EDB Postgres™ Backup and Recovery Guide

EDB Postgres™ Backup and Recovery 2.4

June 17, 2019

EDB Postgres™ Backup and Recovery Guide
by EnterpriseDB® Corporation

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

Table of Contents

1 Introduction

The EDB Backup and Recovery Tool (BART) is an administrative utility providing simplified backup and recovery management for multiple local or remote EDB Postgres™ Advanced Server and PostgreSQL® database servers.

The following are some of the main features provided by BART:

- Supports complete, hot, physical backups of multiple Advanced Servers and PostgreSQL database servers
- Provides two types of backups – full base backups and block-level incremental backups
- Provides backup and recovery management of the database servers on local or remote hosts
- Uses a single, centralized catalog for backup data
- Provides retention policy support for defining and managing how long backups should be kept
- Provides the capability to store the backup data in compressed format
- Verifies backup data with checksums
- Displays backup information in an easy-to-read format
- Simplifies the point-in-time recovery process

The following sections provide the information needed to use BART:

- Section 2 provides an overview of the BART components and concepts.
- Section 3 describes the backup and recovery management process using BART.
- Section 4 provides information about using tablespaces.
- Section 5 provides a comprehensive example of both local and remote database server configuration and operation.
- Section 6 provides information about recent remediations.

The remaining sections in this section describe basic conventions used throughout this document.

1.1 What's New

The following features have been added in BART 2.4:

- You can set the `archive_command` in the PostgreSQL server based on the `archive_command` setting in `bart.cfg` when you invoke the `INIT` subcommand if `archive_mode` is set to `off` and `archive_command` is not set in the PostgreSQL server.
- You can use the `--no-configure` option with the `INIT` subcommand, which prevents the `archive_command` option from being set in the PostgreSQL server.
- You can now take multiple backups of PostgreSQL server versions 9.6 simultaneously. If a backup is interrupted, it can be terminated automatically without the need to run the `pg_stop_backup`() manually.
- A new `worker` parameter is introduced to specify the number of parallel worker processes required to stream the modified blocks of an incremental backup to the restore host. The default value is set to 1.
- BART supports Native Ubuntu 18.04 (Bionic) and Debian 9.x (Stretch) .deb packages.

BART 2.4 Documentation Improvements

- BART 2.4 now has a Quickstart guide. The guide enables users to install and configure BART, and take a full and incremental backup quickly. For details, please see the EDB Postgres Backup and Recovery Quickstart Guide.
- BART installation is now documented in a separate guide titled EDB Postgres Backup and Recovery Installation and Upgrade Guide to improve readability. Refer the EDB Postgres Backup and Recovery Installation and Upgrade Guide for details.

1.2 Typographical Conventions Used in this Guide

Certain typographical conventions are used in this manual to clarify the meaning and usage of various commands, statements, programs, examples, etc. This section provides a summary of these conventions.

In the following descriptions a *term* refers to any word or group of words that are language keywords, user-supplied values, literals, etc. A term's exact meaning depends upon the context in which it is used.

- *Italic font* introduces a new term, typically, in the sentence that defines it for the first time.
- `Fixed-width (mono-spaced) font` is used for terms that must be given literally such as SQL commands, specific table and column names used in the examples, programming language keywords, etc. For example, `SELECT * FROM emp;`
- *`Italic fixed-width font`* is used for terms for which the user must substitute values in actual usage. For example, `DELETE FROM` *`table_name`*;
- A vertical pipe | denotes a choice between the terms on either side of the pipe. A vertical pipe is used to separate two or more alternative terms within square brackets (optional choices) or braces (one mandatory choice).
- Square brackets [ ] denote that one or none of the enclosed term(s) may be substituted. For example, `[ a | b ]`, means choose one of "`a`" or "`b`" or neither of the two.
- Braces {} denote that exactly one of the enclosed alternatives must be specified. For example, `{ a | b }`, means exactly one of "`a`" or "`b`" must be specified.
- Ellipses ... denote that the preceding term may be repeated. For example, `[ a | b ] ...` means that you may have the sequence, "`b a a b a`".

1.3 Other Conventions Used in this Guide

The following is a list of other conventions used throughout this document.

- Much of the information in this document applies interchangeably to the PostgreSQL and EDB Postgres Advanced Server database systems. The term *Advanced Server* is used to refer to EDB Postgres Advanced Server. The term *Postgres* is used to generically refer to both PostgreSQL and Advanced Server. When a distinction needs to be made between these two database systems, the specific names, PostgreSQL or Advanced Server are used.
- The installation directory path of the PostgreSQL or Advanced Server products is referred to as *POSTGRES_INSTALL_HOME*. For PostgreSQL Linux installations, this defaults to `/opt/PostgreSQL/x.x` for version 10 and earlier. For later versions, use the PostgreSQL community packages. For Advanced Server Linux installations accomplished using the interactive installer for version 10 and earlier, this defaults to `/opt/PostgresPlus/x.x`AS or `/opt/edb/asx.x`. For Advanced Server Linux installations accomplished using an RPM package, this defaults to `/usr/ppas-x.x` or `/usr/edb/asx.x`. The product version number is represented by *x.x* or by *xx* for version 10 and later. For Advanced Server Linux installations accomplished using an RPM package for version 11, this defaults to `/usr/edb/as11`.

1.3.1 Restrictions on pg_basebackup

BART takes full backups using the `pg_basebackup` utility program under the following conditions:

- The backup is taken on a standby server
- The --with-`pg_basebackup` option has been specified with the BACKUP subcommand for forcing the usage of `pg_basebackup` (see Section 3.4.3).
- The number of thread counts in effect is 1, and the - `pg_basebackup` option is not specified with the BACKUP subcommand. See the configuration section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about setting the `thread_count` parameter.

In the global section of the BART configuration file, parameter `pg_basebackup_path` specifies the complete directory path to the `pg_basebackup` program. See the configuration section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about this parameter.

For information about `pg_basebackup` see the *PostgreSQL Core Documentation* available at:

https://www.postgresql.org/docs/11/static/app-pgbasebackup.html

There are restrictions on using `pg_basebackup` depending upon the `pg_basebackup` version and the Postgres database server version.

Database servers can only be backed up by using `pg_basebackup` of the same or later version than the database server version. For example, `pg_basebackup` version 9.5 can back up database server version 9.5, but it cannot be used to back up database server version 9.6.

2 Overview

BART provides a simplified interface for the continuous archiving and point-in-time recovery method provided with Postgres database servers. This consists of the following processes:

- Capturing a complete image of a database cluster as a full base backup or referred to simply as a *full backup*
- Capturing a modified image of a database cluster called a *block-level incremental backup*, which is similar to a full backup, but contains the modified blocks of the relation files that have been changed since a previous backup instead of all, full relation files
- Archiving the *Write-Ahead Log segments* (WAL files), which continuously record changes to be made to the database files
- Performing *Point-In-Time Recovery* (PITR) to a specified transaction ID or timestamp with respect to a timeline using a full backup along with successive, block-level incremental backups that reside in the same backup chain, and the WAL files

Detailed information regarding WAL files and point-in-time recovery is documented in the *PostgreSQL Core Documentation* available at:

https://www.postgresql.org/docs/11/static/continuous-archiving.html

Block-level incremental backups are referred to simply as *incremental backups*.

The general term *backup* refers to both full backups and incremental backups. When a distinction must be made between the two, the complete term *full backup* or *incremental backup* is used.

Note: For standby servers, only a full backup can be taken. Incremental backups cannot be taken from standby servers. For information about standby servers, see the *PostgreSQL Core Documentation* available at:

https://www.postgresql.org/docs/11/static/high-availability.html

When taking a full backup of a standby server, BART uses the PostgreSQL `pg_basebackup` utility program.

For information about `pg_basebackup`, see the *PostgreSQL Core Documentation* available at:

https://www.postgresql.org/docs/11/static/app-pgbasebackup.html

These features provide a complete backup and recovery methodology for Postgres database servers, however, the management of this process can be quite complex, especially when dealing with multiple database servers in a distributed environment.

BART simplifies this management process by use of a centralized backup catalog, a single configuration file, and a command line interface controlling the necessary operations.

Reasonable defaults are automatically used for various backup and restore options. BART also performs the necessary recovery file configuration required for point-in-time recovery by means of its command line interface.

BART also provides the following features to enhance backup management:

- Automation of the WAL archiving command configuration
- Usage of retention policies to evaluate, categorize, and delete obsolete backups
- Compression of WAL files to conserve disk space
- Customizable naming of backups to ease their usage
- Easy access to comprehensive information about each backup

The key components for using BART are the following:

- **BART Host.** The host system on which BART is installed. BART operations are invoked from this host system. The database server backups and archived WAL files are stored on this host as well.
- **BART User Account.** Linux operating system user account you choose to run BART. The BART user account owns the BART backup catalog directory.
- **BART Configuration File.** File in editable text format containing the configuration information used by BART.
- **BART Backup Catalog.** File system directory structure containing all of the backups and archived WAL files for the database servers managed by BART.
- **BART Backupinfo File.** File in text format containing information for a BART backup. A `backupinfo` file resides in each backup subdirectory within the BART backup catalog.
- **BART Command Line Utility Program.** Single, executable file named `bart`, which is used to commence all BART operations.
- **BART WAL Scanner Program.** Single, executable file named `bart-scanner`, which is used to scan WAL files to locate and record the modified blocks for incremental backups.

Other concepts and terms referred to in this document include the following:

- **Postgres Database Cluster.** Also commonly called the *data directory*, this is the file system directory where all of the data files related to a particular Postgres database server instance are stored. (Each specific running instance is identified by its host and

port number when connecting to a database.) The database cluster is identified by the `-D` option when it is created, started, stopped, etc. by the `Postgres initdb` and `pg_ctl` commands. Typically by default, the initial database cluster is located in directory *`POSTGRES_INSTALL_HOME`*`/data`. A full backup is a copy of a database cluster.
Note: The terms database cluster and database server are used somewhat interchangeably throughout this document, though a single database server can run multiple database clusters.

- **Postgres User Account.** Linux operating system user account that runs the Advanced Server or PostgreSQL database server and owns the database cluster directory.
    - By default, the database user account is `enterprisedb` when Advanced Server is installed to support compatibility with Oracle databases.
    - By default, the database user account is `postgres` when Advanced Server is installed in PostgreSQL compatible mode. For a PostgreSQL database server, the default database user account is also `postgres`.
      Note: The BART configuration parameter `cluster_owner` must be set to the database user account for each database server. See the configuration section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information.
- **Replication Database User.** For each database server managed by BART, a database superuser must be selected to act as the replication database user. This database user is used to connect to the database server when backups are taken. The database superusers created with an initial Postgres database server installation (`enterprisedb or postgres`) may be used for this purpose.
  Note: The BART configuration parameter user must be set to this replication database `user` for each database server. See the configuration section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information.
- **Secure Shell (SSH)/Secure Copy (SCP).** Linux utility programs used to log into hosts (SSH) and copy files (SCP) between hosts. A valid user account must be specified that exists on the target host and in effect, is the user account under which the SSH or SCP operations occur.

Configuration section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* provides information on how all of these components are configured and used with BART.

2.1 Block-Level Incremental Backup

This section describes the basic concepts of the block-level incremental backup referred to simply as an incremental backup. An incremental backup is a unique utility for BART.

An incremental backup provides a number of advantages when compared to using full backups:

- The amount of time required to produce an incremental backup is generally less than a full backup as modified relation blocks are saved instead of all full relation files of the database cluster.
- An incremental backup uses less disk space than a full backup taken in plain text format.
  Note: If the full backup is taken in tar format, this saves disk space as well.

Generally, all BART features such as retention policy management apply to incremental backups as well as full backups. See Section 3.2.5 for information about retention policy management as applied to incremental backups.

2.1.1 Incremental Backup Limitations and Requirements

The following limitations apply to incremental backup:

- If you have restored a full or incremental backup, you must take a full backup before enabling incremental backup.
- If a standby node has been promoted to the role of primary node, you must take a full backup before enabling incremental backup on the cluster.
- An incremental backup cannot be taken on a standby database server; only a full backup can be taken from a standby database server.

The following requirements must be met before implementing incremental backup:

- Create or select an operating system account to be used as the BART user account. See Establishing the BART User Account Section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about choosing and setting up the BART user account.
- Create or select the replication database user, which must be a superuser. See Setting up a Replication Database User Section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information.
- In the BART configuration file, the `cluster_owner` parameter must be set to the Linux operating system user account that owns the directory of the database cluster from which incremental backups are to be taken. The `allow_incremental_backups` parameter must be enabled. See Configuring the Database Server Section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information.
- A password-less SSH/SCP connection must be established between the BART user account on the BART host and the `cluster_owner` user account on the database server.
  Note: This password-less SSH/SCP connection must be established even if BART and the database server are running on the same host and the BART user account and the `cluster_owner` user account are the same account. See Authorizing SSH/SCP Access without a Password Section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information.
- In addition to the BART host where the BART backup catalog resides, the BART package must also be installed on every remote database server on which incremental backups are to be restored. In order to restore an incremental backup, the `bart` program must be executable on the remote host by the remote user specified by the `remote_host` parameter in the BART configuration file or by the `-r` option when using the `RESTORE` subcommand to restore the incremental backup. See Section 2.1.5.2 for information about restoring incremental backups on remote hosts.
- For restoring incremental backups on a remote database server, a password-less SSH/SCP connection must be established from the BART user account on the

BART host to the remote user on the remote host that is specified by the `remote_host` parameter in the BART configuration file or by the `-r` option when using the `RESTORE` subcommand to restore the incremental backup. See Section 2.1.5.2 for information about restoring incremental backups on remote hosts.

- Compression of archived WAL files in the BART backup catalog is not permitted for database servers on which incremental backups are to be taken. The `wal_compression` setting in the BART configuration file must not be enabled for those database servers. Disabled is the default setting unless the parameter is altered in the global section or the server section. See Configuring the BART host and Configuring the Database Server Sections of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about the `wal_compression` parameter.
- The incremental backup must be on the same timeline as the parent backup. The timeline changes after each recovery operation so an incremental backup cannot use a parent backup from an earlier timeline.

The following section provides an overview of the basic incremental backup concepts.

2.1.2 Concept Overview

Using incremental backups involves the following sequence of steps:

1. Configure BART. See EDB Postgres Backup and Recovery Installation and Upgrade Guide for details.
2. Archiving of WAL files to the BART backup catalog must be enabled and initiated in the same manner as done for full backups.
3. An initial full backup must be taken with the `BACKUP` subcommand. This full backup establishes the parent of the first incremental backup.
4. All WAL files produced by database servers on which incremental backups are to be taken must be scanned. These WAL files are scanned once they have been archived to the BART backup catalog. Each scanned WAL file results in a modified block map (MBM) file that records the location of modified blocks obtained from the corresponding WAL file. The BART WAL scanner program `bart-scanner` performs this process.
5. Incremental backups are taken using the `BACKUP` subcommand with the `--parent` option to specify the backup identifier or name of a previous, full backup or an incremental backup. Any previous backup may be chosen as the parent as long as all backups belong to the same timeline.
6. The incremental backup process identifies which WAL files may contain changes from when the parent backup was taken to the starting point of the incremental backup. The corresponding MBM files are used to locate and copy the modified blocks to the incremental backup directory along with other database cluster directories and files. Instead of backing up all, full relation files, only the modified blocks are copied and saved. In addition, the relevant MBM files are condensed into one consolidated block map (CBM) file that is stored with the incremental backup.
   Note: Multiple block copier threads can be used to copy the modified blocks to the incremental backup directory. See Configuring the BART host and Configuring the Database Server Sections of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for setting the `thread_count` parameter in the BART configuration file. See Section 3.4.3 for using the `--thread-count` option with the `BACKUP` subcommand.
7. The restore process for an incremental backup is invoked using the `RESTORE` subcommand in the same manner as restoring a full backup. The `-i` option specifies the backup identifier or name of the incremental backup to restore. The restore process begins by going back through the chain of past, parent incremental backups until the initial full backup starting the chain is identified. This full backup provides the initial set of directories and files to be restored to the location specified with the `-p` option.
8. Each subsequent incremental backup in the chain is then restored. Restoration of an incremental backup uses its CBM file to restore the modified blocks from the incremental backup.

The following sections provide some additional information on these procedures.

2.1.3 WAL Scanning – Preparation for an Incremental Backup

The WAL files created from the time of the parent backup up to the start of the incremental backup are scanned by the WAL scanner program `bart-scanner` to determine which blocks have changed since the parent backup.

The WAL scanner determines which blocks have been modified and records the information in a file called the *modified block map* (MBM) file. One MBM file is created for each WAL file.

The MBM file is stored in the archive path directory:

backup_path/*server_name*/archived_wals

Where:

backup_path is the BART backup catalog parent directory specified in the global section of the BART configuration file.

server_name is the lowercase conversion of the database server name specified for this database server in the server section of the BART configuration file.

This is the same directory where the archived WAL files are stored in the BART backup catalog.

The following 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 comparable manner.)

```
[root@localhost archived_wals]# pwd
/opt/backup/acctg/archived_wals
[root@localhost archived wals]# ls -l
total 131104
-rw------- 1 ... ... 16777216 Oct 12 09:38 000000010000000100000078
-rw------- 1 ... ... 16777216 Oct 12 09:38 000000010000000100000079
-rw------- 1 ... ... 16777216 Oct 12 09:38 00000001000000010000007A
-rw------- 1 ... ... 16777216 Oct 12 09:35 00000001000000010000007B
-rw------- 1 ... ... 16777216 Oct 12 09:38 00000001000000010000007C
-rw------- 1 ... ... 16777216 Oct 12 09:39 00000001000000010000007D
-rw------- 1 ... ... 16777216 Oct 12 09:42 00000001000000010000007E
-rw------- 1 ... ... 16777216 Oct 12 09:47 00000001000000010000007F
-rw-rw-r-- 1 ... ...      161 Oct 12 09:49 000000010000000178000028000000179000000.mbm
-rw-rw-r-- 1 ... ...      684 Oct 12 09:49 00000001000000017900002800000017A000000.mbm
-rw-rw-r-- 1 ... ...      161 Oct 12 09:49 0000000100000017A000028000000017B000000.mbm
-rw-rw-r-- 1 ... ...      161 Oct 12 09:49 0000000100000017B000028000000017C000000.mbm
-rw-rw-r-- 1 ... ...     1524 Oct 12 09:49 0000000100000017C000028000000017D000000.mbm
-rw-rw-r-- 1 ... ...      161 Oct 12 09:49 0000000100000017D000028000000017E000000.mbm
-rw-rw-r-- 1 ... ...      161 Oct 12 09:49 0000000100000017E000028000000017F000000.mbm
-rw-rw-r-- 1 ... ...      161 Oct 12 09:49 0000000100000017F000028000000180000000.mbm
```

The MBM files have the suffix, `.mbm`.

In preparation for any incremental backup, the WAL files should be scanned as soon as they are copied to the BART backup catalog. Thus, the WAL scanner should be running as soon as the WAL files from the database cluster are archived to the BART backup catalog.

If the BART backup catalog contains WAL files that have not yet been scanned, starting the WAL scanner begins scanning these files. If WAL file fails to be scanned (resulting in a missing MBM file), you can use the WAL scanner to specify an individual WAL file.

Note: Under certain circumstances such as when the `rsync` utility is used to copy WAL files to the BART backup catalog, the WAL files may have been missed by the WAL scanner program for scanning and creation of MBM files. Use the `scan_interval` parameter in the BART configuration file to force scanning of WAL files in the archive directory of the BART backup catalog to ensure MBM files are generated. See Configuring the BART host and Configuring the Database Server Sections of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for more information.

See Section 3.5 for information about using the WAL scanner.

2.1.4 Performing an Incremental Backup

The WAL files produced at the time of the parent backup up to the start of the incremental backup contain information about which blocks were modified during that time interval. That information is consolidated into an MBM file for each WAL file by the WAL scanner.

The MBM files for the relevant WAL files are read, and the information is used to copy the modified blocks from the database cluster to the BART backup catalog. When compared to a full, plain backup, the number and sizes of relation files can be significantly less for an incremental backup.

For comparison, the following is an abbreviated list of the files copied to the archived `base` subdirectory of a full backup for one database:

```
[root@localhost 14845]# pwd
/opt/backup/acctg/1476301238969/base/base/14845
[root@localhost 14845]# ls
112         13182_vm    14740  16467  16615     2608_vm    2655  2699      2995      ...
113         13184       14742  16471  174       2609       2656  2701      2995_vm   ...
1247        13186       14745  16473  175       2609_fsm   2657  2702      2996      ...
1247_fsm    13187       14747  16474  2187      2609_vm    2658  2703      2998      ...
1247_vm     13187_fsm   14748  16476  2328      2610       2659  2704      2998_vm   ...
1249        13187_vm    14749  16477  2328_fsm  2610_fsm   2660  2753      2999      ...
1249_fsm    13189       14752  16479  2328_vm   2610_vm    2661  2753_fsm  2999_vm   ...
1249_vm     13191       14754  16488  2336      2611       2662  2753_vm   3079      ...
1255        13192       14755  16490  2336_vm   2611_vm    2663  2754      3079_fsm  ...
                                          .
                                          .
                                          .
13182_fsm   14739       16465  16603  2608_fsm  2654       2696  2893_vm   3501_vm   ...
```

In contrast, the following is the content of the archived `base` subdirectory of the same database from a subsequent incremental backup:

```
[root@localhost 14845]# pwd
/opt/backup/acctg/1476301835391/base/base/14845
[root@localhost 14845]# ls
1247      1249      1259      16384  17006  2608      2610      2658  2663  2678  ...
1247_fsm  1249_fsm  1259_fsm  16387  17009  2608_fsm  2610_fsm  2659  2673  2679  ...
1247_vm   1249_vm   1259_vm   16389  17011  2608_vm   2610_vm   2662  2674  2703  ...
```

The information from the MBM files are consolidated into one file called a *consolidated block map* (CBM) file. During the restore operation for the incremental backup, the CBM file is used to identify the modified blocks to be restored for that backup.

In addition, the incremental backup also stores other required subdirectories and files from the database cluster as is done for full backups.

See Section 3.4.3 for information about using the `BACKUP` subcommand to take an incremental backup.

2.1.5 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:

- **BART Host.** If an incremental backup is to be restored onto the same host where BART has been installed, the restore process is outlined in Section 2.1.5.1.
- **Remote Host.** If an incremental backup is to be restored onto a remote host where BART has not been installed, the restore process still follows the steps outlined in Section 2.1.5.1, but in addition, the requirements in Section 2.1.5.2 must be met.

The `bart` program must be available on the remote host because the `RESTORE` subcommand invocation for an incremental backup results in the execution of the `bart` program on the remote host to restore the modified blocks to their proper location within the restore directory.

The following section describes the general incremental backup restore process.

2.1.5.1 General Restore Process for an Incremental Backup

Specify a backup identifier or name, and include the `-i` option when invoking the `RESTORE` subcommand to restore an incremental backup. All `RESTORE` options are used in the same manner as when restoring a full backup.

First, all files from the full backup from the beginning of the backup chain are restored.

For each incremental backup, the CBM file is used to identify and restore blocks from the incremental backup.

If there are new relations or databases identified in the CBM file, then relevant relation files are copied. If consolidated block map information is found indicating the drop of a relation or a database, then the relevant files are removed from the restore directory. Similarly if there is any indication of a table truncation, then the related files are truncated.

See Section 3.4.8 for information about using the `RESTORE` subcommand for restoring an incremental backup. Also note the usage of the `-w` option of the `RESTORE` subcommand to specify a multiple number of parallel worker processes to stream the modified blocks to the restore host.

2.1.5.2 Restoring an Incremental Backup on a Remote Host

If an incremental backup is to be restored on a remote host on which BART has not been installed, you must perform the following steps.

Step 1: Install `BART` on the remote host to which an incremental backup is to be restored. See *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for instructions on how to install `BART`.

Note: No editing is needed in the `bart.cfg` file installed on the remote host.

Step 2: Determine the Linux operating system user account on the remote host to be used as the remote user. This user is specified by the `remote_host` parameter in the BART configuration file or by the `-r` option when using the `RESTORE` subcommand to restore the incremental backup. The remote user must be the owner of the directory where the incremental backup is to be restored on the remote host. By default, the user account is `enterprisedb` for Advanced Server or `postgres` for PostgreSQL.

Step 3: Ensure a password-less SSH/SCP connection is established from the BART user on the BART host to the remote user on the remote host. See the configuration section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for details.

2.1.6 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.

It is possible to have a *multi-forked* backup chain, which is two or more successive lines of incremental backups, all of which begin with the same, full backup. Thus, within the chain there is a backup that serves as the parent of more than one incremental backup.

Since restoration of an incremental backup is dependent upon first restoring the full backup, then all successive incremental backups up to, and including the incremental backup specified by the `RESTORE` subcommand, it is crucial to note the following:

- Deletion or corruption of the full backup destroys the entire backup chain. It is not possible to restore any of the incremental backups that were part of that chain.
- Deletion or corruption of an incremental backup within the chain results in the inability to restore any incremental backup that was added to that successive line of backups following the deleted or corrupted backup. The full backup and incremental backups prior to the deleted or corrupted backup can still be restored.

The actions of retention policy management are applied to the full backup and all of its successive incremental backups within the chain in an identical manner as if they were one backup. Thus, use of retention policy management does not result in the breakup of a backup chain. See Section 3.2.5 for information about retention policy management of incremental backups.

The following are some examples of backup chains.

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: 000000010000000000000E
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. See Section 3.4.3 for information about the `BACKUP` subcommand.

```
-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 ...
```

Note: 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 is done with the `RESTORE` subcommand and its options 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 by the following:

```
-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`.

See Section 3.4.8 for information about the `RESTORE` subcommand.

3 Using BART

Once you have installed and configured the BART host and the database servers, you can start using BART. See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for installation and configuration details.

https://www.enterprisedb.com/resources/product-documentation

This section describes how to perform backup and recovery management operations using BART.

- Section 3.1 presents the BART management process overview.
- Section 3.2 describes managing backups using a retention policy.
- Section 3.3 describes the basic usage of the BART command line program and its subcommands.
- Section 3.4 provides a description of each BART subcommand.
- Section 3.5 describes usage of the BART WAL scanner.

Review these sections before proceeding with any BART operation.

3.1 BART Management Overview

After performing the configuration process described, you can begin the BART backup and recovery management process.

First, perform the following steps:

1. Run the `CHECK-CONFIG` subcommand without the `-s` option. When used without specifying the `-s` option, the `CHECK-CONFIG` subcommand checks the parameters in the global section of the BART configuration file.
2. If you have not already done so, run the `INIT` subcommand to finish creation of the BART backup catalog, which results in the complete directory structure to which backups and WAL files are saved. This step must be done before restarting the database servers with enabled WAL archiving, otherwise the copy operation in the `archive_command` parameter of the `postgresql.conf` file or the `postgresql.auto.conf` file fails due to the absence of the target archive directory. When the directory structure is complete, the lowest level subdirectory named *`server_name`*`/archived_wals`, referred to as the *archive path*, should exist for each database server.
3. Start the Postgres database servers with archiving enabled. Verify that the WAL files are appearing in the *`server_name`*`/archived_wals` archive paths for each database server. (The archiving frequency is dependent upon other `postgresql.conf` configuration parameters.) Check the Postgres database server log files to ensure there are no archiving errors. Archiving should be

operational before taking a backup in order to ensure that the WAL files that may be created during the backup process are archived.

4. If you intend to take incremental backups, start the WAL scanner. Since the WAL scanner processes the WAL files copied to the archive paths in the BART backup catalog, it is advantageous to commence the WAL scanning as soon as the WAL files begin to appear in the BART backup catalog in order to keep the scanning in pace with the WAL archiving.
5. Run the BART `CHECK-CONFIG` subcommand for each database server with the `-s` option specifying the server name. This ensures the database server has been properly configured for taking backups.
6. Create a full backup for each database server. The full backup establishes the starting point of when point-in-time recovery can begin and also establishes the initial parent backup for any incremental backups to be taken.

There are now a number of other BART management processes you may perform:

- Verify the checksum of the full backups using the `VERIFY-CHKSUM` subcommand.
- Display database server information with the `SHOW-SERVERS` subcommand.
- Display backup information with the `SHOW-BACKUPS` subcommand.
- Perform the `BACKU`P subcommand to create additional full backups or incremental backups.
- Compress the archived WAL files in the BART backup catalog by enabling WAL compression in the BART configuration file and then invoking the `MANAGE` subcommand.
- Determine and set the retention policy for backups in the BART configuration file.
- Establish the procedure for using the `MANAGE` subcommand to enforce the retention policy for backups. This may include using cron jobs to schedule the `MANAGE` subcommand.

3.1.1 Point-In-Time Recovery Operation

The following steps outline how to perform a point-in-time recovery operation for a database cluster:

1. Use your system-specific command to shut down the database server.
2. Decide if you want to restore the database cluster and tablespace files to (a) new directories or (b) to reuse the existing database cluster directories.
   a. For option (a), create the new directories with the appropriate directory ownership and permissions.
   b. For option (b), delete all the files and subdirectories in the existing directories, but it is strongly recommended that you make a copy of this data before deleting it. Be sure to save any recent WAL files in the `pg_xlog` subdirectory

that have not been archived to the BART backup catalog *server_name*/archived_wals subdirectory.

3. Run the BART SHOW-BACKUPS -s *server_name* subcommand to list the backup IDs and backup names of the backups for the database server. You will need to supply the appropriate backup ID or backup name with the BART RESTORE subcommand, unless you intend to restore the latest backup in which case the -i option of the RESTORE subcommand for specifying the backup ID or backup name may be omitted.

4. Run the BART RESTORE subcommand with the appropriate options.

- The backup is restored to the directory specified by the -p *restore_path* option.
- In addition, if the RESTORE subcommand -c option is specified or if the enabled setting of the copy_wals_during_restore BART configuration parameter is applicable to the database server, then the required, archived WAL files from the BART backup catalog are copied to the *restore_path*/archived_wals subdirectory.
  Note: Ensure the *restore_path* directory and its subdirectories and files are owned by the proper Postgres user account (for example, enterprisedb or postgres). Also ensure that only the Postgres user account has access permission to the *restore_path* directory. Make the appropriate adjustments wherever needed such as with the command, chown -R enterprisedb:enterprisedb *restore_path* or chmod 700 *restore_path*.

5. Copy any saved WAL files from Step 2 that were not archived to the BART backup catalog to the *restore_path*/pg_xlog subdirectory.
6. If generated for point-in-time recovery, verify that the recovery.conf file created in the directory specified with the RESTORE subcommand's -p *restore_path* option was generated with the correct recovery parameter settings.
   Note: If the RESTORE subcommand -c option is specified or if the enabled setting of the copy_wals_during_restore BART configuration parameter is applicable to the database server, then the restore_command parameter retrieves the archived WAL files from the *restore_path*/archived_wals subdirectory that was created by the RESTORE subcommand, otherwise the restore_command retrieves the archived WAL files from the BART backup catalog.
7. The BART RESTORE subcommand disables WAL archiving in the restored database cluster. If you want to immediately enable WAL archiving, modify the postgresql.conf file by deleting the archive_mode = off parameter that BART appends to the end of the file.
8. Start the database server, which will then perform the point-in-time recovery operation if a recovery.conf file was generated.

See Section 3.4 for a detailed description of the BART subcommands.

3.2 Managing Backups Using a Retention Policy

Over the course of time when using BART, the number of backups can grow significantly. This ultimately leads to a large consumption of disk space unless an administrator periodically performs the process of deleting the oldest backups that are no longer needed. This process of determining when a backup is old enough to be deleted and then actually deleting such backups can be accomplished and eventually automated with the following basic steps:

1. Determine and set a retention policy in the BART configuration file. A *retention policy* is a rule that determines when a backup is considered obsolete. The retention policy can be applied globally to all servers, but each server can override the global retention policy with its own (see the configuration section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for details).
2. Use the `MANAGE` subcommand to categorize and manage backups according to the retention policy. Such functionality includes determining which active backups should be considered obsolete at this current point in time, selecting backups to keep indefinitely, and physically deleting obsolete backups. When an obsolete backup is deleted, its backup taken with the `BACKUP` subcommand along with their backup's archived WAL files are deleted.
3. Once the retention policies have been determined and verified, you can create a cron job to periodically run the `MANAGE` subcommand to evaluate the backups and then list and/or delete the obsolete backups.

There is a difference on how retention policy management applies to incremental backups as compared to full backups. See Section 3.2.5 for information about how retention policy management applied to full backups affects incremental backups.

The following sections describe how retention policy management generally applies to backups, and its specific usage and effect on full backups:

- Section 3.2.1 provides an overview of the terminology and types of retention policies.
- Section 3.2.2 describes the concept of marking the status of backups according to the retention policy.
- Section 3.2.3 describes setting the different types of retention policies.
- Section 3.2.4 describes the process of managing the backups such as marking the backup status, keeping selected backups indefinitely, listing obsolete backups, and deleting obsolete backups.

Note: The examples shown in the previously listed sections were generated with BART version 1.1. The retention policy management process is the same for the current BART version, however the displayed output of the `SHOW-BACKUPS` and `SHOW-SERVERS`

subcommands now include a few additional fields that do not influence the retention policy.

3.2.1 Overview

The BART retention policy results in the categorization of each backup in one of three statuses – *active*, *obsolete*, and *keep*:

- **Active.** The backup satisfies the retention policy applicable to its server. Such backups would be considered necessary to ensure the recovery safety for the server and thus should be retained.
- **Obsolete.** The backup does not satisfy the retention policy applicable to its server. The backup is no longer considered necessary for the recovery safety of the server and thus can be deleted.
- **Keep.** The backup is to be retained regardless of the retention policy applicable to its server. The backup is considered vital to the recovery safety for the server and thus should not be deleted for an indefinite period of time.

There are two types of retention policies:

- **Redundancy Retention Policy.** The *redundancy retention policy* relies on a specified, maximum number of most recent backups to retain for a given server. When the number of backups exceeds that maximum number, the oldest backups are considered obsolete (except for backups marked as keep). Section 3.2.3.1 describes this type of retention policy.
- **Recovery Window Retention Policy.** The *recovery window retention policy* relies on a time frame (the recovery window) for when a backup should be considered active. The boundaries defining the recovery window are the current date/time (the ending boundary of the recovery window) and the date/time going back in the past for a specified length of time (the starting boundary of the recovery window). If the date/time the backup was taken is within the recovery window (that is, the backup date/time is on or after the starting date/time of the recovery window), then the backup is considered active, otherwise it is considered obsolete (except for backups marked as keep). Section 3.2.3.2 describes this type of retention policy.

Thus, for the recovery window retention policy, the recovery window time frame dynamically shifts, so the end of the recovery window is always the current date/time when the `MANAGE` subcommand is run. As you run the `MANAGE` subcommand at future points in time, the starting boundary of the recovery window moves forward in time.

At some future point, the date/time of when a backup was taken will be earlier than the starting boundary of the recovery window. This is when an active backup's status will then be considered obsolete.

You can see the starting boundary of the recovery window at any point in time by running the `SHOW-SERVERS` subcommand. The `RETENTION POLICY` field of the `SHOW-SERVERS` subcommand displays the starting boundary of the recovery window.

3.2.2 Marking the Backup Status

When a backup is initially created with the `BACKUP` subcommand, it is always recorded with active status.

It is only by using the `MANAGE` subcommand at a particular point in time that active backups are evaluated to determine if their status should be changed to obsolete in accordance with the retention policy.

In addition, it is only when the `MANAGE` subcommand is invoked either with no options or with only the `-s` option (in order to specify the database server) that active backups are evaluated and also *marked* (that is, internally recorded by BART) as obsolete. See Section 3.4.7 for information about the `MANAGE` subcommand usage with its options.

Once a backup has been marked as obsolete, it cannot be changed back to active unless you perform the following steps:

- Using the `MANAGE` subcommand, specify the `-c` option along with the backup identifier or name with the `-i` option. If you wish to keep this particular backup indefinitely, use `-c keep`, otherwise use `-c nokeep`.

- If you use the `-c nokeep` option, the backup status is changed back to active. When the `MANAGE` subcommand is used the next time, the backup is re-evaluated to determine if its status needs to be changed back to obsolete based on the current retention policy in the BART configuration file.

Note: If the `retention_policy` parameter is set in a certain manner, you run the `MANAGE` subcommand to mark the backups according to that `retention_policy` setting, and then you change or disable the `retention_policy` parameter by commenting it out, the current, marked status of the backups are probably inconsistent with the current `retention_policy` setting.

If you wish to modify the backup status to be consistent with the current `retention_policy` setting, then perform the following:

- If there are backups currently marked as obsolete that would no longer be considered obsolete under a new retention policy, run the `MANAGE` subcommand with the `-c nokeep` option for such backups to change the obsolete status to active status. You can also specify the `-i all` option, which would change all backups back to active status, including those currently marked as keep.

- Run the `MANAGE` subcommand either with no options or with only the `-s` option to reset the marked status based on the new `retention_policy` setting in the BART configuration file.

Deletion of backups with the `MANAGE -d` option relies on the last occasion when the backups have been marked.

The current marking (the status) of the backups can be viewed with the `SHOW-BACKUPS` subcommand.

3.2.3 Setting the Retention Policy

The retention policy is determined by the `retention_policy` parameter in the BART configuration file. For information about creating a global retention policy (that applies to all servers), see Configuring the BART host section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide*. For information about creating a retention policy that applies to an individual database server, see Configuring the Database Server of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide*.

The two types of retention policies are the redundancy retention policy described in Section 3.2.3.1 and the recovery window retention policy described in Section 3.2.3.2.

3.2.3.1 Redundancy Retention Policy

To use the redundancy retention policy, set `retention_policy = max_number BACKUPS` where *`max_number`* is a positive integer designating the maximum number of most recent backups.

The following are some additional restrictions:

- The keyword `BACKUPS` must always be specified in plural form (for example, `1 BACKUPS`).
- BART will accept a maximum integer value of 2,147,483,647 for *`max_number`*; however, you should use a realistic, practical value based on your system environment.

The redundancy retention policy is the default type of retention policy if all keywords `BACKUPS`, `DAYS`, `WEEKS`, and `MONTHS` following the *`max_number`* integer are omitted as shown by the following example:

```
retention_policy = 3
```

In the following example the redundancy retention policy setting considers the three most recent backups as the active backups. Any older backups, except those marked as keep, are considered obsolete.

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

The `SHOW-SERVERS` subcommand displays the 3 backup redundancy retention policy in the `RETENTION POLICY` field:

```
-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    : 3 Backups
DISK UTILIZATION    : 627.04 MB
NUMBER OF ARCHIVES  : 25
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"
```

3.2.3.2 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 in one of the following ways:

- Set to *`max_number`* `DAYS` to define the start date/time recovery window boundary as the number of days specified by *`max_number`* going back in time from the current date/time.
- Set to *`max_number`* `WEEKS` to define the start date/time recovery window boundary as the number of weeks specified by *`max_number`* going back in time from the current date/time.
- Set to *`max_number`* `MONTHS` to define the start date/time recovery window boundary as the number of months specified by *`max_number`* going back in time from the current date/time.

The following are some additional restrictions:

- The keywords `DAYS`, `WEEKS`, and `MONTHS` must always be specified in plural form (for example, `1 DAYS`, `1 WEEKS`, or `1 MONTHS`).
- BART will accept a maximum integer value of 2,147,483,647 for *`max_number`*, however, a realistic, practical value based on your system environment must always be used.

A backup is considered active if the date/time of the backup, down to a second accuracy, is equal to or greater than the start of the recovery window date/time.

There are a couple of ways you can view the actual, calculated recovery window as described by the following.

Invoking BART in debug mode (with the `-d` option) using the `MANAGE` subcommand with the `-n` option displays the calculated time length of the recovery window 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).

Note: 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.

Using the SHOW-SERVERS subcommand, the RETENTION POLICY field displays the start of the recovery window.

In the following example, 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 example, 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 example, 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"
```

The following section describes how to manage the backups based on the retention policy.

3.2.4 Managing the Backups

The MANAGE subcommand is used to evaluate and categorize backups according to the retention policy set in the BART configuration file. When a backup is first created with the BACKUP subcommand, it is always marked as active. Upon usage of the MANAGE subcommand, an active backup may be marked as obsolete. Obsolete backups can then be deleted.

The various aspects of backup management are discussed in the following sections:

- Section 3.2.4.1 discusses the rules for deleting backups dependent upon the backup status and the subcommand used.
- Section 3.2.4.2 shows how to retain a backup indefinitely by marking it as keep as well as resetting backups marked as obsolete and keep back to active status.
- Section 3.2.4.3 demonstrates the general process for evaluating, marking, then deleting obsolete backups.

3.2.4.1 Deletions Permitted Under a Retention Policy

This section describes how and under what conditions backups may be deleted under a retention policy.

You should use the MANAGE subcommand to delete obsolete backups. The DELETE subcommand should be used only for special administrative purposes.

The deletion behavior of the MANAGE subcommand and the DELETE subcommand are based on different aspects of the retention policy.

- The MANAGE subcommand deletion relies solely upon how a backup status is currently marked (that is, internally recorded by BART). The current setting of the retention_policy parameter in the BART configuration file is ignored.
- The DELETE subcommand relies solely upon the current setting of the retention_policy parameter in the BART configuration file. The current active, obsolete, or keep status of a backup is ignored.

The specific deletion rules for the MANAGE and DELETE subcommands are as follows:

- The MANAGE subcommand with the -d option can only delete backups marked as obsolete. This deletion occurs regardless of the current retention_policy setting in the BART configuration file.
- Under a redundancy retention policy currently set with the retention_policy parameter in the BART configuration file, any backup regardless of its marked status, can be deleted with the DELETE subcommand when the backup identifier

or name is specified with the `-i` option, and if the current total number of backups for the specified database server is greater than the maximum number of redundancy backups currently specified with the `retention_policy` parameter. If the total number of backups is less than or equal to the specified, maximum number of redundancy backups, then no additional backups can be deleted using `DELETE` with the `-i backup` option.

- Under a recovery window retention policy currently set with the `retention_policy` parameter in the BART configuration file, any backup regardless of its marked status, can be deleted with the `DELETE` subcommand when the backup identifier or name is specified with the `-i` option, and if the backup date/time is not within the recovery window currently specified with the `retention_policy` parameter. If the backup date/time is within the recovery window, then it cannot be deleted using `DELETE` with the `-i backup` option.
- Invoking the `DELETE` subcommand with the `-i all` option results in the deletion of all backups regardless of the retention policy and regardless of whether the status is marked as active, obsolete, or keep.

The following table summarizes the deletion rules of backups according to their marked status. An entry of "Yes" indicates the backup may be deleted under the specified circumstances. An entry of "No" indicates that the backup may not be deleted.

Table 3-1 Allowable Backup Deletion by Status

Operation	Redundancy Retention Policy			Recovery Window Retention Policy		
	Active	Obsolete	Keep	Active	Obsolete	Keep
`MANAGE -d`	No	Yes	No	No	Yes	No
`DELETE -i backup`	Yes (see Note 1)	Yes (see Note 1)	Yes (see Note 1)	Yes (see Note 2)	Yes (see Note 2)	Yes (see Note 2)
`DELETE -i all`	Yes	Yes	Yes	Yes	Yes	Yes

Note 1: Deletion occurs only if the total number of backups for the specified database server is greater than the specified, maximum number of redundancy backups currently set with the `redundancy_policy` parameter in the BART configuration file.

Note 2: Deletion occurs only if the backup is not within the recovery window currently set with the `redundancy_policy` parameter in the BART configuration file.

3.2.4.2 Marking Backups for Indefinite Keep Status

There may be certain backups that you wish to keep for an indefinite period of time and do not wish to delete based upon the retention policy applied to the database server. Such backups can be marked as keep to exclude them from being marked as obsolete.

Use the `MANAGE` subcommand with the `-c keep` option to retain such backups indefinitely as shown by the following example:

```
-bash-4.1$ bart MANAGE -s acctg -i 1428355371389 -c keep
INFO:  changing status of backup '1428355371389' of server 'acctg' from
'active' to 'keep'
INFO:  1 WAL file(s) changed
```

The backup status is now displayed as `keep`:

```
-bash-4.1$ bart SHOW-BACKUPS -s acctg -i 1428355371389 -t
SERVER NAME    : acctg
BACKUP ID      : 1428355371389
BACKUP NAME    : none
BACKUP STATUS  : keep
BACKUP TIME    : 2015-04-06 17:22:53 EDT
BACKUP SIZE    : 5.71 MB
WAL(S) SIZE    : 16.00 MB
NO. OF WALS    : 1
FIRST WAL FILE : 0000000100000000000000AA
CREATION TIME  : 2015-04-06 17:22:53 EDT
LAST WAL FILE  : 0000000100000000000000AA
CREATION TIME  : 2015-04-06 17:22:53 EDT
```

If at later point in time, you decide to return such a backup to a state where it can be evaluated for deletion based upon the retention policy, you must first remove its `keep` status by applying the `MANAGE` subcommand with the `-c nokeep` option as shown by the following example:

```
-bash-4.1$ bart MANAGE -s acctg -i 1428355371389 -c nokeep
INFO:  changing status of backup '1428355371389' of server 'acctg' from
'keep' to 'active'
INFO:  1 WAL file(s) changed
```

The backup status is changed back to `active`:

```
-bash-4.1$ bart SHOW-BACKUPS -s acctg -i 1428355371389 -t
SERVER NAME    : acctg
BACKUP ID      : 1428355371389
BACKUP NAME    : none
BACKUP STATUS  : active
BACKUP TIME    : 2015-04-06 17:22:53 EDT
BACKUP SIZE    : 5.71 MB
WAL(S) SIZE    : 16.00 MB
NO. OF WALS    : 1
FIRST WAL FILE : 0000000100000000000000AA
CREATION TIME  : 2015-04-06 17:22:53 EDT
LAST WAL FILE  : 0000000100000000000000AA
CREATION TIME  : 2015-04-06 17:22:53 EDT
```

The next time the `MANAGE` subcommand is invoked with either no options or with only the `-s` option, the active backup may be marked as obsolete according to the current `retention_policy` setting.

3.2.4.3 Evaluating, Marking, and Deleting Obsolete Backups

Based upon the current number of backups for the database server for a redundancy retention policy or the current date/time for a recovery window retention policy, when the

MANAGE subcommand is invoked it evaluates active backups for the database server specified by the -s option, or for all database servers if -s all is specified or the -s option is omitted.

Note: The status of backups currently marked as obsolete or keep are not changed. To re-evaluate such backups and then classify them, their status must first be reset back to active with the MANAGE -c nokeep option. See Section 3.2.2 for information.

This section illustrates the evaluation, marking, and deletion process with two examples – the first for a redundancy retention policy and the second for a recovery window retention policy.

Redundancy Retention Policy

The following example uses a redundancy retention policy 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 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 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:

```
-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 '000000010000000000000E1'
INFO:  removing WAL file '000000010000000000000E0'
```

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
```

Recovery Window Retention Policy

The following example uses a recovery window retention policy 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 by the following:

```
-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
```

3.2.5 Managing Incremental Backups

The following section describes how retention policy management affects incremental backups. In summary, the basic effects are the following:

- The retention policy rules are applied to full backups. Determining when a backup is obsolete by the redundancy retention policy is based solely on the number of **full backups**. All incremental backups are excluded from the comparison count against the `retention_policy` setting for the maximum number of backups. Determining when a backup is obsolete by the recovery window retention policy is based solely on the backup date/time of the **full backup**. The backup date/time of any successive incremental backups in the chain are ignored when comparing with the recovery window.
- The retention status of all incremental backups in a chain is set to the same status applied to the full backup of the chain.
- The actions applied by the `MANAGE` and `DELETE` subcommands on a full backup are applied to all incremental backups in the chain in the same manner.
- Thus, a backup chain (that is, the full backup and all its successive incremental backups) are treated by retention policy management as if they are all one, single backup. The status setting applied to the full backup is also applied to all incremental backups in its chain. If a full backup is marked as obsolete and then deleted according to the retention policy, all incremental backups in the chain are also marked obsolete and then deleted as well.

The following are some specific points regarding the `MANAGE` and `DELETE` subcommands on incremental backups:

- When the `MANAGE` subcommand is invoked, the status applied to the full backup is also applied to all successive incremental backups in the chain.
- The `MANAGE` subcommand with the `-c { keep | nokeep }` option cannot specify the backup identifier or backup name of an incremental backup with `-i` backup option. The `-i` backup option can only specify the backup identifier or backup name of a full backup. You can also use the `-i all` option to take a backup of all backups. When the `MANAGE` subcommand with the `-c { keep | nokeep }` option is applied to a full backup, the same status change is made to all incremental backups in the chain.
- The `DELETE` subcommand with the `-s server -i` backup option specifies the backup identifier or backup name of an incremental backup in which case that incremental backup along with all its successive incremental backups are deleted, thus shortening that backup chain.
- For examples of the redundancy retention policy on incremental backups, see Section 3.2.5.1.

For examples of the recovery window retention policy on incremental backups, see Section 3.2.5.2.

3.2.5.1 Redundancy Retention with Incremental Backups

When the redundancy retention policy is used and the `MANAGE` subcommand is invoked, the status of the oldest, active, full backups is changed to `obsolete` if the number of full backups exceeds the maximum number specified by the `retention_policy` parameter in the BART configuration file.

See Section 3.2.3.1 for information about the redundancy retention policy.

If a full backup is changed from active to obsolete, all successive incremental backups in the chain of this full backup are also changed from active to obsolete.

When determining the number of backups that exceeds the number specified by the `retention_policy` parameter, only full backups are counted for the comparison. The number of incremental backups is not included in the count for this comparison against the `retention_policy` parameter setting.

The following examples show usage of the `MANAGE` and `DELETE` subcommands when a 3 backup redundancy retention policy is in effect as shown by 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 following is the current set of backups. (In these examples, 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 example 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
```

Note: 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 example 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
```

The following section provides an example using the recovery window retention policy. Example usage of the `MANAGE` subcommand with other options along with usage of the `DELETE` subcommand are shown in this next section.

3.2.5.2 Recovery Window Retention with Incremental Backups

When the recovery window retention policy is used and the `MANAGE` subcommand is invoked, the status of active, full backups are changed to obsolete if the date/time of the full backup is outside of the recovery window (that is, the full backup date/time is prior to the start of the recovery window as defined by current date/time when the `MANAGE` subcommand is invoked, and then going back in time by the amount of time set by the `retention_policy` parameter in the BART configuration file).

See Section 3.2.3.2 for information about the recovery window retention policy.

If a full backup is changed from active to obsolete, all successive incremental backups in the chain of this full backup are also changed from active to obsolete.

The status of an incremental backup is changed to `obsolete` regardless of whether or not the date/time of when the incremental backup was taken still lies within the recovery window.

The following examples show usage of the `MANAGE` and `DELETE` subcommands when a 1-day recovery window retention policy is in effect as shown by 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 following is the current set of backups. (In these examples, 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 entire first backup chain is now marked `obsolete`.

Note: 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 example 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 example 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 example 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 example 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
```

3.3 Basic BART Subcommand Usage

Invoke the `bart` program (located in the *`BART_HOME`*`/bin`) directory with the desired subcommand and any applicable subcommand options to manage BART:

- **CHECK-CONFIG.** Check the setting of the parameters in the BART configuration file and the proper setup of the database servers for WAL archiving and taking of backups. See Section 3.4.1.
- **INIT.** Create the BART backup catalog directories, rebuild the `backupinfo` files, and set the `archive_command` in the Postgres server based on the setting of the `archive_command` in the `bart.cfg`. See Section 3.4.2 for details.
- **BACKUP.** Take a full backup or an incremental backup. See Section 3.4.3.
- **SHOW-SERVERS.** Display the database servers managed by BART. See Section 3.4.4.
- **SHOW-BACKUPS.** Display information for the backups taken by BART. See Section 3.4.5.
- **VERIFY-CHKSUM.** Verify the checksums on the full backups. See Section 3.4.6.
- **MANAGE.** Manage backups using the retention policy. Compress the archived WAL files. See Section 3.4.7.
- **RESTORE.** Restore a backup and generate an appropriate `recovery.conf` file including, if desired, a restore_command for restoring archived WAL files for point-in-time recovery. See Section 3.4.8.
- **DELETE.** Delete a backup. See Section 3.4.9.

For instructions and information about using the BART WAL scanner, see Section 3.5.

General BART Options

There are a number of general BART options that can be given immediately following specification of the `bart` program. Following the general BART options (if any are given), you can specify a BART subcommand.

- When invoking a subcommand, the subcommand name is case-insensitive (that is, the subcommand can be specified in uppercase, lowercase, or mixed case).
- Each subcommand has a number of its own applicable options that are specified following the subcommand. All options are available in both single-character and multi-character forms.
- The option keywords must generally be in lowercase except when specified differently in this section.
- When invoking BART, the current user must be the BART user account. Determine the operating system user account that will be used to run the BART command line program. This operating system user is referred to as the BART

user account. The chosen operating system user account must have the following capabilities:

The BART user account must:

- Own the BART backup catalog directory.
- Be able to run the `bart` program and the `bartscanner` program.
- Have a password-less SSH/SCP connection established between database servers managed by BART.

For example, enterprisedb or postgres can be selected as the BART user account when the managed database servers are Advanced Server or PostgreSQL respectively.

The general syntax for invoking BART is the following:

```
bart [ gen_option ]... [ subcmd ] [ subcmd_option ]...
```

The following are the general BART options denoted as *gen_option* in the above syntax diagram.

Options

`-h, --help`

Displays general syntax and information on BART usage.

`-v, --version`

Displays the BART version information.

`-d, --debug`

Displays debugging output while executing BART subcommands.

`-c, --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`.

Setting Path Environment Variable

If execution of BART subcommands fails with the following error message, then you need to set the `LD_LIBRARY_PATH` to include the `libpq` library directory:

```
./bart: symbol lookup error: ./bart: undefined symbol: PQping
```

Set the `LD_LIBRARY_PATH` environment variable for the BART user account to include the directory containing the `libpq` library. This directory is *POSTGRES_INSTALL_HOME*`/lib` as shown by the following example:

```
export LD_LIBRARY_PATH=/opt/PostgresPlus/9.5AS/lib/:$LD_LIBRARY_PATH
```

It is suggested that the PATH and the LD_LIBRARY_PATH environment variable settings be placed in the BART user account's profile. See Step 2 in Adding Configuration Parameter Values Section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for details.

The following examples illustrate the previously described ways for invoking BART. The BART user account is `bartuser` in these examples.

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

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
```

Use a BART configuration file other than *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.

3.4 BART Subcommands

This section describes the syntax and usage of the BART subcommands.

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 is an example of 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
```

Note: In the following sections, the help option is omitted from the syntax diagrams for the purpose of providing clarity for the subcommand options.

For clarity, the syntax diagrams also show only the single-character form of the option. The `Options` subsections list both the single-character and multi-character forms of the options.

3.4.1 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.

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

- When the `-s` option is omitted, the global section [BART], which contains parameters including `bart_host`, `backup_path`, and `pg_basebackup_path` is checked.

- When the `-s` option is specified, the parameters in the specified server section are checked. In addition, certain `postgresql.conf` parameters for the database server must be properly set and the database server must be activated for certain processes. These requirements include the following:

    - The `cluster_owner` parameter must be set to the user account owning the database cluster directory.

    - A password-less SSH/SCP connection must be set between the BART user and the user account specified by the `cluster_owner` parameter.

    - A database superuser must be specified by the BART `user` parameter.

    - The `pg_hba.conf` file must contain a replication entry for the database superuser specified by the BART `user` parameter.

    - The `archive_mode` parameter in the `postgresql.conf` file must be enabled.

    - The `archive_command` parameter in the `postgresql.auto.conf` or the `postgresql.conf` file must be set.

    - The `allow_incremental_backups` parameter in the BART configuration file must be enabled for database servers for which incremental backups are to be taken.

    - Archiving of WAL files to the BART backup catalog must be in process.

    - The WAL scanner program must be running.

- The `CHECK-CONFIG` subcommand displays an error message if the required configuration is not properly set.

Options

`-s, --server` *`server_name`*

`server_name` is the name of the database server to be checked for proper configuration. If the option is omitted, the settings of the global section of the BART configuration file are checked.

Example

The following example shows successful checking of the global section of the BART configuration file:

```
bash-4.1$ bart CHECK-CONFIG
INFO:  Verifying that pg_basebackup is executable
INFO:  success - pg_basebackup(/opt/PostgresPlus/9.5AS/bin/pg_basebackup)
returns version 9.500000
```

The following example shows successful checking of a database server:

```
bash-4.1$ bart CHECK-CONFIG -s mktg
INFO:  Checking server mktg
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:  /opt/backup/mktg
INFO:  Verifying database server configuration
INFO:  success
INFO:  Verifying that WAL archiving is working
INFO:  success
INFO:  Verifying that bart-scanner is configured and running
INFO:  success
```

3.4.2 INIT

The INIT subcommand may be invoked to perform the following actions:

- Create the BART backup catalog directory.
- Rebuild the BART backupinfo file.
- Set the archive_command in the PostgreSQL server based on the archive_command setting in the bart.cfg file.

```
bart INIT [ –s { server_name | all } ] [ -o ]
  [ -r [ -i { backup_id | backup_name | all } ] ]
bart INIT [ –s { server_name | all } ] [ -o ][-- no
configure]
```

Note: Do not invoke the INIT subcommand while the BART BACKUP subcommand is in progress. Backups affected by the backup process will be ignored by the INIT subcommand.

The following table summarizes the actions performed by INIT subcommand for the server specified by the –s option, or for all servers if -s all is specified or the –s option is omitted. See BART INIT examples section for examples of each option.

Options	Comments
INIT	Creates the BART backup catalog directory always (if it does not exist already). It also sets the archive_command in the PostgreSQL server (if it has not been set already) based on the archive_command setting in the bart.cfg. See BART INIT section for details.
-o	Overrides the archive_command setting in the PostgreSQL server if archive_mode is on and archive_command is already set.
INIT –r	Rebuilds the backupinfo file for all backups.
–r –i	Rebuilds the backupinfo file for the specified backup. The –i option can be used with the -r option only.

`--no configure`	Prevents the `archive_command` from being set in the PostgreSQL server.
`server_name`	Database server name to which the INIT actions are to be applied. If all is specified or if the option is omitted, the actions are applied to all servers.

BART INIT

As described in the above table, when you invoke the `BART INIT` subcommand, the BART backup catalog directory is created (if it does not exist already) and the `archive_command` is set in the PostgreSQL server. After the `archive_command` is set, you need to either restart the PostgreSQL server or reload the configuration in the PostgreSQL server based on the following scenarios. For examples, see BART INIT examples section.

Scenario 1 - If the `archive_mode` is set to off and `archive_command` is not set in the PostgreSQL server, the `archive_command` is set based on the `archive_command` setting in the `bart.cfg` and also sets the `archive_mode` to on. In this case, you need to restart the PostgreSQL server using `'pg_ctl restart'`

Scenario 2 – If the `archive_mode` is set to on and `archive_command` is not set in the PostgreSQL server, the `archive_command` is set based on the `archive_command` setting in the `bart.cfg`. In this case, you need to reload the sconfiguration in the PostgreSQL server using `pg_reload_conf()` or `'pg_ctl reload'`

Scenario 3 – If the `archive_mode` is set to off and `archive_command` is already set in the PostgreSQL server, the `archive_mode` is set to on. In this case, you need to restart the PostgreSQL server using `'pg_ctl restart'`

Scenario 4 - If the `archive_mode` is set to on and `archive_command` is already set in the PostgreSQL server, then the `archive_command` is not set unless `-o` option is specified.

BART INIT EXAMPLES

In this section you can view examples of `BART INIT` subcommand.

Example 1 – In this following example when you invoke `BART INIT` subcommand with `archive_mode` = off and `archive_command` not set, `archive_mode` is set to on and `archive_command` will be 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'
```

Example 2 - `BART INIT` with `archive_mode` = on and `archive_command` not 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'
```

Example 3 - `BART INIT` with `archive_mode` = on and `archive_command` already set

```
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.
```

Note: To override the existing `archive_command`, use the `-o` option.

Example 4 - `BART INIT` with `archive_mode` = off and `archive_command` already set

```
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'
```

Example 5 - `BART INIT -o`

The following example overrides an existing archive command setting by resetting the `archive_command` in the PostgreSQL server from the `archive_command = 'cp %p %a/%f'` parameter in the `bart.cfg` file.

The following is 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)
```

The `INIT` subcommand is invoked 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. (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'
```

Example 6 - `BART INIT-r`

When you invoke the `BART INIT` command with the `-r` option, it rebuilds the `backupinfo` file using the content of the backup directory for the server specified by the `-s` option, or for all servers if `-s all` is specified or the `-s` option is omitted. The BART `backupinfo` file (named `backupinfo)` 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 rebuilds 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 example 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 `backupinfo` file content is as follows:

```
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)
```

If the `backupinfo` file is missing, then the following error message appears when a BART subcommand is invoked:

```
-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 could be missing if the `BACKUP` subcommand did not complete successfully.

The following example rebuilds 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 example shows 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
```

Example 7 - `BART INIT –r -i`

```
bart INIT [ -r [ -i { backup_id | backup_name | all } ]
```

backup_id is a backup identifier. *backup_name* is the user-defined alphanumeric name for the backup.

```
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
```

Example 8 - `BART INIT –-no configure`

```
[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.
```

3.4.3 BACKUP

The BACKUP subcommand is used to create a full backup or an incremental backup.

```
bart BACKUP –s { server_name | all } [ -F { p | t } ]
  [ -z ] [ –c compression_level ]
  [ --parent { backup_id | backup_name } ]
  [ --backup-name backup_name ]
  [ --thread-count number_of_threads ]
  [ { --with-pg_basebackup | --no-pg_basebackup } ]
  [ --check ]
```

Note: While a BACKUP subcommand is in progress, no other processes are allowed to interfere with the affected backups. Any of the following subcommands issued while a backup is in progress will skip and ignore those affected backups.

– INIT, SHOW-BACKUPS, VERIFY-CHKSUM, MANAGE, and DELETE.

- By default, the target format is a tar file.
- The backup is saved in the directory formed by *backup_path*/*server_name*/*backup_id* where *backup_path* is the value assigned to the backup_path parameter in the BART configuration file, *server_name* is the lowercase name of the database server as listed in the configuration file, and *backup_id* is a backup identifier assigned by BART to the particular backup.
- MD5 checksums of the full backup and any user-defined tablespaces are saved as well for tar backups.
- When you use BART to take backup of PostgreSQL server versions 9.5 and prior:
  - Only one backup per server may be in progress at any given time.
  - If a backup is interrupted, you must manually run pg_stop_backup() to terminate the backup mode.

- When you use BART to take backup of PostgreSQL server versions 9.6 or greater:

  - Multiple backups can be taken simultaneously.
  - If a backup is interrupted, the backup mode is terminated automatically without the need to run pg_stop_backup() manually to terminate the backup.

Disk Space

Before performing the backup, BART checks to ensure there is enough disk space to completely store the backup in the BART backup catalog. If BART detects there is not

enough disk space, then no backup files are copied to the BART backup catalog and an error message is displayed as shown by the following:

```
edb@localhost bin]$ ./bart backup -s mktg -Ft

WARNING: xlog_method is empty, defaulting to global policy
ERROR: backup failed for server 'mktg'
free disk space is not enough to backup the server 'mktg'
space available 13.35 GB, approximately required 14.65 GB
```

Note: Although this capability to check and warn if there is not enough disk space is available before copying backup files is provided with the `BACKUP` subcommand, the `RESTORE` subcommand does not have this same capability. Thus, it is possible that the `RESTORE` subcommand may result in an error while copying files if there is not enough disk space available.

Wal_Keep_Segments Configuration Parameter

In the `postgresql.conf` file, ensure the `wal_keep_segments` configuration parameter is set to a sufficiently large value, otherwise you may encounter the following error during usage of the `BACKUP` subcommand:

```
ERROR: backup failed for server 'mktg'
command failed with exit code 1
pg_basebackup: could not get transaction log end position from server: ERROR:
requested WAL segment 00000001000000D50000006B has already been removed
```

- A low setting of the `wal_keep_segments` configuration parameter may result in the deletion of some WAL files before the BART `BACKUP` subcommand has had a chance to save them to the BART backup catalog.
- For information about the `wal_keep_segments` parameter, see the *PostgreSQL Core Documentation* available at:

  https://www.postgresql.org/docs/11/static/runtime-config-replication.html

Transaction Log Files

If in the BART configuration file, parameter setting `xlog_method=stream` applies to a given database server, streaming of the transaction log in parallel with creation of the backup is performed for that database server, otherwise the transaction log files are collected upon completion of the backup.

See *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for details about global setting of `xlog_method`. See database server section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for setting of `xlog_method` by database server.

Note: If the transaction log streaming method is used, the `-F p` option for a plain text backup format must be specified with the `BACKUP` subcommand.

Options

`-s, --server { server_name | all }`

server_name is the name of the database server to be backed up as specified in the BART configuration file. If `all` is specified, all servers are backed up.

Note: If `all` is specified, and a connection to a database server listed in the BART configuration file cannot be opened, the backup for that database server is skipped, but the backup operation continues for the other database servers. The following error message is displayed when a database server connection fails:

```
ERROR: backup failed for server 'mktg'
connection to the server failed: could not connect to server: Connection refused
    Is the server running on host "172.16.114.132" and accepting
    TCP/IP connections on port 5444?
```

`-F, --format { p | t }`

Specifies the backup file format. Use `p` for plain text or `t` for tar. If the option is omitted, the default is tar format.

Note: For taking incremental backups, the `-F p` option must be specified.

`-z, --gzip`

Specifies usage of gzip compression on the tar file output using the default compression level. The default compression level is typically 6. This option is applicable only for the tar format.

`-c, --compress-level compression_level`

Specifies the gzip compression level on the tar file output. *compression_level* is a digit from 1 through 9, with 9 being the best compression. This option is applicable only for the tar format.

`--parent { backup_id | backup_name }`

backup_id is the backup identifier of a parent backup. *backup_name* is the user-defined alphanumeric name of a parent backup. Specify this option to take an incremental backup. The parent is a backup taken prior to the incremental backup currently being invoked. The parent backup can be either a full backup or an incremental backup. The `-F p` option must be specified as well since an incremental backup can only be taken in plain text format.

Note: An incremental backup cannot be taken on a standby database server. See Section 2.1 for additional information on incremental backups.

`--backup-name` *`backup_name`*

User-defined, friendly name to be assigned to the backup. This is an alphanumeric string that may include the following variables to be substituted by the timestamp values when the backup is taken: 1) `%year` – 4-digit year, 2) `%month` – 2-digit month, 3) `%day` – 2-digit day, 4) `%hour` – 2-digit hour, 5) `%minute` – 2-digit minute, and 6) `%second` – 2-digit second. To include the percent sign (`%`) as a character in the backup name, specify `%%` in the alphanumeric string. Enclose the string in single quotes (`'`) or double quotes (`"`) if it contains space characters. Use of space characters, however, then requires enclosing the backup name in quotes when referenced with the `-i` option by other subcommands. The maximum permitted length of backup names is 49 characters.

This option overrides the `backup_name` parameter in the server section of the BART configuration file. If the `--backup-name` option 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`*

If the `--thread-count` option is specified, *`number_of_threads`* is the number of worker threads to run in parallel to copy blocks for a backup.

Note: If parallel backup is run with N number of worker threads, then it will initiate N+ 1 concurrent connections with the server.

If the `--thread-count` option is omitted, then the `thread_count` parameter in the BART configuration file applicable to this database server is used. If `thread_count` is not enabled for this database server, then the `thread_count` setting in the global section of the BART configuration file is used. If this is not set as well, the default number of threads is 1. See Configuring the BART host and Configuring the Database Server sections of EDB Postgres Backup and Recovery Installation and Upgrade Guide for information about the `thread_count` parameter.

`--with-pg_basebackup`

Specifies that `pg_basebackup` is to be used 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 (see Configuring the BART host and Configuring the Database Server sections of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide)*.

Note: 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`

Specifies that `pg_basebackup` is not to be used to take a full backup.

Note: 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.

`--check`

Before taking an incremental backup, verifies that the required MBM files are present in the BART backup catalog. A message is displayed informing that the incremental backup can be successfully taken or that it will fail. The `--parent` option must be specified when the `--check` option is used. An actual incremental backup is not taken when the `--check` option is specified.

Example

The following example creates a full backup in the default tar format with gzip compression. Note that checksums are generated for the full backup and user-defined tablespaces for the tar format backup.

```
-bash-4.2$ bart BACKUP -s mktg -z
DEBUG: Server: acctg, No. Backups 8
DEBUG: Server: hr, Now: 2016-10-27 10:41:07 EDT, RetentionWindow: 345600 (secs) ==> 96
hour(s)
DEBUG: Exec Command: /opt/PostgresPlus/9.5AS/bin/pg_basebackup --version
INFO:  creating backup for server 'mktg'
INFO:  backup identifier: '1477579267918'
DEBUG: internal backup Command to be execute:
'/opt/PostgresPlus/9.5AS/bin/pg_basebackup -D /opt/backup/mktg/1477579267918 -X fetch -
P  -Ft -z -d "host=192.168.2.24 port=5443 user=repuser" '
55006/55006 kB (100%), 3/3 tablespaces

INFO:  backup completed successfully
DEBUG: Exec Command: tar  -C /opt/backup/mktg/1477579267918 -xzf
/opt/backup/mktg/1477579267918/base.tar.gz backup_label
WARNING: log_timezone is not set in the server, using the local timezone information
DEBUG: calculate checksum for backup '/opt/backup/mktg/1477579267918'
DEBUG: calculating checksum of file '/opt/backup/mktg/1477579267918/17283.tar.gz'
INFO:  backup checksum: 4f69a5f2ed7092aede490de040e685fb of 17283.tar.gz
DEBUG: calculating checksum of file '/opt/backup/mktg/1477579267918/17284.tar.gz'
INFO:  backup checksum: 103e1e39003e0eb6acad11d4f791be45 of 17284.tar.gz
DEBUG: calculating checksum of file '/opt/backup/mktg/1477579267918/base.tar.gz'
INFO:  backup checksum: 6b5efb3e701ac30372db74e3ad8eac21 of base.tar.gz
WARNING: cannot get the tablespace(s) information for backup '1477579267918'
DEBUG: start time: 1477582868, stop time: 1477582870, duration: 2
DEBUG: Backup Info file created at '/opt/backup/mktg/1477579267918/backupinfo'
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1477579267918
BACKUP NAME: none
BACKUP PARENT: none
```

```
BACKUP LOCATION: /opt/backup/mktg/1477579267918
BACKUP SIZE: 5.45 MB
BACKUP FORMAT: tar.gz
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 3
 ChkSum                           File
 4f69a5f2ed7092aede490de040e685fb   17283.tar.gz
 103e1e39003e0eb6acad11d4f791be45   17284.tar.gz
 6b5efb3e701ac30372db74e3ad8eac21   base.tar.gz

TABLESPACE(s): 4294967295

START WAL LOCATION: 000000010000000200000051
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2016-10-27 10:41:08 EDT
STOP TIME: 2016-10-27 10:41:10 EDT
TOTAL DURATION: 2 sec(s)
```

The following example shows the directory containing the full backup:

```
-bash-4.2$ pwd
/opt/backup
-bash-4.2$ ls -l mktg
total 4
drwx------ 2 enterprisedb enterprisedb   79 Oct 27 10:41 1477579267918
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Oct 27 10:41 archived_wals
```

The following example shows the creation of a full backup while streaming the transaction log. Note that the `-F p` option must be specified with the `BACKUP` subcommand when streaming is used.

```
-bash-4.2$ bart BACKUP -s ACCTG -F p
DEBUG: Server: acctg, No. Backups 8
DEBUG: Server: hr, Now: 2016-10-27 10:46:36 EDT, RetentionWindow: 345600 (secs) ==> 96
hour(s)
DEBUG: Exec Command: /opt/PostgresPlus/9.5AS/bin/pg basebackup --version
INFO:  creating backup for server 'acctg'
INFO:  backup identifier: '1477579596637'
DEBUG: internal backup Command to be execute:
'/opt/PostgresPlus/9.5AS/bin/pg_basebackup -D /opt/backup/acctg/1477579596637/base -X
stream -P  -Fp -d "host=127.0.0.1 port=5444 user=enterprisedb"  '
40145/40145 kB (100%), 1/1 tablespace

INFO:  backup completed successfully
WARNING: log_timezone is not set in the server, using the local timezone information
DEBUG: start time: 1477583196, stop time: 1477583197, duration: 1
DEBUG: Backup Info file created at '/opt/backup/acctg/1477579596637/backupinfo'
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1477579596637
BACKUP NAME: acctg_2016-10-27T10:46:36
BACKUP PARENT: none
BACKUP LOCATION: /opt/backup/acctg/1477579596637
BACKUP SIZE: 54.50 MB
BACKUP FORMAT: plain
XLOG METHOD: stream
BACKUP CHECKSUM(s): 0
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000010000000EC
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2016-10-27 10:46:36 EDT
STOP TIME: 2016-10-27 10:46:37 EDT
TOTAL DURATION: 1 sec(s)
```

The following example shows the assignment of a user-defined backup name with the `--backup-name` option:

```
-bash-4.2$ bart BACKUP -s acctg --backup-name acctg_%year-%month-%day
INFO:  creating backup for server 'acctg'
INFO:  backup identifier: '1482700280852'
60944/60944 kB (100%), 1/1 tablespace

INFO:  backup completed successfully
WARNING: log_timezone is not set in the server, using the local timezone information
INFO:  backup checksum: e47107a0677dcc5acb8de40d66058e65 of base.tar
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1482700280852
BACKUP NAME: acctg_2016-12-25
BACKUP PARENT: none
BACKUP LOCATION: /opt/backup/acctg/1482700280852
BACKUP SIZE: 59.52 MB
BACKUP FORMAT: tar
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
 ChkSum                           File
 e47107a0677dcc5acb8de40d66058e65   base.tar

TABLESPACE(s): 0
START WAL LOCATION: 000000010000000100000045
STOP WAL LOCATION: 000000010000000100000045
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2016-12-25 16:11:21 EST
STOP TIME: 2016-12-25 16:11:21 EST
TOTAL DURATION: 0 sec(s)
```

The following example shows an incremental backup taken by specifying the `--parent` option. The `-F p` option must be specified as well for plain text format.

```
-bash-4.1$ bart BACKUP -s hr -F p --parent hr_full_1 --backup-name hr_incr_1
INFO:  creating incremental backup for server 'hr'
INFO:  checking mbm files /opt/backup/hr/archived_wals
INFO:  new backup identifier generated 1490819642608
INFO:  reading directory /opt/backup/hr/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: 1490819642608
BACKUP NAME: hr_incr_1
BACKUP PARENT: 1490819418664
BACKUP LOCATION: /opt/backup/hr/1490819642608
BACKUP SIZE: 16.53 MB
BACKUP FORMAT: plain
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 0
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000007
STOP WAL LOCATION: 000000010000000000000007
BACKUP METHOD: pg_start_backup
BACKUP FROM: master
START TIME: 2017-03-29 16:34:04 EDT
STOP TIME: 2017-03-29 16:34:05 EDT
TOTAL DURATION: 1 sec(s)
```

3.4.4 SHOW-SERVERS

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

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

The default action is to show all servers.

Options

`-s, --server { server_name | all }`

`server_name` is the name of the database server whose BART configuration information is to be displayed. If `all` is specified or if the option is omitted, information for all database servers is displayed.

Example

The following example shows all database servers managed by BART:

```
-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"
```

3.4.5 SHOW-BACKUPS

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

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

If all options are omitted, the default action is to show all backups of all servers with the exception as described by the following note:

Note: If `SHOW-BACKUPS` is invoked while the BART `BACKUP` subcommand is in progress, backups affected by the backup process are shown in progress status in the displayed backup information.

Options

`-s, --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, --backupid { backup_id | backup_name | all }`

`backup_id` is a backup identifier. *`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, --toggle`

Display more comprehensive backup information in list format. If omitted, the default is a briefer, tabular format.

Example

The following example 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 example shows more 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 : 0000000100000001000000EC
CREATION TIME  : 2016-10-27 10:46:37 EDT
LAST WAL FILE  : 0000000100000001000000F0
CREATION TIME  : 2016-10-27 11:22:01 EDT
```

The following example 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   none                acctg_2016-10-27    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 example 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 : 0000000100000002000000D9
CREATION TIME  : 2016-10-27 10:58:13 EDT
LAST WAL FILE  : 0000000100000002000000D9
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 : 0000000100000002000000D8
CREATION TIME  : 2016-10-27 10:55:12 EDT
LAST WAL FILE  : 0000000100000002000000D8
CREATION TIME  : 2016-10-27 10:55:12 EDT
```

3.4.6 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.

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

The checksum is verified by comparing the current checksum of the backup against the checksum when the backup was taken. The `VERIFY-CHKSUM` subcommand is only used for tar format backups. It is not applicable to plain format backups.

Note: If `VERIFY-CHKSUM` is invoked while the BART `BACKUP` subcommand is in progress, backups affected by the backup process will be skipped and ignored by the `VERIFY-CHKSUM` subcommand.

Options

`-s, --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 are verified for all database servers.

`-i, --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 example 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
```

3.4.7 MANAGE

The MANAGE subcommand evaluates backups, marks their status, and deletes obsolete backups based on the retention_policy parameter in the BART configuration file (See Section 3.2 for information about retention policy management). The MANAGE subcommand also invokes compression on the archived WAL files based on the wal_compression parameter in the BART configuration file (See Configuring the BART host and Configuring the Database Server sections of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about setting this parameter).

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

Note: Do not invoke the MANAGE subcommand while the BART BACKUP subcommand is in progress. Backups affected by the backup process will be skipped and ignored by the MANAGE subcommand.

The following summarizes the actions performed under certain conditions and options when the MANAGE subcommand is invoked.

- When the MANAGE subcommand is invoked with no options or with only the -s option to specify all or a particular database server, the following actions are performed:
  - For the server specified by the -s option, or for all servers if -s all is specified or the -s option is omitted, active backups are marked as obsolete in accordance with the retention policy.
  - All backups that were marked obsolete or keep prior to invoking the MANAGE subcommand remain marked with the same prior status.
  - If WAL compression is enabled for the database server, then any uncompressed, archived WAL files in the BART backup catalog of the database server are compressed with gzip.
- When the MANAGE subcommand is invoked with any other option besides the -s option, the following actions are performed:
  - For the server specified by the -s option, or for all servers if -s all is specified or the -s option is omitted, the action performed is determined by the other specified options (that is, -l to list obsolete backups, -d to delete obsolete backups, -c to keep or to return backups to active status, or -n to perform a dry run of any action).
  - No marking of active backups to obsolete status is performed regardless of the retention policy.
  - All backups that were marked obsolete or keep prior to invoking the MANAGE subcommand remain marked with the same prior status unless the -c option

(without the `-n` option) is specified to change the backup status of the particular backup or all backups referenced with the `-i` option.

- No compression is applied to any uncompressed, archived WAL file in the BART backup catalog regardless of whether or not WAL compression is enabled.

The following are additional considerations when using WAL compression:

- Compression of archived WAL files is not permitted for database servers on which incremental backups are to be taken. In other words, parameters `wal_compression` and `allow_incremental_backups` both cannot have an enabled effect on the same database server.

- The gzip compression program must be installed on the BART host and be accessible in the `PATH` of the BART user account.

- When the `RESTORE` subcommand is invoked, if the `-c` option is specified or if the enabled setting of the `copy_wals_during_restore` BART configuration parameter is in effect for the database server, then the following actions occur:

  If compressed, archived WAL files are stored in the BART backup catalog and the location to which the WAL files are to be restored is on a remote host relative to the BART host, then the archived WAL files are transmitted across the network to the remote host in compressed format, but only if the gzip compression program is accessible in the `PATH` of the remote user account that is used to log into the remote host when performing the `RESTORE` operation. This remote user is specified with either the `remote_host` parameter in the BART configuration file (See the configuration information in the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information) or the `RESTORE -r` option (see Section 3.4.8). Transmission of compressed WAL files results in less network traffic. After the compressed WAL files are transmitted across the network, the `RESTORE` subcommand uncompresses the files for the point-in-time recovery operation.

- If the gzip program is not accessible on the remote host in the manner described in the previous bullet point, then the compressed, archived WAL files are first uncompressed while on the BART host, then transmitted across the network to the remote host in uncompressed format.

- When the `RESTORE` subcommand is invoked without the `-c` option and the disabled setting of the `copy_wals_during_restore` BART configuration parameter is in effect for the database server, then any compressed, archived WAL files needed for the `RESTORE` operation are uncompressed in the BART backup catalog. The uncompressed WAL files can then be streamed to the

remote host by the `restore_command` in the `recovery.conf` file when the database server archive recovery begins.

Options

`-s, --server { server_name | all }`

`server_name` is the name of the database server to which the actions are to be applied. If `all` is specified or if the `-s` option is omitted, the actions are applied to all database servers.

`-l, --list-obsolete`

List the backups marked as obsolete.

`-d, --delete-obsolete`

Delete 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, --change-status { keep | nokeep }`

Change the status of a backup to `keep` to retain it indefinitely. Specify `nokeep` to change the status of any backup, regardless of its current marked status, back to active status. The backup can then be re-evaluated and possibly be marked to obsolete according to the retention policy by subsequent usage of the `MANAGE` subcommand.

Note: The `-i` option must be included when using the `-c` option.

`-i, --backupid { backup_id | backup_name | all }`

`backup_id` is a backup identifier. *`backup_name`* is the user-defined alphanumeric name for the backup. If `all` is specified, the actions are applied to all backups.

Note: The `-i` option must only be used with the `-c` option.

`-n, --dry-run`

Displays the results as if the operation was performed, however, no changes are actually made. In other words, a test run is performed so that you can see the results prior to actually implementing the actions. Thus, `-n` specified with the `-d` option displays which backups would be deleted, but does not actually delete the backups. Specifying the `-n` option with the `-c` option displays the keep or nokeep action, but does not actually change the backup from its current status. Specifying `-n` alone with no other options, or with only the `-s` option, 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 example 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 example 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 example 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 example 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 example 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 example 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 example 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 example shows the archived WAL files in compressed format:

```
-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
```

3.4.8 RESTORE

The `RESTORE` subcommand restores the backup and its archived WAL files for the designated database server to the specified directory location. If the appropriate `RESTORE` options are specified, a `recovery.conf` file is generated with the recovery configuration parameters.

```
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 ]
```

Review the information about using a continuous archive backup for recovery in the *PostgreSQL Core Documentation* available at:

https://www.postgresql.org/docs/11/static/continuous-archiving.html

This reference material provides detailed information about the underlying point-in-time recovery process and the meaning and usage of the restore options that are generated into the `recovery.conf` file by BART.

Note: See Section 2.1.5.2 for special requirements when restoring an incremental backup to a remote database server.

Note: Check to ensure that the host where the backup is to be restored contains enough disk space for the backup and its archived WAL files. The `RESTORE` subcommand does not have the capability to detect if there is sufficient disk space available before restoring the backup files. Thus, it is possible that the `RESTORE` subcommand may result in an error while copying files if there is not enough disk space available.

The steps for performing a restore operation are the following:

Step 1: Stop the Postgres database server on which you will be performing the restore operation.

Step 2: Inspect the `pg_xlog` subdirectory of the data directory and ensure to save any WAL files that have not yet been archived to the BART backup catalog (`backup_path/server_name/archived_wals`).

If there are some files that have not been archived, save them to a temporary location. You will later need to copy these files to the restored `pg_xlog` subdirectory after completing the `RESTORE` subcommand and before restarting the database server.

Step 3: Decide if you want to restore to the current data directory or to a new directory.

- If you are restoring to the current data directory, delete all files and subdirectories under the data directory. For example, for an initial Postgres database server installation, this directory is *`POSTGRES_INSTALL_HOME`*`/data`.
- If you are restoring to a new directory, create the directory on which you want to restore the backed up database cluster. Make sure the data directory can be written to by the BART user account or by the user account specified by the `remote_host` configuration parameter, or by the `--remote-host` option of the `RESTORE` subcommand if these are to be used.

Step 4: Perform the same process for tablespaces as described in Step 3. The `tablespace_path` parameter in the BART configuration file must contain the tablespace directory paths to which the tablespace data files are to be restored. See Configuring the BART host section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for more information.

Step 5: Identify the timeline ID you wish to use to perform the restore operation.

The available timeline IDs can be identified by the first non-zero digit of the WAL file names reading from left to right.

In the following example, 1 is the only timeline ID in all of the available WAL files.

```
-bash-4.1$ pwd
/opt/backup/acctg/archived wals
-bash-4.1$ ls -l
total 49160
-rw------- 1 ... ... 16777216 Mar 29 13:47 000000010000000000000003
-rw------- 1 ... ...      302 Mar 29 13:47 000000010000000000000003.00000028.backup
-rw------- 1 ... ... 16777216 Mar 29 13:48 000000010000000000000004
-rw------- 1 ... ...      302 Mar 29 13:48 000000010000000000000004.00000028.backup
-rw------- 1 ... ... 16777216 Mar 29 14:07 000000010000000000000005
```

Step 6: Identify the backup to use for the restore operation and obtain the backup ID or backup name. If you wish to use the latest backup, you can omit the `-i` option and the `RESTORE` subcommand uses that backup by default.

The backups can be listed with the `SHOW-BACKUPS` subcommand as in the following example:

```
-bash-4.1$ bart SHOW-BACKUPS -s acctg
 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
```

Step 7: Run the BART `RESTORE` subcommand.

If any of `-t` *`timeline_id`*, `-x` *`target_xid`*, or `-g` *`target_timestamp`* options are given, then a `recovery.conf` file is generated with the recovery configuration parameters corresponding to the specified options. That is, point-in-time recovery is performed upon restarting the database server.

If you do not specifythe `-t` *`timeline_id`*, `-x` *`target_xid`*, and `-g` *`target_timestamp`* options, then a minimal `recovery.conf` file is generated. Archive recovery will proceed only until consistency is reached, with no restoration of files from the archive.

Use the `--debug` option to display the underlying commands used by BART.

Note: If invalid values are specified for the options, or if invalid option combinations are specified (for example, if both the `-x` *`target_xid`* and the `-g` *`target_timestamp`* options are given), no error message is generated by BART. The invalid options are accepted and passed to the `recovery.conf` file, which will then be processed by the database server when it is restarted.

Ensure that valid options are specified when using the RESTORE subcommand.

The following example uses the default, latest backup by omitting the `-i` option:

```
-bash-4.1$ bart --debug RESTORE -s acctg -p /opt/restore
DEBUG: Server: Global, No of Retained Backups 6
DEBUG: Exec Command: set -o pipefail;  test -d /opt/restore && echo "exists"
DEBUG: Exec Command: set -o pipefail;  touch /opt/restore/tmp-1490809695281
&& echo "exists"
DEBUG: Exec Command: set -o pipefail;  rm -f /opt/restore/tmp-1490809695281
DEBUG: Exec Command: set -o pipefail;  ls -A /opt/restore
INFO:  restoring backup '1490809695281' of server 'acctg'
DEBUG: restoring backup to /opt/restore
DEBUG: restore command: cat /opt/backup/acctg/1490809695281/base.tar.gz |
tar -C /opt/restore -xzf -
DEBUG: Exec Command: set -o pipefail; cat
/opt/backup/acctg/1490809695281/base.tar.gz |   tar -C /opt/restore -xzf -
INFO:  base backup restored
DEBUG: Exec Command: set -o pipefail; echo "
archive_mode = off" |   cat >> /opt/restore/postgresql.conf
INFO:  archiving is disabled
DEBUG: Exec Command: set -o pipefail;   chmod 0700 /opt/restore
INFO:  permissions set on $PGDATA
INFO:  restore completed successfully
```

Note: If the `-c` option is specified or if the enabled setting of the `copy_wals_during_restore` BART configuration parameter is in effect for this database server, then the following actions occur:

- In addition to restoring the database cluster to the directory specified by the `-p` *`restore_path`* option, the archived WAL files of the backup are copied from the BART backup catalog to the subdirectory *`restore_path`*`/archived_wals`.
- If a `recovery.conf` file is generated, the command string set in the `restore_command` parameter retrieves the WAL files from this `archived_wals` subdirectory relative to the *`restore_path`* parent directory as: `restore_command = 'cp archived_wals/%f %p'`

Step 8: Copy any saved WAL files from Step 2 to the *`restore_path`*`/pg_xlog` subdirectory.

Step 9: Inspect the restored directories and data files of the restored database cluster in directory *`restore_path`*.

All files and directories must be owned by the user account that you intend to use to start the database server. This user account is typically the Postgres user account (`enterprisedb` or `postgres`), but it may be some other user account of your choice. Recursively change the user and group ownership of the *`restore_path`* directory, its files, and its subdirectories if necessary.

There must only be directory access privileges for the user account that will start the database server. No other groups or users can have access to the directory.

Step 10: If you are performing a point-in-time recovery, inspect the `recovery.conf` file located in the *`restore_path`* directory to verify it contains the appropriate parameter settings to recover to the indicated point. Otherwise, the `recovery.conf` file should be configured to recover only until the cluster reaches consistency. In either case, the settings may be modified as desired.

Step 11: WAL archiving is disabled at this point.

The BART `RESTORE` subcommand adds an `archive_mode = off` parameter at the end of the `postgresql.conf` file.

The following shows the end of the file where this parameter is added:

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

- If you want to restart the database server with WAL archiving activated, be sure to delete this additional parameter.
- The original `archive_mode` parameter still resides in the `postgresql.conf` file in its initial location with its last setting.

Step 12: Start the database server to initiate recovery. After completion, check the database server log file to ensure the recovery was successful.

Note: If the backup is restored to a different database cluster directory than where the original database cluster resided, then certain operations dependent upon the database cluster location may fail if their supporting service scripts are not updated to reflect the new directory location where the backup has been restored.

For information about the usage and modification of service scripts, see the *EDB Postgres Advanced Server Installation Guide* available at:

https://www.enterprisedb.com/resources/product-documentation

The following table lists the service scripts for RHEL 6/CentOS 6.

Table 3-2 Scripts with Database Cluster Location for RHEL 6/CentOS 6

File Name	Location	Description
`postgres-reg.ini`	`/etc`	Product information for upgrades
`ppas-9.5`	`/etc/init.d`	Service script for Advanced Server 9.5 from interactive installer
`edb-as-9.6`	`/etc/init.d`	Service script for Advanced Server 9.6 from interactive installer
`edb-as-10`	`/etc/init.d`	Service script for Advanced Server 10 from interactive installer
`edb-as-11`	`/etc/init.d`	Service script for Advanced Server 11 from interactive installer
`ppas-9.5`	`/etc/sysconfig/ppas`	Configuration script for Advanced Server 9.5 from RPM package
`edb-as-9.6.sysconfig`	`/etc/sysconfig/edb/as9.6`	Configuration script for Advanced Server 9.6 from RPM package

edb-as-10.sysconfig	/etc/sysconfig/edb/as10	Configuration script for Advanced Server 10 from RPM package
edb-as-11.sysconfig	/etc/sysconfig/edb/as11	Configuration script for Advanced Server 11 from RPM package
postgresql-9.5	/etc/init.d	Service script for PostgreSQL 9.5 from interactive installer
postgresql-9.6	/etc/init.d	Service script for PostgreSQL 9.6 from interactive installer
postgresql-10	/etc/init.d	Service script for PostgreSQL 10 from interactive installer
postgresql-11	/etc/init.d	Service script for PostgreSQL 11 from interactive installer

Note: Before modifying the service unit files for Advanced Server in RHEL 7/CentOS 7, review the instructions in the section titled: *Modifying the Data Directory Location on CentOS or RedHat 7.x* in the *EDB Postgres Advanced Server Installation Guide* for your release.

The following table lists the service unit files and scripts for RHEL 7/CentOS 7.

Table 3-3 Unit Files and Scripts with Database Cluster Location for RHEL 7/CentOS 7

File Name	Location	Description
postgres-reg.ini	/etc	Product information for upgrades
ppas-9.5.service ppas-9.5.sh	/usr/lib/systemd/system	Service unit file and script for Advanced Server 9.5 from interactive installer
edb-as-9.6.service edb-as-9.6.sh	/usr/lib/systemd/system	Service unit file and script for Advanced Server 9.6 from interactive installer
edb-as-10.service edb-as-10.sh	/usr/lib/systemd/system	Service unit file and script for Advanced Server 10 from interactive installer
edb-as-11	/usr/lib/systemd/system	Service unit file and script for Advanced Server 11 from interactive installer
ppas-9.5.service ppas-9.5.sh	/usr/lib/systemd/system	Service unit file and script for Advanced Server 9.5 from RPM package
edb-as-9.6.service	/usr/lib/systemd/system	Service unit file for Advanced Server 9.6 from RPM package

File Name	Location	Description
edb-as-10.service	/usr/lib/systemd/system	Service unit file for Advanced Server 10 from RPM package
edb-as-11	/usr/lib/systemd/system	Service unit file for Advanced Server 11 from RPM package
postgresql-9.5.service	/usr/lib/systemd/system	Service unit file for PostgreSQL 9.5 from interactive installer
postgresql-9.6.service	/usr/lib/systemd/system	Service unit file for PostgreSQL 9.6 from interactive installer
postgresql-10.service	/usr/lib/systemd/system	Service unit file for PostgreSQL 10 from interactive installer
postgresql-11	/usr/lib/systemd/system	Service unit file for PostgreSQL 11 from interactive installer

Options

`-s, --server` *`server_name`*

`server_name` is the name of the database server to be restored.

`-p, --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, --backupid { ` *`backup_id`* `|` *`backup_name`* `}`

`backup_id` is the backup identifier of the backup to be used for the restoration. *`backup_name`* is the user-defined alphanumeric name for the backup. If the option is omitted, the default is to use the latest backup.

`-r, --remote-host` *`remote_user`*`@`*`remote_host_address`*

`remote_user` is the user account on the remote database server host that accepts a password-less 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.

Note: If the BART user account is not the same as the operating system account that owns the *`restore_path`* directory given with the `-p` option, the `remote_host` BART configuration parameter or the `RESTORE` subcommand `-r` option must be used to specify the *`restore_path`* directory owner even when restoring to a directory on the same host

as the BART host. See Configuring the BART host section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about the `remote_host` parameter.

`-w, --workers` *`number_of_workers`*

`number_of_workers` is the specification of the number of worker processes to run in parallel to stream the modified blocks of an incremental backup to the restore location. For example, if 4 worker processes are specified, 4 receiver processes on the restore host and 4 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. If the `-w` option is omitted, the default is `1` worker process.

`-t, --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, --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. Only one of the `-x` *`target_xid`* or the `-g` *`target_timestamp`* option should be included if point-in-time recovery is desired.

`-g, --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. Only one of the `-x` *`target_xid`* or the `-g` *`target_timestamp`* option should be included if point-in-time recovery is desired.

`-c, --copy-wals`

If specified, the archived WAL files are copied from the BART backup catalog to directory *`restore_path`*`/archived_wals`. If BART generates the `recovery.conf` file for point-in-time recovery, 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, the default action is that the `restore_command` in the `recovery.conf` file is generated to retrieve the archived WAL files directly from the BART backup catalog. See the *EDB Postgres*

Backup and Recovery Installation and Upgrade Guide for information about the `copy_wals_during_restore` parameter.

Example

The following example restores database server `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:  creating recovery.conf file
INFO:  archiving is disabled
INFO:  tablespace(s) restored
```

The following is the content of the generated `recovery.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 displays 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 recovery.conf
```

Example

The following example performs the `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:  creating recovery.conf file
INFO:  archiving is disabled
INFO:  permissions set on $PGDATA
INFO:  restore completed successfully
```

The following is the content of the generated `recovery.conf` file:

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

The following displays 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 recovery.conf
```

3.4.9 DELETE

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

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

Note that a specific database server must be specified.

Note: Do not invoke the `DELETE` subcommand while the BART `BACKUP` subcommand is in progress. Backups affected by the backup process will be skipped and ignored by the `DELETE` subcommand.

For database servers under a retention policy, there are conditions where certain backups may not be deleted. See Section 3.2.4.1 for information regarding permitted backup deletions.

Options

`-s, --server server_name`

server_name is the name of the database server whose backups are to be deleted.

`-i, --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. If `all` is specified, all of the backups and their archived WAL files for the specified database server are deleted.

`-n, --dry-run`

Displays the results as if the deletions were done, however, no physical removal of the files are actually performed. In other words, a test run is performed so that you can see the potential results prior to actually initiating the action.

Example

The following example deletes a backup from the specified database server.

```
$ bart DELETE -s acctg -i acctg_2015-04-15T16:00:24
INFO:  deleting backup 'acctg 2015-04-15T16:00:24' of server 'acctg'
INFO:  deleting backup '1429128024311'
INFO:  4 WAL file(s) will be removed
INFO:  1 Unused WAL file(s) will be removed
INFO:  deleting WAL file '000000010000000010000004E'
INFO:  deleting WAL file '000000010000000010000004D'
INFO:  deleting WAL file '000000010000000010000004B'
INFO:  deleting WAL file '000000010000000010000004A'
INFO:  deleting (unused) WAL file '000000010000000100000042.00000028'
INFO:  backup(s) deleted
```

After the deletion, the BART backup catalog for the database server no longer contains the corresponding directory for the deleted backup ID. The `archived_wals` subdirectory no longer contains the WAL files of the backup.

```
$ pwd
/opt/backup/acctg
$ ls -l
total 8
drwx------ 2 enterprisedb enterprisedb 4096 Apr 20 10:09 1429219568720
drwx------ 2 enterprisedb enterprisedb 4096 Apr 20 10:21 archived_wals
$ ls -l archived_wals
total 49164
-rw------- 1 enterprisedb enterprisedb      305 Apr 15 16:00
000000010000000010000004A.00000028.backup
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 16 17:26 000000010000000010000004F
-rw------- 1 enterprisedb enterprisedb      305 Apr 16 17:26
000000010000000010000004F.00000028.backup
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 17 15:09 000000010000000100000050
-rw------- 1 enterprisedb enterprisedb 16777216 Apr 17 15:14 000000010000000100000051
-rw------- 1 enterprisedb enterprisedb      305 Apr 17 15:14
000000010000000100000052.00000028.backup
```

The following example deletes multiple backups from the database server.

```
$ bart DELETE -s acctg -i 1428355371389,1428422324880,acctg_2015-04-17T15:14:33
INFO:  deleting backup '1428355371389' of server 'acctg'
INFO:  deleting backup '1428355371389'
INFO:  1 WAL file(s) will be removed
INFO:  deleting WAL file '0000000100000000000000AA'
INFO:  backup(s) deleted
INFO:  deleting backup '1428422324880' of server 'acctg'
INFO:  deleting backup '1428422324880'
INFO:  2 WAL file(s) will be removed
INFO:  1 Unused WAL file(s) will be removed
INFO:  deleting WAL file '0000000100000000000000E1'
INFO:  deleting WAL file '0000000100000000000000E0'
INFO:  deleting (unused) WAL file '0000000100000000000000AA.00000028'
INFO:  backup(s) deleted
INFO:  deleting backup 'acctg_2015-04-17T15:14:33' of server 'acctg'
INFO:  deleting backup '1429298073247'
INFO:  1 WAL file(s) will be removed
INFO:  deleting WAL file '000000010000000100000052'
INFO:  backup(s) deleted
```

The following example also deletes multiple backups, but since there are space characters in the comma-separated list, the entire list must be enclosed within single quotes.

```
$ bart DELETE -s acctg -i '1428502049836, 1428589759899, 1428684537299'
INFO:  deleting backup '1428502049836' of server 'acctg'
INFO:  deleting backup '1428502049836'
INFO:  6 WAL file(s) will be removed
INFO:  deleting WAL file '000000010000000100000003'
INFO:  deleting WAL file '000000010000000100000002'
INFO:  deleting WAL file '000000010000000100000001'
INFO:  deleting WAL file '000000010000000100000000'
INFO:  deleting WAL file '0000000100000000000000E3'
INFO:  deleting WAL file '0000000100000000000000E2'
INFO:  backup(s) deleted
INFO:  deleting backup '1428589759899' of server 'acctg'
INFO:  deleting backup '1428589759899'
INFO:  6 WAL file(s) will be removed
INFO:  1 Unused WAL file(s) will be removed
INFO:  deleting WAL file '000000010000000100000009'
INFO:  deleting WAL file '000000010000000100000008'
INFO:  deleting WAL file '000000010000000100000007'
INFO:  deleting WAL file '000000010000000100000006'
INFO:  deleting WAL file '000000010000000100000005'
INFO:  deleting WAL file '000000010000000100000004'
INFO:  deleting (unused) WAL file '0000000100000000000000E2.00000028'
INFO:  backup(s) deleted
INFO:  deleting backup '1428684537299' of server 'acctg'
INFO:  deleting backup '1428684537299'
INFO:  17 WAL file(s) will be removed
INFO:  2 Unused WAL file(s) will be removed
INFO:  deleting WAL file '000000010000000100000024'
INFO:  deleting WAL file '000000010000000100000023'
INFO:  deleting WAL file '000000010000000100000022'
INFO:  deleting WAL file '000000010000000100000021'
INFO:  deleting WAL file '000000010000000100000020'
INFO:  deleting WAL file '00000001000000010000001F'
INFO:  deleting WAL file '00000001000000010000001E'
INFO:  deleting WAL file '00000001000000010000001D'
INFO:  deleting WAL file '00000001000000010000001C'
INFO:  deleting WAL file '00000001000000010000001B'
INFO:  deleting WAL file '00000001000000010000001A'
INFO:  deleting WAL file '000000010000000100000019'
INFO:  deleting WAL file '000000010000000100000018'
INFO:  deleting WAL file '000000010000000100000017'
INFO:  deleting WAL file '000000010000000100000016'
INFO:  deleting WAL file '000000010000000100000015'
INFO:  deleting WAL file '000000010000000100000014'
INFO:  deleting (unused) WAL file '000000010000000100000004.00000028'
INFO:  deleting (unused) WAL file '00000001000000010000000A.00000028'
INFO:  backup(s) deleted
```

3.5 Running the BART WAL Scanner

The BART WAL scanner is used by invoking the `bart-scanner` program located in the *`BART_HOME`*`/bin` directory.

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

For clarity, the syntax diagram shows only the single-character form of the option when the multi-character form is supported as well. The **Options** subsection lists both the single-character and multi-character forms of the options.

The WAL scanner processes each WAL file to find and record modified blocks in a corresponding modified block map (MBM) file.

The default approach is that the WAL scanner gets notified whenever a new WAL file arrives in the `archived_wals` directory of the BART backup catalog. It then scans the WAL file and produces the MBM file. This approach does not work in some cases, for example when the WAL files are shipped to the BART backup catalog using the `rsync` utility and also in case of some specific platforms.

This results in the WAL files being copied to the `archived_wals` directory, but the WAL scanner is not aware of it so it does not scan them and produce the MBM files. This results in the failure of an incremental backup.

This can be avoided by using the timer-based WAL scanning approach, which is done by using the `scan_interval` parameter in the BART configuration file (see Configuring the BART host and Configuring the Database Server sections of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for more information). The value for `scan_interval` is the number of seconds after which the WAL scanner will scan the new WAL files.

See Upading the Configuration file Section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for additional information on WAL scanning.

When the `bart-scanner` program is invoked, it forks a separate process for each database server enabled with the `allow_incremental_backups` parameter. See Configuring the Database Server section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about 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 omitted, then 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. See Configuring the Database Server section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about the `logfile` parameter in the BART configuration file.

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.

When invoking the WAL scanner, the current user must be the BART user account as described in Establishing the BART user accout section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide*.

Note: The BART user account's `LD_LIBRARY_PATH` environment variable may need to be set to include the directory containing the `libpq` library if invocation of the WAL scanner program fails. See Section 3.3 for information about setting `LD_LIBRARY_PATH`.

Options

`-h, --help`

Displays general syntax and information on WAL scanner usage.

`-v, --version`

Displays the WAL scanner version information.

`-d, --debug`

Displays debugging output while executing the WAL scanner with any of its options.

`-c, --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`

Run the WAL scanner as a background process.

`-p, --print` *`mbm_file`*

Full directory path to an MBM file whose content is to be printed. The archive path directory *`backup_path`*`/`*`server_name`*`/archived_wals` contains the MBM files. **Note:** This option is to be used for assisting the EnterpriseDB support team for debugging problems that may have been encountered.

`wal_file`

Full directory path to a WAL file to be scanned. The archive path directory *`backup_path`*`/`*`server_name`*`/archived_wals` contains the WAL files. Use it if a WAL file in the archive path is missing its MBM file.

`RELOAD`

Reload the BART configuration file. The keyword `RELOAD` is case-insensitive. This 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`

Stop the WAL scanner. The keyword `STOP` is case-insensitive.

Example

The following example 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 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 observable 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 000000010000000F000002800000000F1000000.mbm
-rw-rw-r-- 1 ... ...      161 Dec 20 09:18 000000010000000F100002800000000F2000000.mbm
```

To stop the interactively running WAL scanner, either enter `cntrl-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: 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
INFO:  bart-scanner stopped
-bash-4.2$
```

The following example shows how to invoke 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/000000010000000000000FF
-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 in the following example:

```
-bash-4.2$ bart-scanner -p
/opt/backup/acctg/archived wals/0000000100000000FF000028000000100000000.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
```

4 Using Tablespaces

If the database cluster contains user-defined tablespaces (that is, tablespaces created with the `CREATE TABLESPACE` command), note the following:

- You can take full backups with the `BACKUP` subcommand in either tar (`-F t`) or plain text (`-F p`) backup file format.
- You must take incremental backups in the plain text (`-F p`) backup file format.
- You can take full backups using the transaction log streaming method (xlog_method = stream in the BART configuration file) `--with-pg_basebackup` and the `BACKUP` subcommand in either tar (`-F t`) or plain text (`-F p`) backup file format.

If the particular database cluster you plan to back up contains tablespaces created by the `CREATE TABLESPACE` command, be sure to set the `tablespace_path` parameter in the BART configuration file before you perform a BART `RESTORE` operation.

The `tablespace_path` parameter specifies the directory paths to which you want the tablespaces to be restored.

The `tablespace_path` parameter takes the following format:

```
OID_1=tablespace_path_1;OID_2=tablespace_path_2 ...
```

OID_1, *OID_2*, … are the Object Identifiers of the tablespaces. You can find the OIDs of the tablespaces and their corresponding soft links to the directories by listing the contents of the *POSTGRES_INSTALL_HOME*`/data/pg_tblspc` subdirectory as shown in the following example:

```
[root@localhost pg_tblspc]# pwd
/opt/PostgresPlus/9.5AS/data/pg tblspc
[root@localhost pg_tblspc]# ls -l
total 0
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Aug 22 16:38 16644 -> /mnt/tablespace_1
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Aug 22 16:38 16645 -> /mnt/tablespace_2
```

The OIDs are `16644` and `16645` to directories `/mnt/tablespace_1` and `/mnt/tablespace_2`, respectively. If you later wish to restore the tablespaces to the same locations as indicated in the preceding example, the BART configuration file must contain the following entry:

```
[ACCTG]
host = 127.0.0.1
port = 5444
user = enterprisedb
cluster_owner = enterprisedb
tablespace_path = 16644=/mnt/tablespace_1;16645=/mnt/tablespace_2
description = "Accounting"
```

If you wish to restore the tablespaces to different locations, specify the new directory locations in the `tablespace_path` parameter.

In either case, the directories specified in the `tablespace_path` parameter must exist and be empty at the time you perform the `BART RESTORE` operation.

If the database server is running on a remote host (in other words you are also using the `remote_host` configuration parameter or will specify the `--remote-host` option with the `RESTORE` subcommand), the specified tablespace directories must exist on the specified remote host.

The directories must be owned by the user account with which you intend to start the database server (typically the Postgres user account) with no access by other users or groups as is required for the directory path to which the main full backup is to be restored.

Example

The following is an example of backing up and then restoring a database cluster on a remote host that includes tablespaces.

Note: This example emphasizes the steps that are affected by tablespace usage. See Section 3 of the EDB Postgres Backup and Recovery Guide for the complete process required for backing up and restoring a database cluster.

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 example 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 (described in this section) are performed to ensure BART manages the remote database server, a full backup is taken. See Section 3 of the *EDB Postgres Backup and Recovery Guide* for the preparation steps and action for taking the full backup.

```
-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 in the output from the preceding example that 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 by the following example:

```
-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.

Also note that 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 example shows invocation of 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 example 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
```

Queries within `psql` 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
```

5 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 information about the operational procedures and BART subcommands, see Section 3.

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`

Password-less 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 show the configuration steps and operation for this system for taking full backups only. (For supporting incremental backups as well, enable the `allow_incremental_backups` parameter for the desired database servers and use the `WAL scanner` program).

5.1 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"
```

5.2 Establishing SSH/SCP Password-Less Connections

This section shows how the password-less SSH/SCP connections were established with the authorized public keys files.

5.2.1 Generating a Public Key File for the BART User Account

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

Generation of the public key file is as follows. First, create the `.ssh` subdirectory in the BART user's home 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
```

Make sure there are no groups or other users that can access the `.ssh` directory.

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
```

5.2.2 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 password-less 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 Section 5.2.1, 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, otherwise the password-less connection fails:

```
-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 password-less 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.
```

5.2.3 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.

This is for scenario 1 as described in Required BART Connections with No Password Section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide*.

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 password-less 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 password-less 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.
```

5.2.4 Configuring Access from BART Host to 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.

This is for scenario 2 as described in Required BART Connections with No Password Section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide*.

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 was generated in Section 5.2.1, and 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 password-less 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.
```

5.2.5 Configuring Access from Remote PostgreSQL to BART Host

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

This is for scenario 1 as described in the *Required BART Connections with No Password* section of the *EDB Postgres Backup and Recovery Installation and Upgrade Guide*.

Create the `.ssh` directory for user account `postgres` 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, otherwise the password-less connection fails.

Test the password-less 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.
```

5.2.6 Configuring Access from the BART Host to Remote PostgreSQL

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, `postgres`, on the remote host 192.168.2.24.

This is for scenario 2 as described in Required BART Connections with No Password section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide*.

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 was generated in Section 5.2.1, and is now copied to the remote PostgreSQL host on 192.168.2.24:

```
-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 password-less 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.
```

5.3 Configuring a Replication Database User

This section shows how the 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 at:

https://www.postgresql.org/docs/current/libpq-pgpass.html

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

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

The pg_hba.conf file for the local Advanced Server, ACCTG is set as follows:

```
# 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
```

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

```
# 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
```

The pg_hba.conf file for the remote PostgreSQL server, HR is set as follows:

```
# 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
# replication privilege.
#local   replication    postgres                                md5
host     replication    postgres        192.168.2.22/32         md5
```

5.4 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, `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, to prevent the `archive_command` from re-setting invoke the `INIT` subcommand with the -- `no configure` option. See Required BART Connections with No Password section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for details.

```
 [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"
```

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 = '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 owning 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, 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.

5.5 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
```

Make sure it 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.

Note:

- 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. See Step 3 in Configuring the BART host section of *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information.
- 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
```

Note: 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.

5.6 Starting the Database Servers with WAL Archiving

After the BART backup catalog directory structure has been completed, begin 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.

5.7 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
```

5.8 Using Point-In-Time Recovery

The following section demonstrates the point-in-time recovery operation on the remote PostgreSQL database server.

The following tables were created about two minutes apart while WAL archiving is 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 lists the current PostgreSQL server WAL files. The unarchived WAL files are indicated with bold font.

```
-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, the directory is created to which the PostgreSQL database cluster is to be restored. This restore path is `/opt/restore_pg95` 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
```

Note: 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:  creating recovery.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 recovery.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/recovery.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 it 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 the following:

```
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
```

Note: 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
(4
```

6. Recent Remediations

The following defects are fixed in this release:

- To restore an incremental backup, user had to specify dropped tablespaces location in the `bart.cfg tablespace_path` configuration. This is now fixed and user can now restore an incremental backup without specifying the dropped tablespaces location.
- While restoring an incremental backup, the following error was displayed. This error is fixed in this release.
  WARNING: relation node size not known:
  `/tmp/restore/base/15099/16388"` in `bart-auto.log.`
- While verifying the database server configuration, BART `check-config` command displayed an error if `wal_level` was set to `hot_standby`. This error is now fixed.
- In the `bart.cfg,` the `batch_size` was set to different default values in the server section and global [BART] section. This is fixed and now the `batch_size` is set to same default value of 3/8 GB in both the global and server sections.
- Duplicate querying of list of tablespaces during an incremental backup caused inconsistency. This error is now fixed.
- Fixed the following error that was displayed while performing a full backup: unexpected null in column 1, "size", at row 1380 for query: WITH RECURSIVE files (path, filename, size, isdir) AS:
- When invalid value was used for `mbm_scan_timeout` parameter in `bart.cfg,` instead of generating an error the default value of 20 was used. This is now fixed and error is generated if invalid value is used.
- When invalid values were used for `thread_count` and `batch-size` parameters in `bart.cfg`, error was not generated. This is now fixed and it displays an error if invalid values are used.
- The error “failed to overlay modified blocks” was displayed while restoring backup. This is now fixed.
- Sometimes while invoking the `BART backup` command, success message was displayed even if the backup was partially completed. This is fixed and now success message will be displayed when the backup is entirely completed.
- Restore operation failed if tablespace was deleted between `pg_start_backup` and `pg_stop_backup`. This is now fixed.
- While taking a backup, if decimal value was used with the – c option, an error was displayed after the backup process started. This is fixed and now if decimal value was used, error will be displayed before the backup process initiates.
- Relation missing error was reported while taking the full backup. This is now fixed.

- This release fixed the following errors that were displayed after taking a parallel backup in tar format:
    - The database server log file displayed an error related to backing up Write ahead log files.
    - Restoration and recovery from parallel backup in tar format failed.