settings. Note that the `tablespace_path` parameter does not have to be set at this point.

```
[BART]
bart_host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /usr/edb/as11/bin/pg_basebackup
logfile = /tmp/bart.log
scanner_logfile = /tmp/bart_scanner.log

[ACCTG]
host = 192.168.2.24
port = 5444
user = repuser
cluster_owner = enterprisedb
remote_host = enterprisedb@192.168.2.24
tablespace_path =
description = "Accounting"
```

After the necessary configuration steps are performed to ensure BART manages the remote database server, a full backup is taken as shown in the following code sample:

```
-bash-4.1$ bart BACKUP -s acctg

INFO: creating backup for server 'acctg'
INFO: backup identifier: '1447709811516'
54521/54521 kB (100%), 3/3 tablespaces

INFO: backup completed successfully
INFO: backup checksum: 594f69fe7d26af991d4173d3823e174f of 16587.tar
INFO: backup checksum: 7a5507567729a21c98a15c948ff6c015 of base.tar
```

```
INFO: backup checksum: ae8c62604c409635c9d9e82b29cc0399 of 16588.tar
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1447709811516
BACKUP NAME: none
BACKUP LOCATION: /opt/backup/acctg/1447709811516
BACKUP SIZE: 53.25 MB
BACKUP FORMAT: tar
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 3
ChkSum File
594f69fe7d26af991d4173d3823e174f 16587.tar
7a5507567729a21c98a15c948ff6c015 base.tar
ae8c62604c409635c9d9e82b29cc0399 16588.tar

TABLESPACE(s): 2
Oid Name Location
16587 tblspc_1 /mnt/tablespace_1
16588 tblspc_2 /mnt/tablespace_2
START WAL LOCATION: 000000010000000000000000F
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2015-11-16 16:36:51 EST
STOP TIME: 2015-11-16 16:36:52 EST
TOTAL DURATION: 1 sec(s)
```

Note that in the output from the preceding example, checksums are generated for the tablespaces as well as the full backup.

Within the backup subdirectory `1447709811516` of the BART backup catalog, the tablespace data is stored with file names `16587.tar.gz` and `16588.tar.gz` as shown below:

```
-bash-4.1$ pwd
/opt/backup/acctg
-bash-4.1$ ls -l
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 16:36 1447709811516
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 16:43 archived_wals
-bash-4.1$ ls -l 1447709811516
total 54536
-rw-rw-r-- 1 enterprisedb enterprisedb 19968 Nov 16 16:36 16587.tar
-rw-rw-r-- 1 enterprisedb enterprisedb 19968 Nov 16 16:36 16588.tar
-rw-rw-r-- 1 enterprisedb enterprisedb 949 Nov 16 17:05 backupinfo
-rw-rw-r-- 1 enterprisedb enterprisedb 55792640 Nov 16 16:36 base.tar
```

When you are ready to restore the backup, in addition to creating the directory to which the main database cluster is to be restored, prepare the directories to which the tablespaces are to be restored.

On the remote host, directories `/opt/restore_tblspc_1` and `/opt/restore_tblspc_2` are created and assigned the proper ownership and permissions as shown by the following example. The main database cluster is to be restored to `/opt/restore`.

```
[root@localhost opt]# mkdir restore_tblspc_1
[root@localhost opt]# chown enterprisedb restore_tblspc_1
[root@localhost opt]# chgrp enterprisedb restore_tblspc_1
[root@localhost opt]# chmod 700 restore_tblspc_1
[root@localhost opt]# mkdir restore_tblspc_2
```

```
[root@localhost opt]# chown enterprisedb restore_tblspc_2
[root@localhost opt]# chgrp enterprisedb restore_tblspc_2
[root@localhost opt]# chmod 700 restore_tblspc_2
[root@localhost opt]# ls -l
total 20
drwxr-xr-x 3 root daemon 4096 Nov 10 15:38 PostgresPlus
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:40 restore
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:40
restore_tblspc_1
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:41
restore_tblspc_2
drwxr-xr-x. 2 root root 4096 Nov 22 2013 rh
```

Set the `tablespace_path` parameter in the BART configuration file to specify the tablespace directories. The remote host user and IP address are specified by the `remote_host` configuration parameter.

```
[ACCTG]

host = 192.168.2.24
port = 5444
user = repuser
cluster_owner = enterprisedb
remote_host = enterprisedb@192.168.2.24
tablespace_path =
16587=/opt/restore_tblspc_1;16588=/opt/restore_tblspc_2

description = "Accounting"
```

The following code sample demonstrates invoking the `RESTORE` subcommand:

```
-bash-4.1$ bart RESTORE -s acctg -i 1447709811516 -p /opt/restore
INFO: restoring backup '1447709811516' of server 'acctg'
INFO: restoring backup to enterprisedb@192.168.2.24:/opt/restore
INFO: base backup restored
INFO: archiving is disabled
INFO: tablespace(s) restored
```

The following code sample shows the restored full backup (including the restored tablespaces):

```
bash-4.1$ pwd
/opt
-bash-4.1$ ls -l restore
total 104
-rw------- 1 enterprisedb enterprisedb 206 Nov 16 16:36 backup_label.old
drwx------ 6 enterprisedb enterprisedb 4096 Nov 10 15:38 base
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:46 global
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_clog
-rw------- 1 enterprisedb enterprisedb 4438 Nov 10 16:23 pg_hba.conf
-rw------- 1 enterprisedb enterprisedb 1636 Nov 10 15:38 pg_ident.conf
drwxr-xr-x 2 enterprisedb enterprisedb 4096 Nov 16 17:45 pg_log
drwx------ 4 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_multixact
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:45 pg_notify
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_serial
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_snapshots
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:47 pg_stat
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:47 pg_stat_tmp
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_subtrans
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:42 pg_tblspc
```

```
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_twophase
-rw------- 1 enterprisedb enterprisedb 4 Nov 10 15:38 PG_VERSION
drwx------ 3 enterprisedb enterprisedb 4096 Nov 16 17:47 pg_xlog
-rw------- 1 enterprisedb enterprisedb 23906 Nov 16 17:42 postgresql.conf
-rw------- 1 enterprisedb enterprisedb 61 Nov 16 17:45 postmaster.opts
-bash-4.1$
-bash-4.1$ ls -l restore_tblspc_1
total 4
drwx------ 3 enterprisedb enterprisedb 4096 Nov 16 16:18
PG_9.5_201306121
-bash-4.1$ ls -l restore_tblspc_2
total 4
drwx------ 3 enterprisedb enterprisedb 4096 Nov 16 16:18
PG_9.5_201306121
```

The symbolic links in the `pg_tblspc` subdirectory point to the restored directory location:

```
bash-4.1$ pwd
/opt/restore/pg_tblspc
-bash-4.1$ ls -l
total 0
lrwxrwxrwx 1 enterprisedb enterprisedb 21 Nov 16 17:42 16587 ->
/opt/restore_tblspc_1
lrwxrwxrwx 1 enterprisedb enterprisedb 21 Nov 16 17:42 16588 ->
/opt/restore_tblspc_2
```

`psql` queries also show the restored tablespaces:

```
edb=# \db

             List of tablespaces
Name         | Owner        |   Location
-------------+--------------+-----------------------
pg_default   | enterprisedb |
pg_global    | enterprisedb |
tblspc_1     | enterprisedb | /opt/restore_tblspc_1
tblspc_2     | enterprisedb | /opt/restore_tblspc_2
```

2.2 Restoring an Incremental Backup

Restoring an incremental backup may require additional setup steps depending upon the host on which the incremental backup is to be restored. For more information, see the EDB Postgres Backup and Recovery User Guide.

This section provides an example of creating backup chains and then restoring an incremental backup.

Creating a Backup Chain

A *backup chain* is the set of backups consisting of a full backup and all of its successive incremental backups. Tracing back on the parent backups of all incremental backups in the chain eventually leads back to that single, full backup.

In the following example, the `allow_incremental_backups` parameter is set to `enabled` in the BART configuration file to permit incremental backups on the listed database server:

```
[BART]

bart_host= enterprisedb@192.168.2.27
backup_path = /opt/backup
pg_basebackup_path = /usr/edb/as11/bin/pg_basebackup
logfile = /tmp/bart.log
scanner_logfile = /tmp/bart_scanner.log

[ACCTG]

host = 127.0.0.1
port = 5445
user = enterprisedb
cluster_owner = enterprisedb
allow_incremental_backups = enabled
description = "Accounting"
```

After the database server has been started with WAL archiving enabled to the BART backup catalog, the WAL scanner is started:

```
-bash-4.2$ bart-scanner --daemon
```

First, a full backup is taken.

```
-bash-4.2$ bart BACKUP -s acctg --backup-name full_1
INFO: creating backup for server 'acctg'
INFO: backup identifier: '1490649204327'\
63364/63364 kB (100%), 1/1 tablespace
INFO: backup completed successfully
INFO: backup checksum: aae27d4a7c09dffc82f423221154db7e of base.tar
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1490649204327
BACKUP NAME: full_1
BACKUP PARENT: none
BACKUP LOCATION: /opt/backup/acctg/1490649204327
BACKUP SIZE: 61.88 MB
BACKUP FORMAT: tar
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
ChkSum File
aae27d4a7c09dffc82f423221154db7e base.tar
```

```
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000000E
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2017-03-27 17:13:24 EDT
STOP TIME: 2017-03-27 17:13:25 EDT
TOTAL DURATION: 1 sec(s)
```

A series of incremental backups are taken. The first incremental backup specifies the full backup as the parent. Each successive incremental backup then uses the preceding incremental backup as its parent.

```
-bash-4.2$ bart BACKUP -s acctg -F p --parent full_1 --backup-name
incr_1-a
INFO: creating incremental backup for server 'acctg'
INFO: checking mbm files /opt/backup/acctg/archived_wals
INFO: new backup identifier generated 1490649255649
INFO: reading directory /opt/backup/acctg/archived_wals
INFO: all files processed
NOTICE: pg_stop_backup complete, all required WAL segments have been
archived
INFO: incremental backup completed successfully
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1490649255649
BACKUP NAME: incr_1-a
BACKUP PARENT: 1490649204327
BACKUP LOCATION: /opt/backup/acctg/1490649255649
BACKUP SIZE: 16.56 MB
BACKUP FORMAT: plain
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 0
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000010
STOP WAL LOCATION: 000000010000000000000010
BACKUP METHOD: pg_start_backup
BACKUP FROM: master
START TIME: 2017-03-27 17:14:15 EDT
STOP TIME: 2017-03-27 17:14:16 EDT
TOTAL DURATION: 1 sec(s)
-bash-4.2$ bart BACKUP -s acctg -F p --parent incr_1-a --backup-name
incr_1-b
INFO: creating incremental backup for server 'acctg'
INFO: checking mbm files /opt/backup/acctg/archived_wals
INFO: new backup identifier generated 1490649336845
INFO: reading directory /opt/backup/acctg/archived_wals
INFO: all files processed
NOTICE: pg_stop_backup complete, all required WAL segments have been
archived
INFO: incremental backup completed successfully
.
.
.
-bash-4.2$ bart BACKUP -s acctg -F p --parent incr_1-b --backup-name
incr_1-c
INFO: creating incremental backup for server 'acctg'
INFO: checking mbm files /opt/backup/acctg/archived_wals
```

```
INFO: new backup identifier generated 1490649414316
INFO: reading directory /opt/backup/acctg/archived_wals
INFO: all files processed
NOTICE: pg_stop_backup complete, all required WAL segments have been
archived
INFO: incremental backup completed successfully
.
.
.
```

The following output of the SHOW-BACKUPS subcommand lists the backup chain, which are backups full_1, incr_1-a, incr_1-b, and incr_1-c.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME BACKUP ID      BACKUP NAME   BACKUP PARENT BACKUP TIME ...
acctg       1490649414316  incr_1-c      incr_1-b      2017-03-27 17:16:55 ...
acctg       1490649336845  incr_1-b      incr_1-a      2017-03-27 17:15:37 ...
acctg       1490649255649  incr_1-a      full_1        2017-03-27 17:14:16 ...
acctg       1490649204327  full_1        none          2017-03-27 17:13:25 ...
```

For the full backup full_1, the BACKUP PARENT field contains none. For each incremental backup, the BACKUP PARENT field contains the backup identifier or name of its parent backup.

A second backup chain is created in the same manner with the BACKUP subcommand. The following example shows the addition of the resulting, second backup chain consisting of full backup full_2 and incremental backups incr_2-a and incr_2-b.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME BACKUP ID     BACKUP NAME  BACKUP PARENT  BACKUP TIME ...
acctg       1490649605607 incr_2-b     incr_2-a       2017-03-27 17:20:06 ...
acctg       1490649587702 incr_2-a     full_2         2017-03-27 17:19:48 ...
acctg       1490649528633 full_2       none           2017-03-27 17:18:49 ...
acctg       1490649414316 incr_1-c     incr_1-b       2017-03-27 17:16:55 ...
acctg       1490649336845 incr_1-b     incr_1-a       2017-03-27 17:15:37 ...
acctg       1490649255649 incr_1-a     full_1         2017-03-27 17:14:16 ...
acctg       1490649204327 full_1       none           2017-03-27 17:13:25 ...
```

The following additional incremental backups starting with incr_1-b-1, which designates incr_1-b as the parent, results in the forking from that backup into a second line of backups in the chain consisting of full_1, incr_1-a, incr_1-b, incr_1-b-1, incr_1-b-2, and incr_1-b-3 as shown in the following list:

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID     BACKUP NAME   BACKUP PARENT   BACKUP TIME         ...
acctg        1490649791430 incr_1-b-3    incr_1-b-2      2017-03-27 17:23:12 ...
acctg        1490649763929 incr_1-b-2    incr_1-b-1      2017-03-27 17:22:44 ...
acctg        1490649731672 incr_1-b-1    incr_1-b        2017-03-27 17:22:12 ...
acctg        1490649605607 incr_2-b      incr_2-a        2017-03-27 17:20:06 ...
acctg        1490649587702 incr_2-a      full_2          2017-03-27 17:19:48 ...
acctg        1490649528633 full_2        none            2017-03-27 17:18:49 ...
acctg        1490649414316 incr_1-c      incr_1-b        2017-03-27 17:16:55 ...
acctg        1490649336845 incr_1-b      incr_1-a        2017-03-27 17:15:37 ...
acctg        1490649255649 incr_1-a      full_1          2017-03-27 17:14:16 ...
acctg        1490649204327 full_1        none            2017-03-27 17:13:25 ...
```

Restoring an Incremental Backup

Restoring an incremental backup is done with the RESTORE subcommand in the same manner as for restoring a full backup. Specify the backup identifier or backup name of the incremental backup to be restored as shown in the

following example.

```
-bash-4.2$ bart RESTORE -s acctg -p /opt/restore -i incr_1-b
INFO: restoring incremental backup 'incr_1-b' of server 'acctg'
INFO: base backup restored
INFO: archiving is disabled
INFO: permissions set on $PGDATA
INFO: incremental restore completed successfully
```

Restoring incremental backup `incr_1-b` as shown by the preceding example results in the restoration of full backup `full_1`, then incremental backups `incr_1-a` and finally, `incr_1-b`.

2.3 Managing Backups

This section illustrates evaluating, marking, and deleting backups using the MANAGE subcommand with two examples – the first for a redundancy retention policy and the second for a recovery window retention policy. For detailed information about the MANAGE subcommand, see the EDB Postgres Backup and Recovery User Guide.

2.3.1 Using a Redundancy Retention Policy

The following code sample uses a redundancy retention policy to evaluate, mark, and delete backups as shown by the following server configuration:

```
[ACCTG]
host = 127.0.0.1
port = 5444
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 BACKUPS
description = "Accounting"
```

The following list is the set of backups. Note that the last backup in the list has been marked as keep.

```
-bash-4.1$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES    STATUS
acctg        1428768344061   2015-04-11 12:05:46 EDT   5.72 MB       48.00 MB
3            active
acctg        1428684537299   2015-04-10 12:49:00 EDT   5.72 MB       272.00 MB
17           active
acctg        1428589759899   2015-04-09 10:29:27 EDT   5.65 MB       96.00 MB
6            active
acctg        1428502049836   2015-04-08 10:07:30 EDT   55.25 MB      96.00 MB
6            active
acctg        1428422324880   2015-04-07 11:58:45 EDT   54.53 MB      32.00 MB
2            active
acctg        1428355371389   2015-04-06 17:22:53 EDT   5.71 MB       16.00 MB
1            keep
```

Invoke the MANAGE subcommand with the -n option to perform a dry run to observe which active backups would be changed to obsolete according to the retention policy as shown in the following code sample:

```
-bash-4.1$ bart MANAGE -s acctg -n
INFO: processing server 'acctg', backup '1428768344061'
INFO: processing server 'acctg', backup '1428684537299'
INFO: processing server 'acctg', backup '1428589759899'
INFO: processing server 'acctg', backup '1428502049836'
INFO: marking backup '1428502049836' as obsolete
INFO: 6 WAL file(s) marked obsolete
INFO: processing server 'acctg', backup '1428422324880'
INFO: marking backup '1428422324880' as obsolete
INFO: 2 WAL file(s) marked obsolete
INFO: processing server 'acctg', backup '1428355371389'
```

The dry run shows that backups 1428502049836 and 1428422324880 would be marked as obsolete.

Note: A dry run does not change the backup status. The two backups that would be considered obsolete are still

marked as `active`:

```
-bash-4.1$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES    STATUS
acctg        1428768344061   2015-04-11 12:05:46 EDT   5.72 MB       48.00 MB
3            active
acctg        1428684537299   2015-04-10 12:49:00 EDT   5.72 MB       272.00 MB
17           active
acctg        1428589759899   2015-04-09 10:29:27 EDT   5.65 MB       96.00 MB
6            active
acctg        1428502049836   2015-04-08 10:07:30 EDT   55.25 MB      96.00 MB
6            active
acctg        1428422324880   2015-04-07 11:58:45 EDT   54.53 MB      32.00 MB
2            active
acctg        1428355371389   2015-04-06 17:22:53 EDT   5.71 MB       16.00 MB
1            keep
```

Invoke the `MANAGE` subcommand omitting the `-n` option to change and mark the status of the backups as `obsolete`:

```
-bash-4.1$ bart MANAGE -s acctg
INFO: processing server 'acctg', backup '1428768344061'
INFO: processing server 'acctg', backup '1428684537299'
INFO: processing server 'acctg', backup '1428589759899'
INFO: processing server 'acctg', backup '1428502049836'
INFO: marking backup '1428502049836' as obsolete
INFO: 6 WAL file(s) marked obsolete
INFO: processing server 'acctg', backup '1428422324880'
INFO: marking backup '1428422324880' as obsolete
INFO: 2 WAL file(s) marked obsolete
INFO: processing server 'acctg', backup '1428355371389'
```

The obsolete backups can be observed in a number of ways. Use the `MANAGE` subcommand with the `-l` option to list the `obsolete` backups:

```
-bash-4.1$ bart MANAGE -s acctg -l
INFO: 6 WAL file(s) will be removed
SERVER NAME: acctg
BACKUP ID: 1428502049836
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-08 10:07:30 EDT
BACKUP SIZE: 55.25 MB
WAL FILE(s): 6
WAL FILE: 000000010000000100000003
WAL FILE: 000000010000000100000002
WAL FILE: 000000010000000100000001
WAL FILE: 000000010000000100000000
WAL FILE: 0000000100000000000000E3
WAL FILE: 0000000100000000000000E2
INFO: 2 WAL file(s) will be removed
SERVER NAME: acctg
BACKUP ID: 1428422324880
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-07 11:58:45 EDT
BACKUP SIZE: 54.53 MB
WAL FILE(s): 2
WAL FILE: 0000000100000000000000E1
```

```
WAL FILE: 0000000100000000000000E0
```

The STATUS field of the SHOW-BACKUPS subcommand displays the current status:

```
-bash-4.1$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID      BACKUP TIME              BACKUP SIZE   WAL(s) SIZE
WAL FILES    STATUS
acctg        1428768344061  2015-04-11 12:05:46 EDT  5.72 MB       48.00 MB
3            active
acctg        1428684537299  2015-04-10 12:49:00 EDT  5.72 MB       272.00 MB
17           active
acctg        1428589759899  2015-04-09 10:29:27 EDT  5.65 MB       96.00 MB
6            active
acctg        1428502049836  2015-04-08 10:07:30 EDT  55.25 MB      96.00 MB
6            obsolete
acctg        1428422324880  2015-04-07 11:58:45 EDT  54.53 MB      32.00 MB
2            obsolete
acctg        1428355371389  2015-04-06 17:22:53 EDT  5.71 MB       16.00 MB
1            keep
```

The details of an individual backup can be displayed using the SHOW-BACKUPS subcommand with the -t option. Note the status in the BACKUP STATUS field.

```
-bash-4.1$ bart SHOW-BACKUPS -s acctg -i 1428502049836 -t
SERVER NAME : acctg
BACKUP ID : 1428502049836
BACKUP NAME : none
BACKUP STATUS : obsolete
BACKUP TIME : 2015-04-08 10:07:30 EDT
BACKUP SIZE : 55.25 MB
WAL(S) SIZE : 96.00 MB
NO. OF WALS : 6
FIRST WAL FILE : 0000000100000000000000E2
CREATION TIME : 2015-04-08 10:07:30 EDT
LAST WAL FILE : 000000010000000100000003
CREATION TIME : 2015-04-09 10:25:46 EDT
```

Use the MANAGE subcommand with the -d option to physically delete the obsolete backups including the unneeded WAL files.

```
-bash-4.1$ bart MANAGE -s acctg -d
INFO: removing all obsolete backups of server 'acctg'
INFO: removing obsolete backup '1428502049836'
INFO: 6 WAL file(s) will be removed
INFO: removing WAL file '000000010000000100000003'
INFO: removing WAL file '000000010000000100000002'
INFO: removing WAL file '000000010000000100000001'
INFO: removing WAL file '000000010000000100000000'
INFO: removing WAL file '0000000100000000000000E3'
INFO: removing WAL file '0000000100000000000000E2'
INFO: removing obsolete backup '1428422324880'
INFO: 2 WAL file(s) will be removed
INFO: removing WAL file '0000000100000000000000E1'
INFO: removing WAL file '0000000100000000000000E0'
```

The SHOW-BACKUPS subcommand now displays the remaining backups marked as active or keep:

```
-bash-4.1$ bart SHOW-BACKUPS -s acctg
SERVER NAME    BACKUP ID         BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES      STATUS
acctg          1428768344061     2015-04-11 12:05:46 EDT   5.72 MB       48.00 MB
3              active
acctg          1428684537299     2015-04-10 12:49:00 EDT   5.72 MB       272.00 MB
17             active
acctg          1428589759899     2015-04-09 10:29:27 EDT   5.65 MB       96.00 MB
6              active
acctg          1428355371389     2015-04-06 17:22:53 EDT   5.71 MB       16.00 MB
1              keep
```

2.3.2 Using a Recovery Window Retention Policy

This section illustrates the evaluation, marking, and deletion of backup using a recovery window retention policy. To use the recovery window retention policy, set the `retention_policy` parameter to the desired length of time for the recovery window.

This section provides examples of the following:

- How to view the calculated recovery window.
- How to evaluate, mark, and delete backup using a recovery window retention policy.

2.3.2.1 Viewing the Recovery Window

You can view the actual, calculated recovery window by invoking any of the following subcommands:

- `MANAGE` subcommand in debug mode (along with the `-n` option).
- `SHOW-SERVERS` subcommand.

2.3.2.1.1 Viewing the Recovery Window Using the Manage Subcommand

By invoking BART in debug mode and the `MANAGE` subcommand with the `-n` option, the time length of the recovery window is calculated based on the `retention_policy` setting and the current date/time.

For example, using the following `retention_policy` settings:

```
[ACCTG]

host = 127.0.0.1
port = 5444
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 DAYS
backup-name = acctg_%year-%month-%dayT%hour:%minute:%second
description = "Accounting"

[DEV]

host = 127.0.0.1
port = 5445
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 WEEKS
description = "Development"

[HR]

host = 127.0.0.1
port = 5432
user = postgres
retention_policy = 3 MONTHS
description = "Human Resources"
```

If the `MANAGE` subcommand is invoked in debug mode along with the `-n` option on 2015-04-17, the following results are displayed:

```
-bash-4.1$ bart -d MANAGE -n
DEBUG: Server: acctg, Now: 2015-04-17 16:34:03 EDT, RetentionWindow:
259200 (secs) ==> 72 hour(s)
DEBUG: Server: dev, Now: 2015-04-17 16:34:03 EDT, RetentionWindow:
1814400 (secs) ==> 504 hour(s)
DEBUG: Server: hr, Now: 2015-04-17 16:34:03 EDT, RetentionWindow:
7776000 (secs) ==> 2160 hour(s)
```

For server `acctg`, 72 hours translates to a recovery window of 3 days.

For server `dev`, 504 hours translates to a recovery window of 21 days (3 weeks).

For server `hr`, 2160 hours translates to a recovery window of 90 days (3 months).

For a setting of `<max_number> MONTHS`, the calculated total number of days for the recovery window is dependent upon the actual number of days in the preceding months from the current date/time. Thus, `<max_number> MONTHS` is not always exactly equivalent to `<max_number> x 30 DAYS`. (For example, if the current date/time is in the month of March, a 1-month recovery window would be equivalent to only 28 days because the preceding month is February. Thus, for a current date of March 31, a 1-month recovery window would start on March 3.) However, the typical result is that the day of the month of the starting recovery window boundary will be the same day of the month of when the `MANAGE` subcommand is invoked.

2.3.2.1.2 Viewing the Recovery Window Using the Show-Servers Subcommand

This section provides an example of viewing the recovery window using the `SHOW-SERVERS` subcommand; the `RETENTION POLICY` field displays the start of the recovery window.

In the following code sample, the recovery window retention policy setting considers the backups taken within a 3-day recovery window as the active backups.

```
[ACCTG]
host = 127.0.0.1
port = 5444
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 DAYS
description = "Accounting"
```

The start of the 3-day recovery window displayed in the `RETENTION POLICY` field is `2015-04-07 14:57:36 EDT` when the `SHOW-SERVERS` subcommand is invoked on `2015-04-10`.

At this current point in time, backups taken on or after `2015-04-07 14:57:36 EDT` would be considered active. Backups taken prior to `2015-04-07 14:57:36 EDT` would be considered obsolete except for backups marked as `keep`.

```
-bash-4.1$ date
Fri Apr 10 14:57:33 EDT 2015
-bash-4.1$
-bash-4.1$ bart SHOW-SERVERS -s acctg
SERVER NAME        : acctg
HOST NAME          : 127.0.0.1
USER NAME          : enterprisedb
PORT               : 5444
REMOTE HOST        :
RETENTION POLICY   : 2015-04-07 14:57:36 EDT
DISK UTILIZATION   : 824.77 MB
NUMBER OF ARCHIVES : 37
```

```
ARCHIVE PATH       : /opt/backup/acctg/archived_wals
ARCHIVE COMMAND    : cp %p /opt/backup/acctg/archived_wals/%f
XLOG METHOD        : fetch
WAL COMPRESSION    : disabled
TABLESPACE PATH(s) :
DESCRIPTION : "Accounting"
```

In the following code sample, the recovery window retention policy setting considers the backups taken within a 3-week recovery window as the `active` backups.

```
[DEV]
host = 127.0.0.1
port = 5445
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 WEEKS
description = "Development"
```

The start of the 3-week recovery window displayed in the `RETENTION POLICY` field is `2015-03-20 14:59:42 EDT` when the `SHOW-SERVERS` subcommand is invoked on `2015-04-10`.

At this current point in time, backups taken on or after `2015-03-20 14:59:42 EDT` would be considered `active`. Backups taken prior to `2015-03-20 14:59:42 EDT` would be considered `obsolete` except for backups marked as `keep`.

```
-bash-4.1$ date
Fri Apr 10 14:59:39 EDT 2015
-bash-4.1$
-bash-4.1$ bart SHOW-SERVERS -s dev
SERVER NAME : dev
HOST NAME : 127.0.0.1
USER NAME : enterprisedb
PORT : 5445
REMOTE HOST :
RETENTION POLICY : 2015-03-20 14:59:42 EDT
DISK UTILIZATION : 434.53 MB
NUMBER OF ARCHIVES : 22
ARCHIVE PATH : /opt/backup/dev/archived_wals
ARCHIVE COMMAND : cp %p /opt/backup/dev/archived_wals/%f
XLOG METHOD : fetch
WAL COMPRESSION : disabled
TABLESPACE PATH(s) :
DESCRIPTION : "Development"
```

In the following code sample, the recovery window retention policy setting considers the backups taken within a 3-month recovery window as the `active` backups.

```
[HR]
host = 127.0.0.1
port = 5432
user = postgres
retention_policy = 3 MONTHS
description = "Human Resources"
```

The start of the 3-month recovery window displayed in the `RETENTION POLICY` field is `2015-01-10 14:04:23 EST` when the `SHOW-SERVERS` subcommand is invoked on `2015-04-10`.

At this current point in time, backups taken on or after `2015-01-10 14:04:23 EST` would be considered

active. Backups taken prior to 2015-01-10 14:04:23 EST would be considered obsolete, except for backups marked as keep.

```
-bash-4.1$ date
Fri Apr 10 15:04:19 EDT 2015
-bash-4.1$
-bash-4.1$ bart SHOW-SERVERS -s hr
SERVER NAME : hr
HOST NAME : 127.0.0.1
USER NAME : postgres
PORT : 5432
REMOTE HOST :
RETENTION POLICY : 2015-01-10 14:04:23 EST
DISK UTILIZATION : 480.76 MB
NUMBER OF ARCHIVES : 26
ARCHIVE PATH : /opt/backup/hr/archived_wals
ARCHIVE COMMAND : scp %p
enterprisedb@192.168.2.22:/opt/backup/hr/archived_wals/%f
XLOG METHOD : fetch
WAL COMPRESSION : disabled
TABLESPACE PATH(s) :
DESCRIPTION : "Human Resources"
```

2.3.2.2 Evaluating, Marking, and Deleting Backup Using a Recovery Window Retention Policy

The following code sample uses a recovery window retention policy to evaluate, mark, and delete backups as shown by the following server configuration:

```
[DEV]
host = 127.0.0.1
port = 5445
user = enterprisedb
archive_command = 'cp %p %a/%f'
retention_policy = 3 DAYS
description = "Development"
```

The following is the current set of backups. Note that the last backup in the list has been marked as keep.

```
-bash-4.1$ bart SHOW-BACKUPS -s dev
SERVER NAME    BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
WAL FILES      STATUS
dev            1428933278236   2015-04-13 09:54:40 EDT   5.65 MB       16.00 MB
1              active
dev            1428862187757   2015-04-12 14:09:50 EDT   5.65 MB       32.00 MB
2              active
dev            1428768351638   2015-04-11 12:05:54 EDT   5.65 MB       32.00 MB
2              active
dev            1428684544008   2015-04-10 12:49:06 EDT   5.65 MB       224.00 MB
14             active
dev            1428590536488   2015-04-09 10:42:18 EDT   5.65 MB       48.00 MB
3              active
dev            1428502171990   2015-04-08 10:09:34 EDT   5.65 MB       80.00 MB
5              keep
```

The current date and time is 2015-04-13 16:46:35 EDT as shown below:

```
-bash-4.1$ date
Mon Apr 13 16:46:35 EDT 2015
```

Thus, a 3-day recovery window would evaluate backups prior to `2015-04-10 16:46:35 EDT` as `obsolete` except for those marked as `keep`.

Invoke the `MANAGE` subcommand with the `-n` option to perform a dry run to observe which active backups would be changed to `obsolete` according to the retention policy.

```
-bash-4.1$ bart MANAGE -s dev -n
INFO: processing server 'dev', backup '1428933278236'
INFO: processing server 'dev', backup '1428862187757'
INFO: processing server 'dev', backup '1428768351638'
INFO: processing server 'dev', backup '1428684544008'
INFO: marking backup '1428684544008' as obsolete
INFO: 14 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: processing server 'dev', backup '1428590536488'
INFO: marking backup '1428590536488' as obsolete
INFO: 3 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: processing server 'dev', backup '1428502171990'
```

The dry run shows that backups `1428684544008` and `1428590536488` would be marked as `obsolete`.

Also note that a dry run does not change the backup status. The two backups that would be considered obsolete are still marked as `active`:

```
-bash-4.1$ bart SHOW-BACKUPS -s dev\
  SERVER NAME  BACKUP ID       BACKUP TIME               BACKUP SIZE   WAL(s) SIZE
  WAL FILES    STATUS
  dev          1428933278236   2015-04-13 09:54:40 EDT   5.65 MB       16.00 MB
   1           active
  dev          1428862187757   2015-04-12 14:09:50 EDT   5.65 MB       32.00 MB
  2            active
  dev          1428768351638   2015-04-11 12:05:54 EDT   5.65 MB       32.00 MB
  2            active
  dev          1428684544008   2015-04-10 12:49:06 EDT   5.65 MB       224.00 MB
  14           active
  dev          1428590536488   2015-04-09 10:42:18 EDT   5.65 MB       48.00 MB
  3            active
  dev          1428502171990   2015-04-08 10:09:34 EDT   5.65 MB       80.00 MB
  5            keep
```

Invoke the `MANAGE` subcommand omitting the `-n` option to change and mark the status of the backups as `obsolete`:

```
-bash-4.1$ bart MANAGE -s dev
INFO: processing server 'dev', backup '1428933278236'
INFO: processing server 'dev', backup '1428862187757'
INFO: processing server 'dev', backup '1428768351638'
INFO: processing server 'dev', backup '1428684544008'
INFO: marking backup '1428684544008' as obsolete
INFO: 14 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: processing server 'dev', backup '1428590536488'
INFO: marking backup '1428590536488' as obsolete
INFO: 3 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
```

```
INFO: processing server 'dev', backup '1428502171990'
```

The obsolete backups can be observed in a number of ways. Use the `MANAGE` subcommand with the `-l` option to list the `obsolete` backups:

```
-bash-4.1$ bart MANAGE -s dev -l
INFO: 14 WAL file(s) will be removed
INFO: 1 Unused WAL file(s) will be removed
SERVER NAME: dev
BACKUP ID: 1428684544008
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-10 12:49:06 EDT
BACKUP SIZE: 5.65 MB
WAL FILE(s): 14
UNUSED WAL FILE(s): 1
WAL FILE: 00000001000000000000002E
WAL FILE: 00000001000000000000002D
WAL FILE: 00000001000000000000002C
WAL FILE: 00000001000000000000002B
WAL FILE: 00000001000000000000002A
WAL FILE: 000000010000000000000029
WAL FILE: 000000010000000000000028
WAL FILE: 000000010000000000000027
WAL FILE: 000000010000000000000026
WAL FILE: 000000010000000000000025
WAL FILE: 000000010000000000000024
WAL FILE: 000000010000000000000023
WAL FILE: 000000010000000000000022
WAL FILE: 000000010000000000000021
UNUSED WAL FILE: 00000001000000000000000F.00000028
INFO: 3 WAL file(s) will be removed
INFO: 1 Unused WAL file(s) will be removed
SERVER NAME: dev
BACKUP ID: 1428590536488
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-09 10:42:18 EDT\
BACKUP SIZE: 5.65 MB
WAL FILE(s): 3
UNUSED WAL FILE(s): 1
WAL FILE: 000000010000000000000020
WAL FILE: 00000001000000000000001F
WAL FILE: 00000001000000000000001E
UNUSED WAL FILE: 00000001000000000000000F.00000028
```

The `STATUS` field of the `SHOW-BACKUPS` subcommand displays the current status:

```
-bash-4.1$ bart SHOW-BACKUPS -s dev
SERVER NAME  BACKUP ID       BACKUP TIME               BACKUP SIZE  WAL(s) SIZE
WAL FILES    STATUS
dev          1428933278236   2015-04-13 09:54:40 EDT   5.65 MB     16.00 MB
1            active
dev          1428862187757   2015-04-12 14:09:50 EDT   5.65 MB     32.00 MB
2            active
dev          1428768351638   2015-04-11 12:05:54 EDT   5.65 MB     32.00 MB
2            active
dev          1428684544008   2015-04-10 12:49:06 EDT   5.65 MB     224.00 MB
14           obsolete
dev          1428590536488   2015-04-09 10:42:18 EDT   5.65 MB     48.00 MB
```

```
3            obsolete
dev          1428502171990   2015-04-08 10:09:34 EDT   5.65 MB     80.00 MB
5            keep
```

The details of an individual backup can be displayed using the SHOW-BACKUPS subcommand with the -t option. Note the status in the BACKUP STATUS field.

```
-bash-4.1$ bart SHOW-BACKUPS -s dev -i 1428684544008 -t
SERVER NAME    : dev
BACKUP ID      : 1428684544008
BACKUP NAME    : none
BACKUP STATUS  : obsolete
BACKUP TIME    : 2015-04-10 12:49:06 EDT
BACKUP SIZE    : 5.65 MB
WAL(S) SIZE    : 224.00 MB
NO. OF WALS    : 14
FIRST WAL FILE : 000000010000000000000021
CREATION TIME  : 2015-04-10 12:49:06 EDT
LAST WAL FILE  : 00000001000000000000002E
CREATION TIME  : 2015-04-11 12:02:15 EDT
```

Use the MANAGE subcommand with the -d option to physically delete the obsolete backups including the unneeded WAL files.

```
-bash-4.1$ bart MANAGE -s dev -d
INFO: removing all obsolete backups of server 'dev'
INFO: removing obsolete backup '1428684544008'
INFO: 14 WAL file(s) will be removed
INFO: 1 Unused WAL file(s) will be removed
INFO: removing WAL file '00000001000000000000002E'
INFO: removing WAL file '00000001000000000000002D'
INFO: removing WAL file '00000001000000000000002C'
INFO: removing WAL file '00000001000000000000002B'
INFO: removing WAL file '00000001000000000000002A'
INFO: removing WAL file '000000010000000000000029'
INFO: removing WAL file '000000010000000000000028'
INFO: removing WAL file '000000010000000000000027'
INFO: removing WAL file '000000010000000000000026'
INFO: removing WAL file '000000010000000000000025'
INFO: removing WAL file '000000010000000000000024'
INFO: removing WAL file '000000010000000000000023'
INFO: removing WAL file '000000010000000000000022'
INFO: removing WAL file '000000010000000000000021'
INFO: removing (unused) WAL file '00000001000000000000000F.00000028'
INFO: removing obsolete backup '1428590536488'
INFO: 3 WAL file(s) will be removed
INFO: removing WAL file '000000010000000000000020'
INFO: removing WAL file '00000001000000000000001F'
INFO: removing WAL file '00000001000000000000001E'
```

The SHOW-BACKUPS subcommand now displays the remaining backups marked as active or keep:

```
-bash-4.1$ bart SHOW-BACKUPS -s dev
SERVER NAME  BACKUP ID      BACKUP TIME               BACKUP SIZE     WAL(s) SIZE
WAL FILES    STATUS
dev          1428933278236   2015-04-13 09:54:40 EDT   5.65 MB         16.00 MB
1            active
dev          1428862187757   2015-04-12 14:09:50 EDT   5.65 MB         32.00 MB
```

```
2             active
dev           1428768351638   2015-04-11 12:05:54 EDT   5.65 MB         32.00 MB
2             active
dev           1428502171990   2015-04-08 10:09:34 EDT   5.65 MB         80.00 MB
5             keep
```

2.4 Managing Incremental Backups

This section illustrates evaluating, marking, and deleting incremental backups using the `MANAGE` and `DELETE` subcommands with two examples – the first for a redundancy retention policy and the second for a recovery window retention policy. For detailed information about the `MANAGE` and `DELETE` subcommands, as well as the redundancy retention and recovery window retention policy, see the EDB Postgres Backup and Recovery User Guide.

- *Using a Redundancy Retention Policy* provides an example of using the `MANAGE` and `DELETE` subcommands when a 3 backup redundancy retention policy is in effect.
- *Using a Recovery Window Retention Policy* provides an example of using the `MANAGE` and `DELETE` subcommands when a 1-day recovery window retention policy is in effect.

2.4.1 Using a Redundancy Retention Policy

The following code samples show using the MANAGE and DELETE subcommands to evaluate, mark, and delete incremental backups when a 3 backup redundancy retention policy is in effect. The example uses the following server configuration:

```
[ACCTG]

host = 192.168.2.24
port = 5445
user = enterprisedb
cluster_owner = enterprisedb
remote_host = enterprisedb@192.168.2.24
allow_incremental_backups = enabled
retention_policy = 3 BACKUPS
description = "Accounting"
```

The example uses the following set of backups. (In these code samples, some columns have been omitted from the SHOW-BACKUPS output in order to display the relevant information in a more observable manner).

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID       ... BACKUP PARENT   BACKUP TIME              ... STATUS
acctg        1481749696905   ... 1481749673603   2016-12-14 16:08:17 EST ... active
acctg        1481749673603   ... 1481749651927   2016-12-14 16:07:53 EST ... active
acctg        1481749651927   ... 1481749619582   2016-12-14 16:07:32 EST ... active
acctg        1481749619582   ... none            2016-12-14 16:07:00 EST ... active
```

There is one backup chain. The first backup is the initial full backup.

Backup chain: 1481749619582 => 1481749651927 => 1481749673603 => 1481749696905

The MANAGE subcommand is invoked as shown by the following:

```
-bash-4.2$ bart MANAGE -s acctg
INFO: processing server 'acctg', backup '1481749619582'
INFO: 2 Unused WAL file(s) present
INFO: 4 Unused file(s) (WALs included) present, use 'MANAGE -l' for the
list
```

The following code sample shows the resulting status of the backups:

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME    BACKUP ID     ... BACKUP PARENT   BACKUP TIME              ... STATUS
acctg          1481749696905 ... 1481749673603   2016-12-14 16:08:17 EST ... active
acctg          1481749673603 ... 1481749651927   2016-12-14 16:07:53 EST ... active
acctg          1481749651927 ... 1481749619582   2016-12-14 16:07:32 EST ... active
acctg          1481749619582 ... none            2016-12-14 16:07:00 EST ... active
```

The status remains active for all backups. Even though the total number of backups exceeds the 3 backup redundancy retention policy, it is only the total number of full backups that is used to determine if the redundancy retention policy has been exceeded. Additional full backups are added including a second backup chain. The following example shows the resulting list of backups:

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID     ... BACKUP PARENT     BACKUP TIME              ... STATUS
acctg        1481750365397 ... none              2016-12-14 16:19:26 EST ... active
acctg        1481750098924 ... 1481749997807     2016-12-14 16:14:59 EST ... active
acctg        1481749997807 ... none              2016-12-14 16:13:18 EST ... active
```

```
acctg        1481749992003 ... none              2016-12-14 16:13:12 EST ... active
acctg        1481749696905 ... 1481749673603     2016-12-14 16:08:17 EST ... active
acctg        1481749673603 ... 1481749651927     2016-12-14 16:07:53 EST ... active
acctg        1481749651927 ... 1481749619582     2016-12-14 16:07:32 EST ... active
acctg        1481749619582 ... none              2016-12-14 16:07:00 EST ... active
```

Second backup chain: `1481749997807 => 1481750098924`

The `MANAGE` subcommand is invoked, but now with a total of four active full backups.

```
-bash-4.2$ bart MANAGE -s acctg
INFO: processing server 'acctg', backup '1481750365397'
INFO: processing server 'acctg', backup '1481749997807'
INFO: processing server 'acctg', backup '1481749992003'
INFO: processing server 'acctg', backup '1481749619582'
INFO: marking backup '1481749619582' as obsolete
INFO: 3 incremental(s) of backup '1481749619582' will be marked obsolete
INFO: marking incremental backup '1481749696905' as obsolete
INFO: marking incremental backup '1481749673603' as obsolete
INFO: marking incremental backup '1481749651927' as obsolete
INFO: 4 WAL file(s) marked obsolete
INFO: 2 Unused WAL file(s) present
INFO: 4 Unused file(s) (WALs included) present, use 'MANAGE -l' for the
list
```

The oldest full backup and its chain of incremental backups are now marked as obsolete.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID        ... BACKUP PARENT   BACKUP TIME             ... STATUS
acctg        1481750365397    ... none            2016-12-14 16:19:26 EST ... active
acctg        1481750098924    ... 1481749997807   2016-12-14 16:14:59 EST ... active
acctg        1481749997807    ... none            2016-12-14 16:13:18 EST ... active
acctg        1481749992003    ... none            2016-12-14 16:13:12 EST ... active
acctg        1481749696905    ... 1481749673603   2016-12-14 16:08:17 EST ... obsolete
acctg        1481749673603    ... 1481749651927   2016-12-14 16:07:53 EST ... obsolete
acctg        1481749651927    ... 1481749619582   2016-12-14 16:07:32 EST ... obsolete
acctg        1481749619582    ... none            2016-12-14 16:07:00 EST ... obsolete
```

Invoking the `MANAGE` subcommand with the `-d` option deletes the entire obsolete backup chain.

```
-bash-4.2$ bart MANAGE -s acctg -d
INFO: removing all obsolete backups of server 'acctg'
INFO: removing obsolete backup '1481749619582'
INFO: 4 WAL file(s) will be removed
INFO: 3 incremental(s) of backup '1481749619582' will be removed
INFO: removing obsolete incremental backup '1481749696905'
INFO: removing obsolete incremental backup '1481749673603'
INFO: removing obsolete incremental backup '1481749651927'
INFO: removing WAL file '000000010000000100000000'
INFO: removing WAL file '0000000100000000000000FF'
INFO: removing WAL file '0000000100000000000000FE'
INFO: removing WAL file '0000000100000000000000FD'
INFO: 16 Unused file(s) will be removed
INFO: removing (unused) file '000000010000000100000004.00000028.backup'
.
.
.
INFO: removing (unused) file
```

```
'0000000100000000FB00002800000000FC000000.mbm'
```

The following code sample shows the remaining full backups and the second backup chain.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID       ... BACKUP PARENT   BACKUP TIME             ... STATUS
acctg        1481750365397   ... none            2016-12-14 16:19:26 EST ... active
acctg        1481750098924   ... 1481749997807   2016-12-14 16:14:59 EST ... active
acctg        1481749997807   ... none            2016-12-14 16:13:18 EST ... active
acctg        1481749992003   ... none            2016-12-14 16:13:12 EST ... active
```

2.4.2 Using a Recovery Window Retention Policy

The following example demonstrates using the MANAGE and DELETE subcommands to evaluate, mark, and delete incremental backups when a 1-day recovery window retention policy is in effect. The example uses the following server configuration:

```
[ACCTG]

host = 192.168.2.24
port = 5445
user = enterprisedb
cluster_owner = enterprisedb
remote_host = enterprisedb@192.168.2.24
allow_incremental_backups = enabled
retention_policy = 1 DAYS
description = "Accounting"
```

The example uses the following set of backups. In the code samples, some columns have been omitted from the SHOW-BACKUPS output in order to display the relevant information in a more observable manner.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME BACKUP ID ... BACKUP PARENT BACKUP TIME ... STATUS
acctg 1481559303348 ... 1481554203288 2016-12-12 11:15:03 EST ... active
acctg 1481559014359 ... 1481554802918 2016-12-12 11:10:14 EST ... active
acctg 1481554802918 ... 1481553914533 2016-12-12 10:00:03 EST ... active
acctg 1481554203288 ... 1481553651165 2016-12-12 09:50:03 EST ... active
acctg 1481553914533 ... 1481553088053 2016-12-12 09:45:14 EST ... active
acctg 1481553651165 ... none 2016-12-12 09:40:51 EST ... active
acctg 1481553088053 ... 1481552078404 2016-12-12 09:31:28 EST ... active
acctg 1481552078404 ... none 2016-12-12 09:14:39 EST ... active
```

There are two backup chains. In each of the following chains, the first backup is the initial full backup.

First backup chain: 1481552078404 => 1481553088053 => 1481553914533 => 1481554802918 => 1481559014359

Second backup chain: 1481553651165 => 1481554203288 => 1481559303348

The MANAGE subcommand is invoked when the first full backup 1481552078404 falls out of the recovery window. When the MANAGE subcommand is invoked, it is 2016-12-13 09:20:03 EST, thus making the start of the 1-day recovery window at 2016-12-12 09:20:03 EST exactly one day earlier. This backup was taken at 2016-12-12 09:14:39 EST, which is about 5 ½ minutes before the start of the recovery window, thus making the backup obsolete.

```
-bash-4.2$ date
Tue Dec 13 09:20:03 EST 2016
-bash-4.2$ bart MANAGE -s acctg
INFO: processing server 'acctg', backup '1481553651165'
INFO: processing server 'acctg', backup '1481552078404'
INFO: marking backup '1481552078404' as obsolete
INFO: 4 incremental(s) of backup '1481552078404' will be marked obsolete
INFO: marking incremental backup '1481559014359' as obsolete
INFO: marking incremental backup '1481554802918' as obsolete
INFO: marking incremental backup '1481553914533' as obsolete
INFO: marking incremental backup '1481553088053' as obsolete
INFO: 7 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: 2 Unused file(s) (WALs included) present, use 'MANAGE -l' for the list
```

The incremental backup date and time are within the recovery window since they were taken after the start of the recovery window of `2016-12-12 09:20:03 EST`, but all backups in the chain are marked as `obsolete`.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg\
SERVER NAME      BACKUP ID     ... BACKUP PARENT       BACKUP TIME
... STATUS
acctg            1481559303348 ... 1481554203288       2016-12-12 11:15:03 EST
... active
acctg            1481559014359 ... 1481554802918       2016-12-12 11:10:14 EST
... obsolete
acctg            1481554802918 ... 1481553914533       2016-12-12 10:00:03 EST
... obsolete
acctg            1481554203288 ... 1481553651165       2016-12-12 09:50:03 EST
... active
acctg            1481553914533 ... 1481553088053       2016-12-12 09:45:14 EST
... obsolete
acctg            1481553651165 ... none                2016-12-12 09:40:51 EST
... active
acctg            1481553088053 ... 1481552078404       2016-12-12 09:31:28 EST
... obsolete
acctg            1481552078404 ... none                2016-12-12 09:14:39 EST
... obsolete
```

The following code sample shows how the entire backup chain is changed back to active status by invoking the `MANAGE` subcommand with the `-c nokeep` option on the full backup of the chain.

```
-bash-4.2$ bart MANAGE -s acctg -c nokeep -i 1481552078404
INFO: changing status of backup '1481552078404' of server 'acctg' from
'obsolete' to 'active'
INFO: status of 4 incremental(s) of backup '1481552078404' will be
changed
INFO: changing status of incremental backup '1481559014359' of server
'acctg' from 'obsolete' to 'active'
INFO: changing status of incremental backup '1481554802918' of server
'acctg' from 'obsolete' to 'active'
INFO: changing status of incremental backup '1481553914533' of server
'acctg' from 'obsolete' to 'active'
INFO: changing status of incremental backup '1481553088053' of server
'acctg' from 'obsolete' to 'active'
INFO: 7 WAL file(s) changed
```

The backup chain has now been reset to active status.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME    BACKUP ID        ... BACKUP PARENT   BACKUP TIME             ... STATUS
acctg          1481559303348    ... 1481554203288   2016-12-12 11:15:03 EST ... active
acctg          1481559014359    ... 1481554802918   2016-12-12 11:10:14 EST ... active
acctg          1481554802918    ... 1481553914533   2016-12-12 10:00:03 EST ... active
acctg          1481554203288    ... 1481553651165   2016-12-12 09:50:03 EST ... active
acctg          1481553914533    ... 1481553088053   2016-12-12 09:45:14 EST ... active
acctg          1481553651165    ... none            2016-12-12 09:40:51 EST ... active
acctg          1481553088053    ... 1481552078404   2016-12-12 09:31:28 EST ... active
acctg          1481552078404    ... none            2016-12-12 09:14:39 EST ... active
```

The following code sample shows usage of the `DELETE` subcommand on an incremental backup. The specified incremental backup `1481554802918` in the first backup chain as well as its successive incremental backup `1481559014359` are deleted.

```
-bash-4.2$ bart DELETE -s acctg -i 1481554802918
INFO: deleting backup '1481554802918' of server 'acctg'
INFO: deleting backup '1481554802918'
INFO: 1 incremental backup(s) will be deleted
INFO: deleting incremental backup '1481559014359'
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or
will be marked unused
INFO: 2 Unused file(s) will be removed
INFO: removing (unused) file '000000010000000000000BA'
INFO: removing (unused) file
'000000010000000BA00002800000000BB000000.mbm'
INFO: backup(s) deleted
```

The results show that incremental backup `1481554802918` as well as its successive backup `1481559014359` are no longer listed by the `SHOW-BACKUPS` subcommand.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID       ... BACKUP PARENT   BACKUP TIME               ... STATUS
acctg        1481559303348   ... 1481554203288   2016-12-12 11:15:03 EST   ... active
acctg        1481554203288   ... 1481553651165   2016-12-12 09:50:03 EST   ... active
acctg        1481553914533   ... 1481553088053   2016-12-12 09:45:14 EST   ... active
acctg        1481553651165   ... none            2016-12-12 09:40:51 EST   ... active
acctg        1481553088053   ... 1481552078404   2016-12-12 09:31:28 EST   ... active
acctg        1481552078404   ... none            2016-12-12 09:14:39 EST   ... active
```

The `MANAGE` subcommand is invoked again. This time both backup chains are marked `obsolete` since the full backups of both chains fall out of the start of the recovery window, which is now `2016-12-12 09:55:03 EST`.

```
-bash-4.2$ date
Tue Dec 13 09:55:03 EST 2016
-bash-4.2$ bart MANAGE -s acctg
INFO: processing server 'acctg', backup '1481553651165'
INFO: marking backup '1481553651165' as obsolete
INFO: 2 incremental(s) of backup '1481553651165' will be marked obsolete
INFO: marking incremental backup '1481559303348' as obsolete
INFO: marking incremental backup '1481554203288' as obsolete
INFO: 38 WAL file(s) marked obsolete
INFO: processing server 'acctg', backup '1481552078404'
INFO: marking backup '1481552078404' as obsolete
INFO: 2 incremental(s) of backup '1481552078404' will be marked obsolete
INFO: marking incremental backup '1481553914533' as obsolete
INFO: marking incremental backup '1481553088053' as obsolete
INFO: 7 WAL file(s) marked obsolete
```

The following code sample shows both backup chains marked as `obsolete`.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME    BACKUP ID       ... BACKUP PARENT   BACKUP TIME
... STATUS
acctg          1481559303348   ... 1481554203288   2016-12-12 11:15:03 EST
... obsolete
acctg          1481554203288   ... 1481553651165   2016-12-12 09:50:03 EST
... obsolete
acctg          1481553914533   ... 1481553088053   2016-12-12 09:45:14 EST
... obsolete
acctg          1481553651165   ... none            2016-12-12 09:40:51 EST
... obsolete
acctg          1481553088053   ... 1481552078404   2016-12-12 09:31:28 EST
```

```
... obsolete
acctg          1481552078404   ... none            2016-12-12 09:14:39 EST
... obsolete
```

The following code sample shows usage of the `MANAGE` subcommand with the `-c keep` option to keep a backup chain indefinitely. The `MANAGE` subcommand with the `-c keep` option must specify the backup identifier or backup name of the full backup of the chain, and not any incremental backup.

```
-bash-4.2$ bart MANAGE -s acctg -c keep -i 1481553651165
INFO: changing status of backup '1481553651165' of server 'acctg' from
'obsolete' to 'keep'
INFO: status of 2 incremental(s) of backup '1481553651165' will be
changed
INFO: changing status of incremental backup '1481559303348' of server
'acctg' from 'obsolete' to 'keep'
INFO: changing status of incremental backup '1481554203288' of server
'acctg' from 'obsolete' to 'keep'
INFO: 38 WAL file(s) changed
```

The following now displays the full backup `1481553651165` of the backup chain and its successive incremental backups `1481554203288` and `1481559303348`, changed to `keep` status.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME  BACKUP ID       ... BACKUP PARENT   BACKUP TIME
... STATUS
acctg        1481559303348   ... 1481554203288   2016-12-12 11:15:03 EST
... keep
acctg        1481554203288   ... 1481553651165   2016-12-12 09:50:03 EST
... keep
acctg        1481553914533   ... 1481553088053   2016-12-12 09:45:14 EST
... obsolete
acctg        1481553651165   ... none            2016-12-12 09:40:51 EST
... keep
acctg        1481553088053   ... 1481552078404   2016-12-12 09:31:28 EST
... obsolete
acctg        1481552078404   ... none            2016-12-12 09:14:39 EST
... obsolete
```

Finally, the `MANAGE` subcommand with the `-d` option is used to delete the `obsolete` backup chain.

```
-bash-4.2$ bart MANAGE -s acctg -d
INFO: removing all obsolete backups of server 'acctg'
INFO: removing obsolete backup '1481552078404'
INFO: 7 WAL file(s) will be removed
INFO: 2 incremental(s) of backup '1481552078404' will be removed
INFO: removing obsolete incremental backup '1481553914533'
INFO: removing obsolete incremental backup '1481553088053'
INFO: removing WAL file '000000010000000000000C1'
INFO: removing WAL file '000000010000000000000C0'
INFO: removing WAL file '000000010000000000000BF'
INFO: removing WAL file '000000010000000000000BE'
INFO: removing WAL file '000000010000000000000BD'
INFO: removing WAL file '000000010000000000000BC'
INFO: removing WAL file '000000010000000000000BB'
INFO: 48 Unused file(s) will be removed
INFO: removing (unused) file '000000010000000000000FA'
.
.
```

```
.
INFO: removing (unused) file '000000010000000000000BB.00000028.backup'
```

Only the backup chain with the `keep` status remains as shown by the following.

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
SERVER NAME    BACKUP ID     ... BACKUP PARENT   BACKUP TIME
... STATUS
acctg          1481559303348 ... 1481554203288   2016-12-12 11:15:03 EST
... keep
acctg          1481554203288 ... 1481553651165   2016-12-12 09:50:03 EST
... keep
acctg          1481553651165 ... none            2016-12-12 09:40:51 EST
... keep
```

CHAPTER 3

Sample BART System with Local and Remote Database Servers

This section describes a sample BART managed backup and recovery system consisting of both local and remote database servers. The complete steps to configure and operate the system are provided.

For detailed information about configuring a BART system, see the *EDB Postgres Backup and Recovery Installation and Upgrade Guide*. For detailed information about the operational procedures and BART subcommands, see the *EDB Postgres Backup and Recovery User Guide* both guides are available at the EnterpriseDB documentation web page.

The environment for this sample system is as follows:

- BART on host `192.168.2.22` running with BART user account `enterprisedb`
- Local Advanced Server on host `192.168.2.22` running with user account `enterprisedb`
- Remote Advanced Server on host `192.168.2.24` running with user account `enterprisedb`
- Remote PostgreSQL server on host `192.168.2.24` running with user account `postgres`

Passwordless SSH/SCP connections are required between the following:

- BART on host `192.168.2.22` and the local Advanced Server on the same host `192.168.2.22`
- BART on host `192.168.2.22` and the remote Advanced Server on host `192.168.2.24`
- BART on host `192.168.2.22` and the remote PostgreSQL server on host `192.168.2.24`

The following sections demonstrate configuring and taking full backups only. To support incremental backups as well, enable the `allow_incremental_backups` parameter for the desired database servers and use the `WAL scanner` program.

- *The BART Configuration File* shows the settings used in the BART configuration file.
- *Establishing SSH/SCP Passwordless Connections* provides an example of how to establish an SSH/SCP passwordless connection.
- *Configuring a Replication Database User* provides an example of how to configure the replication database user.
- *WAL Archiving Configuration Parameters* provides an example of how to configure WAL archiving.

- *Creating the BART Backup Catalog* provides information about creating a BART Backup Catalog.
- *Starting the Database Servers with WAL Archiving* provides example of starting the database servers with WAL archiving.
- *Taking a Full Backup* illustrates taking the first full backup of the database servers.
- *Using Point-In-Time Recovery* demonstrates the point-in-time recovery operation on the remote PostgreSQL database server.

3.1 The BART Configuration File

The following code snippet shows the settings used in the BART configuration file for the examples that follow:

```
[BART]
bart_host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /usr/edb/as11/bin/pg_basebackup
retention_policy = 6 BACKUPS
logfile = /tmp/bart.log
scanner_logfile = /tmp/bart_scanner.log

[ACCTG]
host = 127.0.0.1
port = 5444
user = enterprisedb
cluster_owner = enterprisedb
backup_name = acctg_%year-%month-%dayT%hour:%minute
archive_command = 'cp %p %a/%f'
description = "Accounting"

[MKTG]

host = 192.168.2.24
port = 5444
user = repuser
cluster_owner = enterprisedb
backup_name = mktg_%year-%month-%dayT%hour:%minute
remote_host = enterprisedb@192.168.2.24
description = "Marketing"

[HR]

host = 192.168.2.24
port = 5432
user = postgres
cluster_owner = postgres
backup_name = hr_%year-%month-%dayT%hour:%minute
remote_host = postgres@192.168.2.24
copy_wals_during_restore = enabled
description = "Human Resources"
```

3.2 Establishing SSH/SCP Passwordless Connections

This section demonstrates how passwordless SSH/SCP connections are established with the authorized public keys files.

3.2.1 Generating a Public Key File for the BART User Account

The BART user account is `enterprisedb` with a home directory of `/opt/PostgresPlus/9.5AS`.

To generate the public key file, as a root user, first create the `.ssh` subdirectory in the BART user's home directory and assign ownership of this directory to the `enterprisedb` user, ensuring there are no groups or other users that can access the `.ssh` directory.

```
[root@localhost 9.5AS]# pwd
/opt/PostgresPlus/9.5AS
[root@localhost 9.5AS]# mkdir .ssh
[root@localhost 9.5AS]# chown enterprisedb .ssh
[root@localhost 9.5AS]# chgrp enterprisedb .ssh
[root@localhost 9.5AS]# chmod 700 .ssh
[root@localhost 9.5AS]# ls -la | grep ssh
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 13:02 .ssh
```

Now, generate the public key file:

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ pwd
/opt/PostgresPlus/9.5AS
-bash-4.1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key
(/opt/PostgresPlus/9.5AS/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/opt/PostgresPlus/9.5AS/.ssh/id_rsa.
Your public key has been saved in
/opt/PostgresPlus/9.5AS/.ssh/id_rsa.pub.
The key fingerprint is:
de:65:34:d6:b1:d2:32:3c:b0:43:c6:a3:c0:9f:f4:64
enterprisedb@localhost.localdomain
The key's randomart image is:
+----[ RSA 2048]----+
|        .   .+   .  |
|         o .oE+ o o |
|          + * o.X + |
|           + .+ *   |
|           S   o    |
|          . . o     |
|           . .      |
|
|       |
|                    |
+--------------------+
```

The following are the resulting files. `id_rsa.pub` is the public key file of BART user account `enterprisedb`.

```
-bash-4.1$ ls -l .ssh
total 8
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:04 id_rsa.pub
```

3.2.2 Configuring Access between Local Advanced Server and the BART Host

Even when the Advanced Server database is on the same host as the BART user account, and the Advanced Server database cluster owner is also the BART user account (`enterprisedb` is this case), a passwordless SSH/SCP connection must be established from the same user account to itself.

On the BART host where the public key file was just generated (as shown in *Generating a Public Key File for the BART User Account*), create the authorized keys file by appending the public key file to any existing authorized keys file.

Log into the BART host as the BART user account and append the public key file, `id_rsa.pub` onto the `authorized_keys` file in the same `.ssh` directory.

```
[user@localhost ~]$ su - enterprisedb
Password:
Last login: Thu Mar 23 10:27:35 EDT 2017 on pts/0
-bash-4.2$ pwd
/opt/PostgresPlus/9.5AS
-bash-4.2$ ls -l .ssh
total 12
-rw------- 1 enterprisedb enterprisedb 1675 Mar 23 09:54 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Mar 23 09:54 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 345 Mar 23 10:05 known_hosts
-bash-4.2$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
-bash-4.2$ ls -l .ssh
total 16
-rw-rw-r-- 1 enterprisedb enterprisedb 416 Mar 23 10:33 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Mar 23 09:54 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Mar 23 09:54 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 345 Mar 23 10:05 known_hosts
```

The `authorized_keys` file must have file permission `600` as set by the following `chmod 600` command, or the passwordless connection will fail:

```
-bash-4.2$ chmod 600 ~/.ssh/authorized_keys
-bash-4.2$ ls -l .ssh
total 16
-rw------- 1 enterprisedb enterprisedb 416 Mar 23 10:33 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Mar 23 09:54 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Mar 23 09:54 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 345 Mar 23 10:05 known_hosts
```

Test the passwordless connection. Use the `ssh` command to verify that you can access the same user account as you are currently logged in as (`enterprisedb`) without being prompted for a password:

```
-bash-4.2$ ssh enterprisedb@127.0.0.1
Last login: Thu Mar 23 10:27:50 2017
-bash-4.2$ exit
logout
Connection to 127.0.0.1 closed.
```

3.2.3 Configuring Access from Remote Advanced Server to BART Host

On the remote host `192.168.2.24`, create the public key file for the remote database server user account, `enterprisedb`, for access to the BART user account, `enterprisedb`, on the BART host 192.168.2.22.

Create the `.ssh` directory for user account `enterprisedb` on the remote host:

```
[root@localhost 9.5AS]# pwd
/opt/PostgresPlus/9.5AS
[root@localhost 9.5AS]# mkdir .ssh
[root@localhost 9.5AS]# chown enterprisedb .ssh
[root@localhost 9.5AS]# chgrp enterprisedb .ssh
[root@localhost 9.5AS]# chmod 700 .ssh
[root@localhost 9.5AS]# ls -la | grep ssh
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 13:08 .ssh
```

Generate the public key file on the remote host for user account `enterprisedb`:

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key
(/opt/PostgresPlus/9.5AS/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/opt/PostgresPlus/9.5AS/.ssh/id_rsa.
Your public key has been saved in
/opt/PostgresPlus/9.5AS/.ssh/id_rsa.pub.
The key fingerprint is:
15:27:1e:1e:61:4b:48:66:67:0b:b2:be:fc:ea:ea:e6
enterprisedb@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048]---+
|     ..=.@..     |
|      =.O O      |
|     .   *       |
|    .  .         |
|    . S          |
|   . .           |
|    o            |
|   . .           |
| +Eoo..          |
+-----------------+
```

Copy the generated public key file, `id_rsa.pub`, to the BART user account, `enterprisedb`, on the BART host, `192.168.2.22`:

```
-bash-4.1$ scp ~/.ssh/id_rsa.pub enterprisedb@192.168.2.22:/tmp/tmp.pub
The authenticity of host '192.168.2.22 (192.168.2.22)' can't be
established.
RSA key fingerprint is b8:a9:97:31:79:16:b8:2b:b0:60:5a:91:38:d7:68:22.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.22' (RSA) to the list of known hosts.
enterprisedb@192.168.2.22's password:
id_rsa.pub
```

Log into the BART host as the BART user account and append the temporary public key file, `/tmp/tmp.pub` onto the `authorized_keys` file owned by the BART user account.

```
-bash-4.1$ ssh enterprisedb@192.168.2.22
enterprisedb@192.168.2.22's password:
Last login: Tue Apr 21 17:03:24 2015 from 192.168.2.22
-bash-4.1$ pwd
/opt/PostgresPlus/9.5AS
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 12
-rw-rw-r-- 1 enterprisedb enterprisedb 416 Apr 23 13:15 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:04 id_rsa.pub
```

The `authorized_keys` file must have file permission `600` as set by the following `chmod 600` command, otherwise the passwordless connection fails:

```
-bash-4.1$ chmod 600 ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 12
-rw------- 1 enterprisedb enterprisedb 416 Apr 23 13:15 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:04 id_rsa.pub
-bash-4.1$ rm /tmp/tmp.pub
-bash-4.1$ exit
logout
Connection to 192.168.2.22 closed.
```

Test the passwordless connection. From the remote host, verify that you can log into the BART host with the BART user account without being prompted for a password:

```
-bash-4.1$ ssh enterprisedb@192.168.2.22
Last login: Thu Apr 23 13:14:48 2015 from 192.168.2.24
-bash-4.1$ exit
logout
Connection to 192.168.2.22 closed.
```

3.2.4 Configuring Access from the BART Host to a Remote Advanced Server

On the BART host `192.168.2.22`, copy the public key file for the BART user account, `enterprisedb`, for access to the remote database server user account, `enterprisedb`, on the remote host `192.168.2.24`.

The following lists the current SSH keys files in the BART user's `.ssh` directory on the BART host:

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ pwd
/opt/PostgresPlus/9.5AS
-bash-4.1$ ls -l .ssh
total 12
-rw------- 1 enterprisedb enterprisedb 416 Apr 23 13:15 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:04 id_rsa.pub
```

The public key file, `id_rsa.pub`, for BART user account `enterprisedb` on the BART host that was earlier generated in *Generating a Public Key File for the BART User Account*, is now copied to the remote Advanced Server

host on 192.168.2.24:

```
-bash-4.1$ scp ~/.ssh/id_rsa.pub enterprisedb@192.168.2.24:/tmp/tmp.pub
The authenticity of host '192.168.2.24 (192.168.2.24)' can't be
established.
RSA key fingerprint is 59:41:fb:0c:ae:64:3d:3f:a2:d9:90:95:cf:2c:99:f2.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.24' (RSA) to the list of known
hosts.
enterprisedb@192.168.2.24's password:
id_rsa.pub
```

Log into the enterprisedb user account on the remote host and copy the public key file onto the authorized_keys file of the remote enterprisedb user account under its .ssh directory:

```
-bash-4.1$ ssh enterprisedb@192.168.2.24
enterprisedb@192.168.2.24's password:
Last login: Tue Apr 21 09:53:18 2015 from 192.168.2.22
-bash-4.1$ pwd
/opt/PostgresPlus/9.5AS
-bash-4.1$ ls -l .ssh
total 12
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:11 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:11 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 394 Apr 23 13:12 known_hosts
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
```

Adjust the file permission on authorized_keys:

```
-bash-4.1$ chmod 600 ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 enterprisedb enterprisedb 416 Apr 23 13:26 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:11 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:11 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 394 Apr 23 13:12 known_hosts
-bash-4.1$ rm /tmp/tmp.pub
-bash-4.1$ exit
logout
Connection to 192.168.2.24 closed.
```

While logged into the BART host, test the passwordless connection from the BART host to the remote Advanced Server host:

```
-bash-4.1$ ssh enterprisedb@192.168.2.24
Last login: Thu Apr 23 13:25:53 2015 from 192.168.2.22
-bash-4.1$ exit
logout
Connection to 192.168.2.24 closed.
```

3.2.5 Configuring Access from a Remote PostgreSQL Server to a BART Host

On the remote host (192.168.2.24), create a public key file owned by the database server user account (postgres), allowing access to the BART user account (enterprisedb) on the BART host (192.168.2.22).

Create the .ssh directory for the postgres user account on the remote host:

```
[root@localhost 9.5]# cd /opt/PostgreSQL/9.5
[root@localhost 9.5]# mkdir .ssh
[root@localhost 9.5]# chown postgres .ssh
[root@localhost 9.5]# chgrp postgres .ssh
[root@localhost 9.5]# chmod 700 .ssh
[root@localhost 9.5]# ls -la | grep ssh
drwx------ 2 postgres postgres 4096 Apr 23 13:32 .ssh
```

Create and copy the generated public key file, `id_rsa.pub`, to the BART user account (`enterprisedb`), on the BART host (`192.168.2.22`):

```
[user@localhost ~]$ su - postgres
Password:
-bash-4.1$ pwd
/opt/PostgreSQL/9.5
-bash-4.1$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/opt/PostgreSQL/9.5/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /opt/PostgreSQL/9.5/.ssh/id_rsa.
Your public key has been saved in /opt/PostgreSQL/9.5/.ssh/id_rsa.pub.
The key fingerprint is:
1f:f8:76:d6:fc:a5:1a:c5:5a:66:66:01:d0:a0:ca:ba
postgres@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048]----+
|          o+.    |
|         .  ..   |
|        .     .  |
|     . . .   . . |
|      o S .   O  |
|     .   o . @   |
|    .     + = o .|
|     .   . o . o.|
|    E       ... .|
+-----------------+
-bash-4.1$ ls -l .ssh
total 8
-rw------- 1 postgres postgres 1671 Apr 23 13:36 id_rsa
-rw-r--r-- 1 postgres postgres 412 Apr 23 13:36 id_rsa.pub
-bash-4.1$ scp ~/.ssh/id_rsa.pub enterprisedb@192.168.2.22:/tmp/tmp.pub
The authenticity of host '192.168.2.22 (192.168.2.22)' can't be
established.
RSA key fingerprint is b8:a9:97:31:79:16:b8:2b:b0:60:5a:91:38:d7:68:22.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.2.22' (RSA) to the list of known
hosts.
enterprisedb@192.168.2.22's password:
id_rsa.pub
```

Log into the BART host as the BART user account and append the temporary public key file, `/tmp/tmp.pub`, onto the `authorized_keys` file owned by the BART user account.

```
-bash-4.1$ ssh enterprisedb@192.168.2.22
enterprisedb@192.168.2.22's password:
Last login: Thu Apr 23 13:19:25 2015 from 192.168.2.24
-bash-4.1$ pwd
```

```
/opt/PostgresPlus/9.5AS
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 enterprisedb enterprisedb 828 Apr 23 13:40 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:04 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 394 Apr 23 13:24 known_hosts
-bash-4.1$ rm /tmp/tmp.pub
-bash-4.1$ exit
logout
Connection to 192.168.2.22 closed.
```

Make sure the `authorized_keys` file has file permission 600 as shown, or the passwordless connection will fail. Test the passwordless connection; from the remote host, while logged in as user account `postgres`, verify that you can log into the BART host with the BART user account without being prompted for a password:

```
-bash-4.1$ pwd
/opt/PostgreSQL/9.5
-bash-4.1$ ssh enterprisedb@192.168.2.22
Last login: Thu Apr 23 13:40:10 2015 from 192.168.2.24
-bash-4.1$ exit
logout
Connection to 192.168.2.22 closed.
```

3.2.6 Configuring Access from the BART Host to Remote PostgreSQL

Copy the public key file on the BART host that is owned by the BART user account (`enterprisedb`) to the remote database server user account (`postgres`), on the remote host (192.168.2.24).

The following lists the current SSH keys files in the BART user's `.ssh` directory on the BART host:

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 enterprisedb enterprisedb 828 Apr 23 13:40 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:04 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 394 Apr 23 13:24 known_hosts
```

The public key file, `id_rsa.pub`, for BART user account `enterprisedb` on the BART host that was earlier generated in *Generating a Public Key File for the BART User Account*, now resides on the remote PostgreSQL host:

```
-bash-4.1$ scp ~/.ssh/id_rsa.pub postgres@192.168.2.24:/tmp/tmp.pub
postgres@192.168.2.24's password:
id_rsa.pub
```

Log into the `postgres` user account on the remote host and copy the public key file onto the `authorized_keys` file of `postgres` under its `.ssh` directory:

```
-bash-4.1$ ssh postgres@192.168.2.24
postgres@192.168.2.24's password:
Last login: Mon Jan 26 18:08:36 2015 from 192.168.2.19
-bash-4.1$ pwd
```

```
/opt/PostgreSQL/9.5
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
```

Adjust the file permissions on `authorized_keys`:

```
-bash-4.1$ ls -l .ssh
total 16
-rw-rw-r-- 1 postgres postgres 416 Apr 23 13:52 authorized_keys
-rw------- 1 postgres postgres 1671 Apr 23 13:36 id_rsa
-rw-r--r-- 1 postgres postgres 412 Apr 23 13:36 id_rsa.pub
-rw-r--r-- 1 postgres postgres 394 Apr 23 13:36 known_hosts
-bash-4.1$ chmod 600 ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 postgres postgres 416 Apr 23 13:52 authorized_keys
-rw------- 1 postgres postgres 1671 Apr 23 13:36 id_rsa
-rw-r--r-- 1 postgres postgres 412 Apr 23 13:36 id_rsa.pub
-rw-r--r-- 1 postgres postgres 394 Apr 23 13:36 known_hosts
-bash-4.1$ rm /tmp/tmp.pub
-bash-4.1$ exit
logout
Connection to 192.168.2.24 closed.
```

Test the passwordless connection from the BART host to the remote PostgreSQL host:

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ ssh postgres@192.168.2.24
Last login: Thu Apr 23 13:52:25 2015 from 192.168.2.22
-bash-4.1$ exit
logout
Connection to 192.168.2.24 closed.
```

3.3 Configuring a Replication Database User

This section demonstrates how a replication database user is established.

All database servers must use a superuser as the replication database user.

The replication database user for each database server is specified by the `user` parameter in the BART configuration file as shown by the following:

```
[ACCTG]

host = 127.0.0.1
port = 5444
user = enterprisedb <=== Replication Database User
cluster_owner = enterprisedb
backup_name = acctg_%year-%month-%dayT%hour:%minute
archive_command = 'cp %p %a/%f'
description = "Accounting"

[MKTG]
host = 192.168.2.24
port = 5444
user = repuser <=== Replication Database User
cluster_owner = enterprisedb
backup_name = mktg_%year-%month-%dayT%hour:%minute
remote_host = enterprisedb@192.168.2.24
description = "Marketing"

[HR]

host = 192.168.2.24
port = 5432
user = postgres <=== Replication Database User
cluster_owner = enterprisedb
backup_name = hr_%year-%month-%dayT%hour:%minute
remote_host = postgres@192.168.2.24
copy_wals_during_restore = enabled
description = "Human Resources"
```

Add entries to the `.pgpass` file on each server to allow the BART user account to initiate a backup without being prompted for credentials. The `.pgpass` file is located in `/opt/PostgresPlus/9.5AS/.pgpass`:

```
127.0.0.1:5444:*:enterprisedb:password
192.168.2.24:5444:*:repuser:password
192.168.2.24:5432:*:postgres:password
```

For more information about using a `.pgpass` file, please see the PostgreSQL documentation.

While connected to `MKTG` on 192.168.2.24, execute the following `CREATE ROLE` command to create the replication database superuser:

```
CREATE ROLE repuser WITH LOGIN SUPERUSER PASSWORD 'password';
```

Access is granted in the `pg_hba.conf` file for the local Advanced Server:

```
# TYPE     DATABASE            USER        ADDRESS         METHOD
# "local" is for Unix domain socket connections only
local        all                 all                         md5
```

```
# IPv4     local connections:
host       template1         enterprisedb  127.0.0.1/32      md5
host       edb               enterprisedb  127.0.0.1/32      md5
#host      all               all           127.0.0.1/32      md5
# IPv6 local connections:
host       all               all           ::1/128           md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local     replication       enterprisedb                    md5
host       replication       enterprisedb  127.0.0.1/32      md5
```

Similarly, access is granted in the `pg_hba.conf` file for the remote Advanced Server installation:

```
# TYPE    DATABASE           USER        ADDRESS             METHOD
# "local" is for Unix domain socket connections only
local      all               all                               md5
# IPv4 local connections:
host       template1         repuser      192.168.2.22/32      md5
host       all               enterprisedb 127.0.0.1/32         md5
#host      all               all          127.0.0.1/32         md5
# IPv6 local connections:
host       all               all           ::1/128             md5
# Allow replication connections from localhost, by a user with the
# replication privilege.
#local   replication         enterprisedb                      md5
host     replication         repuser       192.168.2.22/32     md5
```

Access is also granted in the `pg_hba.conf` file for the remote PostgreSQL server:

```
# TYPE    DATABASE           USER        ADDRESS               METHOD
# "local" is for Unix domain socket connections only
local      all               all                                 md5
# IPv4 local connections:
host       template1         postgres    192.168.2.22/32         md5
host       all               all         127.0.0.1/32            md5
# IPv6 local connections:
host       all               all         ::1/128                 md5
# Allow replication connections from localhost, by a user with the
q# replication privilege.
#local     replication       postgres                            md5
host       replication       postgres    192.168.2.22/32         md5
```

3.4 WAL Archiving Configuration Parameters

Use the following parameters in the `postgresql.conf` file to enable WAL archiving. The `postgresql.conf` file for the local Advanced Server database (`ACCTG`) is set as follows:

```
wal_level = archive
archive_mode = on                                   # allows archiving to be done
                                                    # (change requires restart)
#archive_command = ''                               # command to use to archive
                                                      a logfile segment
                                                    # placeholders: %p = path of
                                                     file to archive
                                                    # %f = file name only
max_wal_senders = 3
```

When the `INIT` subcommand is invoked, the Postgres `archive_command` configuration parameter in the `postgresql.auto.conf` file will be set based on the BART `archive_command` parameter located in the BART configuration file.

Note: If the Postgres `archive_command` is already set, invoke the `INIT` subcommand with the `--no-configure` option to prevent the `archive_command` from being reset. For details, see *INIT*.

```
[BART]
bart_host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /usr/edb/as11/bin/pg_basebackup
retention_policy = 6 BACKUPS
logfile = /tmp/bart.log
scanner_logfile = /tmp/bart_scanner.log

[ACCTG]
host = 127.0.0.1
port = 5444
user = enterprisedb
cluster_owner = enterprisedb
backup_name = acctg_%year-%month-%dayT%hour:%minute
archive_command = 'cp %p %a/%f'
description = "Accounting"
```

When the `INIT` subcommand is invoked, the `postgresql.auto.conf` file contains the following:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'cp %p /opt/backup/acctg/archived_wals/%f'
```

The `archive_command` uses the `cp` command instead of `scp` since the BART backup catalog is local to this database cluster and the BART user account (the account that owns the backup catalog, `enterprisedb`), is the same user account running Advanced Server. The result is that there is no directory permission conflict during the archive operation.

The `postgresql.conf` file for the remote Advanced Server, `MKTG` is set as follows:

```
wal_level = archive
archive_mode = on                                   # allows archiving to be done
                                                    # (change requires restart)
archive_command = ''                                # command to use to archive a
```

```
                                                  logfile segment
                                                # placeholders: %p = path of
                                                file to archive
                                                # %f = file name only
max_wal_senders = 3
```

When the `INIT` subcommand is invoked, the Postgres `archive_command` configuration parameter in the `postgresql.auto.conf` file will be set by the default BART format of the BART `archive_command` parameter (since it is not explicitly set for this database server in the BART configuration file).

```
[BART]
bart_host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /usr/edb/as11/bin/pg_basebackup
retention_policy = 6 BACKUPS
logfile = /tmp/bart.log
scanner_logfile = /tmp/bart_scanner.log
.
.
.
[MKTG]

host = 192.168.2.24
port = 5444
user = repuser
cluster_owner = enterprisedb
backup_name = mktg_%year-%month-%dayT%hour:%minute
remote_host = enterprisedb@192.168.2.24
description = "Marketing"
```

The default, BART `archive_command` format is the following:

```
archive_command = 'scp %p %h:%a/%f'
```

The `postgresql.auto.conf` file contains the following after the `INIT` subcommand is invoked:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'scp %p
enterprisedb@192.168.2.22:/opt/backup/hr/archived_wals/%f'
```

The `archive_command` uses the `scp` command since the BART backup catalog is remote relative to this database cluster. The BART user account, `enterprisedb`, is specified on the `scp` command since this is the user account owning the BART backup catalog where the archived WAL files are to be copied. The result is that there is no directory permission conflict during the archive operation.

The `postgresql.conf` file for the remote PostgreSQL server (`HR`) is set as follows:

```
wal_level = archive
archive_mode = on                               # allows archiving to be done
                                                # (change requires restart)
#archive_command = ''                           # command to use to archive a
                                                  logfile segment
                                                # placeholders: %p = path of
                                                  file to archive
                                                # %f = file name only
max_wal_senders = 3
```

When the INIT subcommand is invoked, the Postgres archive_command configuration parameter in the postgresql.auto.conf file will be set by the default BART format of the BART archive_command parameter (since it is not explicitly set for this database server in the BART configuration file):

```
[BART]

bart_host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /usr/edb/as11/bin/pg_basebackup
retention_policy = 6 BACKUPS
logfile = /tmp/bart.log
scanner_logfile = /tmp/bart_scanner.log
.
.
.
[HR]

host = 192.168.2.24
port = 5432
user = postgres
cluster_owner = postgres
backup_name = hr_%year-%month-%dayT%hour:%minute
remote_host = postgres@192.168.2.24
copy_wals_during_restore = enabled
description = "Human Resources"
```

The default, the BART archive_command format is:

```
archive_command = 'scp %p %h:%a/%f'
```

The postgresql.auto.conf file contains the following after the INIT subcommand is invoked:

```
# Do not edit this file manually!
# It will be overwritten by ALTER SYSTEM command.
archive_command = 'scp %p
enterprisedb@192.168.2.22:/opt/backup/hr/archived_wals/%f'
```

The archive_command uses the scp command since the BART backup catalog is remote relative to this database cluster. The BART user account, enterprisedb, is specified on the scp command since this is the user account owning the BART backup catalog where the archived WAL files are to be copied. The result is that there is no directory permission conflict during the archive operation.

3.5 Creating the BART Backup Catalog (backup_path)

Create the directory specified by the `backup_path` configuration parameter.

```
[BART]

bart_host= enterprisedb@192.168.2.22
backup_path = /opt/backup
pg_basebackup_path = /usr/edb/as11/bin/pg_basebackup
retention_policy = 6 BACKUPS
logfile = /tmp/bart.log
scanner_logfile = /tmp/bart_scanner.log
```

Ensure that the directory is owned by the BART user account:

```
[root@localhost opt]# pwd
/opt
[root@localhost opt]# mkdir backup
[root@localhost opt]# chown enterprisedb backup
[root@localhost opt]# chgrp enterprisedb backup
[root@localhost opt]# chmod 700 backup
[root@localhost opt]# ls -l | grep backup
drwx------ 2 enterprisedb enterprisedb 4096 Apr 23 15:36 backup
```

Use the BART `INIT` subcommand to complete the directory structure and set the Postgres `archive_command` configuration parameter.

Before invoking any BART subcommands, set up a profile under the BART user account's home directory to set the `LD_LIBRARY_PATH` and `PATH` environment variables. For more information regarding setting this variable, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide.

The `-o` option is specified with the `INIT` subcommand to force the setting of the Postgres `archive_command` configuration parameter when `archive_mode` is `off` or if the Postgres `archive_command` parameter is already set and needs to be overridden.

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ bart INIT -o
INFO: setting archive_command for server 'acctg'
WARNING: archive_command is set. server restart is required
INFO: setting archive_command for server 'hr'
WARNING: archive_command is set. server restart is required
INFO: setting archive_command for server 'mktg'
WARNING: archive_command is set. server restart is required
```

The BART `SHOW-SERVERS` subcommand displays the following:

```
-bash-4.1$ bart SHOW-SERVERS
SERVER NAME :               acctg
BACKUP FRIENDLY NAME:       acctg_%year-%month-%dayT%hour:%minute
HOST NAME :                 127.0.0.1
USER NAME :                 enterprisedb
PORT :                      5444
REMOTE HOST :
RETENTION POLICY :          6 Backups
DISK UTILIZATION :          0.00 bytes
NUMBER OF ARCHIVES :        0
ARCHIVE PATH :              /opt/backup/acctg/archived_wals
```

```
ARCHIVE COMMAND :          (disabled)
XLOG METHOD :              fetch
WAL COMPRESSION :          disabled
TABLESPACE PATH(s) :
INCREMENTAL BACKUP :       DISABLED
DESCRIPTION :              "Accounting"
SERVER NAME :              hr
BACKUP FRIENDLY NAME:      hr_%year-%month-%dayT%hour:%minute
HOST NAME :                192.168.2.24
USER NAME :                postgres
PORT :                     5432
REMOTE HOST :              postgres@192.168.2.24
RETENTION POLICY :         6 Backups
DISK UTILIZATION :         0.00 bytes
NUMBER OF ARCHIVES :       0
ARCHIVE PATH :             /opt/backup/hr/archived_wals
ARCHIVE COMMAND :          (disabled)
XLOG METHOD :              fetch
WAL COMPRESSION :          disabled
TABLESPACE PATH(s) :
INCREMENTAL BACKUP :       DISABLED
DESCRIPTION :              "Human Resources"
SERVER NAME :              mktg
BACKUP FRIENDLY NAME:      mktg_%year-%month-%dayT%hour:%minute
HOST NAME :                192.168.2.24
USER NAME :                repuser
PORT :                     5444
REMOTE HOST :              enterprisedb@192.168.2.24
RETENTION POLICY :         6 Backups
DISK UTILIZATION :         0.00 bytes
NUMBER OF ARCHIVES :       0
ARCHIVE PATH :             /opt/backup/mktg/archived_wals
ARCHIVE COMMAND :          (disabled)
XLOG METHOD :              fetch
WAL COMPRESSION :          disabled
TABLESPACE PATH(s) :
INCREMENTAL BACKUP :       DISABLED
DESCRIPTION :              "Marketing"
-bash-4.1$ cd /opt/backup
-bash-4.1$ pwd
/opt/backup
-bash-4.1$ ls -l
total 12
drwxrwxr-x 3 enterprisedb enterprisedb 4096 Mar 29 13:16 acctg
drwxrwxr-x 3 enterprisedb enterprisedb 4096 Mar 29 13:16 hr
drwxrwxr-x 3 enterprisedb enterprisedb 4096 Mar 29 13:16 mktg
-bash-4.1$ ls -l acctg
total 4
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Mar 29 13:16 archived_wals
-bash-4.1$ ls -l hr
total 4
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Mar 29 13:16 archived_wals
-bash-4.1$ ls -l mktg
total 4
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Mar 29 13:16 archived_wals
```

The `ARCHIVE PATH` field displays the full directory path to where the WAL files are copied. This directory path must match the directory path specified in the Postgres `archive_command` parameter of the `postgresql.conf`

file or the `postgresql.auto.conf` file of each database server.

3.5.1 Starting the Database Servers with WAL Archiving

After the BART backup catalog directory structure has been configured, start the archiving of WAL files from the database servers by restarting each database server. On BART host 192.168.2.22:

```
[root@localhost data]# service ppas-9.5 restart
```

On remote host 192.168.2.24:

```
[root@localhost data]# service ppas-9.5 restart

[root@localhost data]# service postgresql-9.5 restart
```

In the BART backup catalog, verify that the WAL files are archiving.

Archived WAL files may not appear very frequently depending upon how often WAL archiving is set to switch to a new segment file with the `archive_timeout` parameter in your database server configuration settings.

Verify that there are no archiving-related errors in the database server log files.

3.6 Taking a Full Backup

The following code snippet shows the first full backup of the database servers.

```
-bash-4.1$ bart BACKUP -s acctg -z
INFO: creating backup for server 'acctg'
INFO: backup identifier: '1490809695281'
60776/60776 kB (100%), 1/1 tablespace

INFO: backup completed successfully
INFO: backup checksum: 37f3defb98ca88dcf05079815555dfc2 of base.tar.gz
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1490809695281
BACKUP NAME: acctg_2017-03-29T13:48
BACKUP PARENT: none
BACKUP LOCATION: /opt/backup/acctg/1490809695281
BACKUP SIZE: 6.10 MB
BACKUP FORMAT: tar.gz
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
ChkSum File
37f3defb98ca88dcf05079815555dfc2 base.tar.gz

TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000004
STOP WAL LOCATION: 000000010000000000000004
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2017-03-29 13:48:15 EDT
STOP TIME: 2017-03-29 13:48:17 EDT
TOTAL DURATION: 2 sec(s)

-bash-4.1$ bart BACKUP -s mktg -z
INFO: creating backup for server 'mktg'
INFO: backup identifier: '1490809751193'
61016/61016 kB (100%), 1/1 tablespace

INFO: backup completed successfully
INFO: backup checksum: 8b010e130a105e76d01346bb56dfcf14 of base.tar.gz
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1490809751193
BACKUP NAME: mktg_2017-03-29T13:49
BACKUP PARENT: none
BACKUP LOCATION: /opt/backup/mktg/1490809751193
BACKUP SIZE: 6.13 MB
BACKUP FORMAT: tar.gz
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
ChkSum File
8b010e130a105e76d01346bb56dfcf14 base.tar.gz

TABLESPACE(s): 0
```

```
START WAL LOCATION: 000000010000000100000085
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2017-03-29 13:49:11 EDT
STOP TIME: 2017-03-29 13:49:14 EDT
TOTAL DURATION: 3 sec(s)

-bash-4.1$ bart BACKUP -s hr -z
INFO: creating backup for server 'hr'
INFO: backup identifier: '1490809824946'
38991/38991 kB (100%), 1/1 tablespace
INFO: backup completed successfully
INFO: backup checksum: 277e8a1a80ba3474f541eb316a417c9a of base.tar.gz
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1490809824946
BACKUP NAME: hr_2017-03-29T13:50
BACKUP PARENT: none
BACKUP LOCATION: /opt/backup/hr/1490809824946
BACKUP SIZE: 2.59 MB
BACKUP FORMAT: tar.gz
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
ChkSum File
277e8a1a80ba3474f541eb316a417c9a base.tar.gz

TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000002
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2017-03-29 13:50:25 EDT
STOP TIME: 2017-03-29 13:50:26 EDT
TOTAL DURATION: 1 sec(s)
```

The following code snippet shows the backup directories created for each backup of each database server. The backup ID is used as the backup directory name.

```
-bash-4.1$ cd /opt/backup
-bash-4.1$ ls -l
total 12
drwxrwxr-x 4 enterprisedb enterprisedb 4096 Mar 29 13:48 acctg
drwxrwxr-x 4 enterprisedb enterprisedb 4096 Mar 29 13:50 hr
drwxrwxr-x 4 enterprisedb enterprisedb 4096 Mar 29 13:49 mktg
-bash-4.1$ ls -l acctg
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Mar 29 13:48 1490809695281
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Mar 29 13:48 archived_wals
-bash-4.1$ ls -l hr
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Mar 29 13:50 1490809824946
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Mar 29 13:50 archived_wals
-bash-4.1$ ls -l mktg
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Mar 29 13:49 1490809751193
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Mar 29 13:49 archived_wals
```

3.7 Using Point-In-Time Recovery

This section demonstrates using the point-in-time recovery operation on the remote PostgreSQL database server.

The following tables were created about two minutes apart with WAL archiving enabled:

```
postgres=# \dt

             List of relations
Schema  |      Name      |  Type   | Owner

--------+----------------+---------+---------
public  | hr_rmt_t1_1356 | table   | postgres
public  | hr_rmt_t1_1358 | table   | postgres
public  | hr_rmt_t1_1400 | table   | postgres
public  | hr_rmt_t1_1402 | table   | postgres
public  | hr_rmt_t1_1404 | table   | postgres
public  | hr_rmt_t1_1406 | table   | postgres
(6 rows)
```

In the table name `hr_rmt_t<n>_<hhmi>`, `n` represents the active timeline. `<hhmi>` is the approximate time the table was created. For example, `hr_rmt_t1_1356` was created at approximately 1:56 PM while timeline #1 is active.

The PostgreSQL database server was then stopped.

WAL files that have been created, but not yet archived must be identified, and then saved.

The following are the archived WAL files in the BART backup catalog:

```
-bash-4.1$ ls -l hr/archived_wals
total 49156
-rw------- 1 enterprisedb enterprisedb 16777216 Mar 29 13:50
000000010000000000000001
-rw------- 1 enterprisedb enterprisedb 16777216 Mar 29 13:50
000000010000000000000002
-rw------- 1 enterprisedb enterprisedb 302 Mar 29 13:50
000000010000000000000002.00000028.backup
-rw------- 1 enterprisedb enterprisedb 16777216 Mar 29 14:07
000000010000000000000003
```

The following snippet lists the current PostgreSQL server WAL files. The unarchived WAL files are marked with two stars (**).

```
-bash-4.1$ cd /opt/PostgreSQL/9.5/data/pg_xlog
-bash-4.1$ pwd
/opt/PostgreSQL/9.5/data/pg_xlog
-bash-4.1$ ls -l
total 49160
-rw------- 1 postgres postgres 302 Mar 29 13:50
000000010000000000000002.00000028.backup
-rw------- 1 postgres postgres 16777216 Mar 29 14:07
000000010000000000000003
-rw------- 1 postgres postgres 16777216 Mar 29 14:07
**000000010000000000000004**
-rw------- 1 postgres postgres 16777216 Mar 29 13:50
**000000010000000000000005**
drwx------ 2 postgres postgres 4096 Mar 29 14:07 archive_status
```

Copies of the unarchived WAL files are saved to a temporary location:

```
-bash-4.1$ mkdir /tmp/unarchived_pg95_wals
-bash-4.1$ pwd
/opt/PostgreSQL/9.5/data/pg_xlog
bash-4.1$ cp -p 000000010000000000000004 /tmp/unarchived_pg95_wals
bash-4.1$ cp -p 000000010000000000000005 /tmp/unarchived_pg95_wals
bash-4.1$ ls -l /tmp/unarchived_pg95_wals
total 32768
-rw------- 1 postgres postgres 16777216 Mar 29 14:07 000000010000000000000004
-rw------- 1 postgres postgres 16777216 Mar 29 13:50 000000010000000000000005
```

On the remote host, a directory is created to which the PostgreSQL database cluster is to be restored. This restore path is named `/opt/restore_pg95` and is owned by user account `postgres`.

```
[user@localhost ~]$ su root
Password:
[root@localhost user]# cd /opt
[root@localhost opt]# mkdir restore_pg95
[root@localhost opt]# chown postgres restore_pg95
[root@localhost opt]# chgrp postgres restore_pg95
[root@localhost opt]# chmod 700 restore_pg95
[root@localhost opt]# ls -l
total 16
drwxr-xr-x 4 root daemon 4096 Mar 29 12:10 PostgresPlus
drwxr-xr-x 3 root daemon 4096 Mar 29 12:25 PostgreSQL
drwx------ 2 postgres postgres 4096 Mar 29 14:15 restore_pg95
drwxr-xr-x. 2 root root 4096 Nov 22 2013 rh
```

In the BART configuration file, the remote user and remote host IP address, `postgres@192.168.2.24`, have been set with the `remote_host` parameter. If not given in the BART configuration file, this information must then be specified by the `--remote-host` option when giving the `RESTORE` subcommand (for example, `bart RESTORE --remote-host postgres@192.168.2.24 ...`).

```
[HR]

host = 192.168.2.24
port = 5432
user = postgres
cluster_owner = postgres
backup_name = hr_%year-%month-%dayT%hour:%minute
remote_host = postgres@192.168.2.24
copy_wals_during_restore = enabled
description = "Human Resources"
```

Use the `SHOW-BACKUPS` subcommand to identify the backup to use with the `RESTORE` subcommand.

```
SERVER NAME          BACKUP ID         BACKUP NAME             BACKUP PARENT
BACKUP TIME
BACKUP SIZE          WAL(s) SIZE       WAL FILES               STATUS
acctg                1490809695281     acctg_2017-03-29T13:48  none
2017-03-29 13:48:17 EDT
6.10 MB              32.00 MB          2                       active
hr                   1490809824946     hr_2017-03-29T13:50     none
2017-03-29 13:50:26 EDT
2.59 MB              32.00 MB          2                       active
mktg                 1490809751193     mktg_2017-03-29T13:49   none
2017-03-29 13:49:14 EDT
```

```
6.13 MB              64.00 MB          4                       active
```

The `-t` option with the `SHOW-BACKUPS` subcommand displays additional backup information:

```
-bash-4.1$ bart SHOW-BACKUPS -s hr -i 1490809824946 -t
SERVER NAME    : hr
BACKUP ID      : 1490809824946
BACKUP NAME    : hr_2017-03-29T13:50
BACKUP PARENT  : none
BACKUP STATUS  : active
BACKUP TIME    : 2017-03-29 13:50:26 EDT
BACKUP SIZE    : 2.59 MB
WAL(S) SIZE    : 32.00 MB
NO. OF WALS    : 2
FIRST WAL FILE : 000000010000000000000002
CREATION TIME  : 2017-03-29 13:50:31 EDT
LAST WAL FILE  : 000000010000000000000003
CREATION TIME  : 2017-03-29 14:07:35 EDT
```

A recovery is made using timeline `1` to `2017-03-29 14:01:00`.

```
-bash-4.1$ bart RESTORE -s hr -i hr_2017-03-29T13:50 -p
/opt/restore_pg95 -t 1 -g '2017-03-29 14:01:00'
INFO: restoring backup 'hr_2017-03-29T13:50' of server 'hr'
INFO: base backup restored
INFO: copying WAL file(s) to
postgres@192.168.2.24:/opt/restore_pg95/archived_wals
INFO: writing recovery settings to postgresql.auto.conf file
INFO: archiving is disabled
INFO: permissions set on $PGDATA
INFO: restore completed successfully
```

The following example shows the restored backup files in the restore path directory, `/opt/restore_pg95`:

```
-bash-4.1$ pwd
/opt/restore_pg95
-bash-4.1$ ls -l
total 128
drwxr-xr-x 2 postgres postgres 4096 Mar 29 14:27 archived_wals
-rw------- 1 postgres postgres 206 Mar 29 13:50 backup_label
drwx------ 5 postgres postgres 4096 Mar 29 12:25 base
drwx------ 2 postgres postgres 4096 Mar 29 14:27 global
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_clog
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_commit_ts
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_dynshmem
-rw------- 1 postgres postgres 4212 Mar 29 13:18 pg_hba.conf
-rw------- 1 postgres postgres 1636 Mar 29 12:25 pg_ident.conf
drwxr-xr-x 2 postgres postgres 4096 Mar 29 13:45 pg_log
drwx------ 4 postgres postgres 4096 Mar 29 12:25 pg_logical
drwx------ 4 postgres postgres 4096 Mar 29 12:25 pg_multixact
drwx------ 2 postgres postgres 4096 Mar 29 13:43 pg_notify
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_replslot
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_serial
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_snapshots
drwx------ 2 postgres postgres 4096 Mar 29 13:43 pg_stat
drwx------ 2 postgres postgres 4096 Mar 29 13:50 pg_stat_tmp
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_subtrans
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_tblspc
```

```
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_twophase
-rw------- 1 postgres postgres 4 Mar 29 12:25 PG_VERSION
drwx------ 3 postgres postgres 4096 Mar 29 14:27 pg_xlog
-rw------- 1 postgres postgres 169 Mar 29 13:24 postgresql.auto.conf
-rw-r--r-- 1 postgres postgres 21458 Mar 29 14:27 postgresql.conf
-rw-r--r-- 1 postgres postgres 118 Mar 29 14:27 postgresql.auto.conf
```

Copy the saved, unarchived WAL files to the restore path `pg_xlog` subdirectory (`/opt/restore_pg95/pg_xlog`):

```
-bash-4.1$ pwd
/opt/restore_pg95/pg_xlog
-bash-4.1$ ls -l
total 16388
-rw------- 1 postgres postgres 16777216 Mar 29 13:50
000000010000000000000002
drwx------ 2 postgres postgres 4096 Mar 29 14:27 archive_status
-bash-4.1$ ls -l /tmp/unarchived_pg95_wals
total 32768
-rw------- 1 postgres postgres 16777216 Mar 29 14:07
000000010000000000000004
-rw------- 1 postgres postgres 16777216 Mar 29 13:50
000000010000000000000005
-bash-4.1$ cp -p /tmp/unarchived_pg95_wals/* .
-bash-4.1$ ls -l
total 49156
-rw------- 1 postgres postgres 16777216 Mar 29 13:50
000000010000000000000002
-rw------- 1 postgres postgres 16777216 Mar 29 14:07
000000010000000000000004
-rw------- 1 postgres postgres 16777216 Mar 29 13:50
000000010000000000000005
drwx------ 2 postgres postgres 4096 Mar 29 14:27 archive_status
```

Inspect the `/opt/restore_pg95/postgresql.auto.conf` file to verify that it contains the correct recovery settings:

```
restore_command = 'cp archived_wals/%f %p'
recovery_target_time = '2017-03-29 14:01:00'
recovery_target_timeline = 1
```

Note that the command restores from the `archived_wals` subdirectory of `/opt/restore_pg95` since the `copy_wals_during_restore` parameter in the BART configuration file is set to `enabled` for database server `hr`.

Start the database server to initiate the point-in-time recovery operation:

```
[user@localhost ~]$ su postgres
Password:
bash-4.1$ cd /opt/restore_pg95
bash-4.1$ /opt/PostgreSQL/9.5/bin/pg_ctl start -D /opt/restore_pg95 -l
/opt/restore_pg95/pg_log/logfile
server starting
```

Inspect the database server log file to ensure the operation did not result in any errors:

```
2017-03-29 14:33:23 EDT LOG: database system was interrupted; last known
up at 2017-03-29 13:50:25 EDT
```

```
2017-03-29 14:33:23 EDT LOG: starting point-in-time recovery to
2017-03-29 14:01:00-04
2017-03-29 14:33:23 EDT LOG: restored log file
"000000010000000000000002" from archive
2017-03-29 14:33:23 EDT LOG: redo starts at 0/2000098
2017-03-29 14:33:23 EDT LOG: consistent recovery state reached at
0/20000C0
2017-03-29 14:33:23 EDT LOG: restored log file
"000000010000000000000003" from archive
2017-03-29 14:33:23 EDT LOG: recovery stopping before commit of
transaction 1762, time 2017-03-29 14:02:28.100072-04
2017-03-29 14:33:23 EDT LOG: redo done at 0/303F390
2017-03-29 14:33:23 EDT LOG: last completed transaction was at log time
2017-03-29 14:00:43.351333-04
cp: cannot stat `archived_wals/00000002.history': No such file or
directory
2017-03-29 14:33:23 EDT LOG: selected new timeline ID: 2
cp: cannot stat `archived_wals/00000001.history': No such file or
directory
2017-03-29 14:33:23 EDT LOG: archive recovery complete
2017-03-29 14:33:23 EDT LOG: MultiXact member wraparound protections are
now enabled
2017-03-29 14:33:23 EDT LOG: database system is ready to accept
connections
2017-03-29 14:33:23 EDT LOG: autovacuum launcher started
```

The tables that exist in the recovered database cluster are:

```
postgres=# \dt
          List of relations
Schema  | Name           | Type  | Owner
--------+----------------+-------+----------
public  | hr_rmt_t1_1356 | table | postgres
public  | hr_rmt_t1_1358 | table | postgres
public  | hr_rmt_t1_1400 | table | postgres
(3 rows)
```

Since recovery was up to and including 2017-03-29 14:01:00, the following tables created after 14:01 are not present:

```
public  | hr_rmt_t1_1402  | table  | postgres
public  | hr_rmt_t1_1404  | table  | postgres
public  | hr_rmt_t1_1406  | table  | postgres
```

The BART `RESTORE` operation stops WAL archiving by adding an `archive_mode = off` parameter at the very end of the `postgresql.conf` file. This last parameter in the file overrides any other previous setting of the same parameter in the file. Delete the last setting and restart the database server to start WAL archiving.

```
# Add settings for extensions here
archive_mode = off
```

CHAPTER 4

Conclusion

EnterpriseDB® Corporation

34 Crosby Drive, Suite 201, Bedford, MA 01730, USA

T +1 781 357 3390 F +1 978 467 1307 E

info@enterprisedb.com

www.enterprisedb.com

- EDB designs, establishes coding best practices, reviews, and verifies input validation for the logon UI for EDB Postgres product where present. EDB follows the same approach for additional input components, however the nature of the product may require that it accepts freeform SQL, WMI or other strings to be entered and submitted by trusted users for which limited validation is possible. In such cases it is not possible to prevent users from entering incorrect or otherwise dangerous inputs.
- EDB reserves the right to add features to products that accept freeform SQL, WMI or other potentially dangerous inputs from authenticated, trusted users in the future, but will ensure all such features are designed and tested to ensure they provide the minimum possible risk, and where possible, require superuser or equivalent privileges.
- EDB does not that warrant that we can or will anticipate all potential threats and therefore our process cannot fully guarantee that all potential vulnerabilities have been addressed or considered.

Index