



EDB

**EDB Postgres™ Backup and Recovery
User Guide**

Release 2.6

EDB Postgres™ Backup and Recovery User Guide

Oct 08, 2020

Contents

1	Introduction	1
1.1	What's New	2
1.2	Conventions Used in this Guide	2
1.3	Restrictions on pg_basebackup	3
2	Overview	4
2.1	Block-Level Incremental Backup	8
2.1.1	Incremental Backup Limitations and Requirements	8
2.1.2	Concept Overview	9
2.1.3	WAL Scanning – Preparation for an Incremental Backup	10
2.1.4	Performing an Incremental Backup	11
2.1.5	Restoring an Incremental Backup	12
2.2	Creating a Backup Chain	14
3	Using BART	15
3.1	BART Management Overview	15
3.1.1	Performing a Restore Operation	17
3.1.2	Point-In-Time Recovery Operation	19
3.2	Managing Backups Using a Retention Policy	21
3.2.1	Overview - Managing Backups Using a Retention Policy	21
3.2.2	Marking the Backup Status	22
3.2.3	Setting the Retention Policy	23
3.2.4	Managing the Backups Based on the Retention Policy	24
3.2.5	Managing Incremental Backups	26
3.3	Basic BART Subcommand Usage	28
3.3.1	CHECK-CONFIG	30
3.3.2	INIT	31
3.3.3	BACKUP	33
3.3.4	SHOW-SERVERS	38
3.3.5	SHOW-BACKUPS	39
3.3.6	VERIFY-CHKSUM	40
3.3.7	MANAGE	41
3.3.8	RESTORE	44
3.3.9	DELETE	47
3.4	Running the BART WAL Scanner	48
4	Using Tablespaces	51

5 Conclusion

53

Index

54

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

BART provides the following features:

- Support for complete, hot, physical backups of multiple Advanced Servers and PostgreSQL database servers
- Support for two types of backups – full base backups and block-level incremental backups
- Backup and recovery management of database servers on local or remote hosts
- A single, centralized catalog for backup data
- Retention policy support for defining and managing how long backups should be kept
- The capability to store the backup data in compressed format
- Verified backup data with checksums
- Backup information displayed in an easy-to-read format
- A simplified point-in-time recovery process

This guide provides the following information about using BART:

- an *overview* of the BART components and concepts.
- *backup and recovery management process*.
- *using tablespaces*.

For information about installing BART, see the *EDB Postgres Backup and Recovery Installation and Upgrade Guide*; for examples of BART operations and subcommand usage, see the *EDB Postgres Backup and Recovery Reference Guide*. These guides are available at the [EDB website](#).


1.1 What's New

The following features have been added to BART 2.6:

- Improved Postgres support: EDB Advanced Server version 13 and PostgreSQL version 13.
- Improved platform support: Ubuntu 20.04 LTS (Focal Fossa) package.
- Improved FIPS (Federal Information Processing Standards) supportability:
 - disable-checksum option added to skip verifying the MD5 or SHA256 checksum files
 - checksum-algorithm added option to generate the MD5 and SHA256 checksum files, as well as to skip generating checksum files
 - bart_socket_name option added to configure the bart_socket_name

For more information, please see the *EDB Postgres Backup and Recovery Installation and Upgrade Guide*, available at the [EDB website](#).

- End-of-Support Reminder: Support for PostgreSQL and Advanced Server v9.5 is revoked.
- Improved documentation: Clarifying the use of scan_interval for NFS file systems.
- Improved documentation: The HTML version of the documentation improves the user experience by allowing customers to copy commands to the browser clipboard in one click. The figure below shows the copy icon to the far-right of an installation command; in the HTML version of the guide, you can click the copy icon to copy the command.



```
yum install edb-bart
```

1.2 Conventions Used in this Guide

The following is a list of 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 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, the installation directory is `/var/lib/pgsql/<x.x>`.
 - For Advanced Server Linux installations performed using the interactive installer for version 10 and earlier, this defaults to `/opt/PostgresPlus/<x.x>AS` or `/opt/edb/as<x.x>`. For Advanced Server Linux installations performed with an RPM package, this defaults to `/usr/ppas-<x.x>` or `/usr/edb/as<x.x>`. For Advanced Server Linux installations performed with an RPM package for version 11 or later, this defaults to `/usr/edb/as<xx>`.

1.3 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 is specified with the `BACKUP` subcommand (see *Backup*).
- The number of thread count in effect is 1, and the `with-pg_basebackup` option is not specified with the `BACKUP` subcommand.
- Database servers can only be backed up using `pg_basebackup` utility program 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 back up database server version 9.6.

In the global section of the BART configuration file, the `pg_basebackup_path` parameter specifies the complete directory path to the `pg_basebackup` program. For information about the `pg_basebackup_path` parameter and the `thread_count`, see the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at the [EDB website](#).

For information about `pg_basebackup`, see the [PostgreSQL Core Documentation](#).

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* or referred as *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.
- 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](#).

The general term *backup* refers to both full backups and incremental backups.

When taking a full backup of a standby server, BART uses the PostgreSQL `pg_basebackup` utility program. However, it must be noted that for standby servers, you can only take a full backup, but cannot take an incremental or parallel backups. For information about standby servers, see the [PostgreSQL Documentation](#).

BART uses a centralized backup catalog, a single configuration file, and a command line interface controlling the necessary operations to simplify the management process. 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 using 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 of a BART installation are:

- **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 that BART uses.
- **BART Backup Catalog.** File system directory structure containing all of the backups for the database servers that BART manages. It is also the default `archive_path` to store archived WAL files.
- **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 manage 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. A full backup is a copy of a database cluster.

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`.
The BART configuration parameter `cluster_owner` must be set to the database user account for each database server.
- **Replication Database User.** For each database server that BART manages, 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. The BART configuration parameter `user` must be set to this replication database user for each database server.

- **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 fact is the user account under which the SSH or SCP operations occur.

For information on how all of these components are configured and used with BART, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide available at the [EDB website](#).

Supported BART Operations

The following tables are not a conclusive list of the scenarios supported by BART, but instead provides an overview of some of the most common scenarios in both `pg_basebackup` (thread count=1) as well as parallel backup mode (thread count greater than 1).

Table 1: Backup

	-Fp-xlog-method=fetch	-Fp-xlog-method=stream	-Ft-xlog-method=fetch	-Ft-xlog-method=stream
Primary Database Server/Full backup	Supported	Supported	Supported	Supported
Primary Database Server/Incremental backup	Supported	Supported	Not Supported	Not Supported
Standby Database Server/Full backup	Supported	Supported	Supported	Supported
Standby Database Server/Incremental backup	Not Supported	Not Supported	Not Supported	Not Supported

Table 2: Wal Archiving

	Wal compression by BART	WAL scanner
Primary Database Server/Full backup	Supported	N/A
Primary Database Server/Incremental backup	Not Supported	N/A
Standby Database Server/Full backup	Supported	N/A
Standby Database Server/Incremental backup	Not Supported	N/A

Table 3: Restore

	Wal compression = enabled	Wal compression = disabled
Restore	Supported	Supported
Parallel restore	Supported	Supported

2.1 Block-Level Incremental Backup

This section describes the basic concepts of a block-level incremental backup (referred to as an incremental backup). An incremental backup is a unique functionality of BART.

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

- 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.

Generally, all BART features (such as retention policy management) apply to incremental backups and full backups. See *Managing Incremental Backups* for more information.

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 a primary node, you must take a full backup before enabling incremental backup on the cluster.
- On a standby database server, you cannot take an incremental backup.

You must meet the following requirements before implementing incremental backup:

- You must create or select an operating system account to be used as the BART user account.
- You must create or select the replication database user, which must be a superuser.
- In the BART configuration file:
 - You must set the `cluster_owner` parameter to the Linux operating system user account that owns the database cluster directory from which incremental backups are to be taken.
 - You must enable the `allow_incremental_backups` parameter.
- A passwordless 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.

It must be noted that a passwordless 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.

- 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. To restore an incremental backup, the `bart` program must be executable on the remote host by the remote user (the remote 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).
- When *restoring incremental backups on a remote database server*, a passwordless SSH/SCP connection must be established from the BART user account on the BART host to the remote user on the remote host (the remote 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).
- 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 be disabled for those database servers.

- 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.

For information about configuring these requirements, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide available at the [EDB website](#).

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, and enable and initiate archiving of WAL files to the `archive_path` in the same manner as done for full backups.

The default `archive_path` is the BART backup catalog (`<backup_path>/<server_name>/archived_wals`). Using the `<archive_path>` parameter in the server section of the BART configuration file, you can specify the location where WAL files will be archived.

For more information about the `archive_path` parameter and configuring BART, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide available at the [EDB website](#).

3. Take an initial full backup with the `BACKUP` subcommand. This full backup establishes the parent of the first incremental backup.
4. Scan all WAL files produced by database servers on which incremental backups are to be taken. These WAL files are scanned once they have been archived to the `archive_path`.

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. Take incremental backups 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.

Multiple block copier threads can be used to copy the modified blocks to the incremental backup directory. See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at the [EDB website](#) for information about setting the `thread_count` parameter in the BART configuration file. See *Backup* for information about using the `--thread-count` option with the `BACKUP` subcommand.

7. Invoke the restore process for an incremental backup 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. 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 scanner program (`bart-scanner`) scans the WAL files created from the time of the parent backup up to the start of the incremental backup to determine which blocks have modified since the parent backup, 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 directory where `archived_wals` will be stored, as specified in the `archive_path` parameter in the `bart.cfg` file. If the `archive_path` is not specified, the default `archived_wals` directory is:

```
<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 in the server section of the BART configuration file.

The following code snippet 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 0000000100000001780000280000000179000000.mbm
-rw-rw-r-- 1 ... .. 684 Oct 12 09:49 000000010000000179000028000000017A000000.mbm
-rw-rw-r-- 1 ... .. 161 Oct 12 09:49 00000001000000017A000028000000017B000000.mbm
-rw-rw-r-- 1 ... .. 161 Oct 12 09:49 00000001000000017B000028000000017C000000.mbm
-rw-rw-r-- 1 ... ..1524 Oct 12 09:49 00000001000000017C000028000000017D000000.mbm
-rw-rw-r-- 1 ... .. 161 Oct 12 09:49 00000001000000017D000028000000017E000000.mbm
-rw-rw-r-- 1 ... .. 161 Oct 12 09:49 00000001000000017E000028000000017F000000.mbm
-rw-rw-r-- 1 ... .. 161 Oct 12 09:49 00000001000000017F0000280000000180000000.mbm
```

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 `archive_path`. Thus, the WAL scanner should be running as soon as the WAL files from the database cluster are archived to the `archive_path`. If the `archive_path` 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.

Under certain conditions such as when the Network File System (NFS) is used to copy WAL files to the `archive_path`, 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 initiate force scanning of WAL files in the `archive_path` to ensure MBM files are generated. See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at the [EDB website](#) for more information about the `scan_interval` parameter.

See *Running the BART WAL Scanner* 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 `archived_wals` directory as specified in the `archive_path` parameter in the `bart.cfg` file. When compared to a full 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.

Before taking an incremental backup, an initial full backup must be taken with the `BACKUP` subcommand. This full backup establishes the parent of the first incremental backup.

The syntax for taking a full backup is:

```
bart BACKUP -s { <server_name> | all } [ -F { p | t } ]
[ -z ] [ -c <compression_level> ]
[ --backup-name <backup_name> ]
[ --thread-count <number_of_threads> ]
[ { --with-pg_basebackup | --no-pg_basebackup } ]
[--checksum-algorithm ]
```

The syntax for taking an incremental backup is:

```
bart BACKUP -s { <server_name> | all } [ -F p]
[ --parent { <backup_id> | <backup_name> } ]
[ --backup-name <backup_name> ]
[ --thread-count <number_of_threads> ]
[ --check ]
[ --checksum-algorithm ]
```

You must specify the following before taking an incremental backup:

- `-Fp` option for plain text format as incremental backup can only be taken in the plain text format.
- `--check` option to verify if the required MBM files are present in the `archived_wals` directory. The `--parent` option must be specified when the `--check` option is used.

See *BACKUP* for more information about using the *BACKUP* subcommand.

2.1.5 Restoring an Incremental Backup

Restoring an incremental backup may require additional steps depending upon the host on which the incremental backup is to be restored:

- *Restoring an Incremental Backup on a BART Host* - This section outlines restoring an incremental backup onto the same host where BART has been installed.
- *Restoring an Incremental Backup on a Remote Host* - This section outlines restoring an incremental backup onto a remote host where BART has not been installed.

Restoring an Incremental Backup on a BART Host

Specify a backup identifier or name, and include the `-i` option when invoking the *RESTORE* subcommand to restore an incremental backup. All *RESTORE* options may be used in the same manner as when restoring a full backup. See *RESTORE* command for more details.

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.

Also note that you can use 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.

Note: If you set the *BART scanner* or *backup* with the `--checksum-algorithm=NONE` option, then you must specify the `--disable checksum` option while restoring an incremental backup.

Restoring an Incremental Backup on a Remote Host

Ensure the `bart` program is available on the remote host when restoring an incremental backup on a remote host; the invocation of the `RESTORE` subcommand 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.

To restore an incremental backup onto a remote host where BART has not been installed, perform the following steps:

Step 1: Install BART on the remote host to which an incremental backup is to be restored.

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 passwordless SSH/SCP connection is established from the BART user on the BART host to the remote user on the remote host. For information about creating a passwordless SSH/SCP connection, see the *EDB Postgres Backup and Recovery Installation and Upgrade Guide*, available at the [EDB website](#).

When restoring an incremental backup, specify the backup identifier or name of the incremental backup that will be restored. See the *RESTORE* documentation for more details. To view an example of restoring an incremental backup, see the *EDB Postgres Backup and Recovery Reference Guide* available at the [EDB website](#).

Note: If you set the *BART scanner* or *backup* with the `--checksum-algorithm=NONE` option, then you must specify the `--disable_checksum` option while restoring an incremental backup.

2.2 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 the *EDB Postgres Backup and Recovery Reference Guide*, available at the [EDB website](#) for examples of creating a backup chain and restoring an incremental backup.

After installing and configuring the BART host and the database servers, you can start using BART.

This section describes how to perform backup and recovery management operations using BART. Review the sections that follow before proceeding with any BART operation.

3.1 BART Management Overview

After configuring BART, you can begin the backup and recovery management process. The following steps will help you get started:

1. Run the `CHECK-CONFIG` subcommand without the `-s` option. When the `CHECK-CONFIG` subcommand is used without specifying the `-s` option, it checks the parameters in the global section of the BART configuration file.
2. Run the `INIT` subcommand (if you have not already done so) 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 `archived_wals` subdirectory should exist for each database server.
3. Start the Postgres database servers with archiving enabled. Verify that the WAL files are appearing in the `archive_path`. 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. Start the WAL scanner if you intend to take incremental backups. Since the WAL scanner processes the WAL files copied to the `archive_path`, it is advantageous to commence the WAL scanning as soon as the WAL files begin to appear in the `archive_path` 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 is 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:

- Execute the `BACKUP` subcommand to create additional full backups or incremental backups.
- Use the `VERIFY-CHKSUM` subcommand to verify the checksum of the full backups.
- Display database server information with the `SHOW-SERVERS` subcommand.
- Display backup information with the `SHOW-BACKUPS` subcommand.
- Compress the archived WAL files in the `archive_path` 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 Performing a Restore Operation

The following steps describe the process of restoring a backup:

Step 1: Use your system-specific command to shut down the database server.

Step 2: Inspect the `pg_wal` subdirectory (inspect the `pg_xlog` subdirectory if you are using server 9.6 version) of the data directory and save any WAL files that have not yet been archived to the `archive_path`. If there are files that have not been archived, save them to a temporary location.

Step 3: If you want to restore to current data directory, it is recommended to make a copy of the current data directory and then delete all files and subdirectories under the data directory if you have enough extra space. If there is not enough space, then make a copy of `pg_wal` directory (or `pg_xlog` if you are using server 9.6 version) until the server is successfully restored.

If you want to restore to a new, empty directory, create the directory on which you want to restore the backed up database cluster. Ensure 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 the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at the [EDB website](#) for more information about this parameter.

Step 5: Identify the backup to use for the restore operation and obtain the backup ID or backup name.

To use the latest backup, omit the `-i` option; the `RESTORE` subcommand uses that backup by default. The backups can be listed with the `SHOW-BACKUPS` subcommand.

Step 6: Run the BART `RESTORE` subcommand.

- Minimal recovery settings will be saved in the `postgresql.auto.conf` file and archive recovery will proceed only until consistency is reached, with no restoration of files from the archive. See *Restore* for detailed information about `Restore` subcommand.
- If the `-c` option is specified or if the `copy_wals_during_restore` BART configuration parameter is enabled 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 recovery settings are saved in the `postgresql.auto.conf` file, 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`

You must ensure that valid options are specified when using the `RESTORE` subcommand. BART will not generate an error message if invalid option values or invalid option combinations are provided. BART will accept the invalid options and pass them to the `postgresql.auto.conf` file, which will then be processed by the database server when it is restarted.

Step 7: Copy any saved WAL files from Step 2 to the `restore_path/pg_xlog` subdirectory.

Step 8: 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. 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 9: The `postgresql.auto.conf` file should be configured to recover only until the cluster reaches consistency. In either case, the settings may be modified as desired.

Step 10: Disable WAL archiving at this point. The BART `RESTORE` subcommand adds `archive_mode = off` to the end of the `postgresql.conf` file.

- If you want to restart the database server with WAL archiving enabled, ensure that this additional parameter is deleted.
- The original `archive_mode` parameter still resides in the `postgresql.conf` file in its initial location with its last setting.

Step 11: Start the database server to initiate recovery. After completion, check the database server log file to ensure the recovery was successful.

If the backup is restored to a different location than where the original database cluster resided, operations dependent upon the database cluster location may fail if supporting service scripts are not updated to reflect the location where the backup has been restored. For information about the use and modification of service scripts, see the EDB Postgres Advanced Server Installation Guide available at the [EDB website](#).

See *Restore* for more information about using the BART `Restore` subcommand.

An example of a restore operation is documented in the EDB Postgres Backup and Recovery Reference Guide available at the [EDB website](#).

Note: If you set the `backup --checksum-algorithm=NONE` option, then you must specify the `--disable-checksum` option while restoring a backup.

3.1.2 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. If you want to:
 - a. restore the database cluster and tablespace files to new, empty directories, create the new directories with the appropriate directory ownership and permissions.
 - b. reuse the existing database cluster directories, delete all the files and subdirectories in the existing directories. We strongly recommend that you make a copy of this data before deleting it. Be sure to save any recent WAL files in the `pg_wal` subdirectory (`pg_xlog` subdirectory if you are using server 9.6 version) that have not been archived to `archive_path`.
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 provide 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 `archive_path` are copied to the `restore_path/archived_wals` subdirectory.

Ensure the `restore_path` directory and all subdirectories and files in the `restore_path` 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.

Use the `chown` command to make the appropriate adjustments to file permissions; for example, the following command changes the ownership of `restore_path` to `enterprisedb`:

```
chown -R enterprisedb:enterprisedb restore_path
```

The following command restricts access to `restore_path`:

```
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_wal` subdirectory (`pg_xlog` subdirectory if you are using server 9.6 version).
6. 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.

7. Verify that the `postgresql.auto.conf` file created in the directory specified with the `RESTORE` subcommand's `-p <restore_path>` option was generated with the correct recovery parameter settings.

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.

8. 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.

9. Start the database server, which will then perform the point-in-time recovery operation if recovery settings are saved in the `postgresql.auto.conf` file.

For a detailed description of the `RESTORE` subcommand, see *Basic BART Subcommand Usage*. An example of a Point-in-Time Recovery operation is documented in the *EDB Postgres Backup and Recovery Reference Guide* available at the [EDB website](#). See *Restore* for more information about using the `Restore` subcommand.

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 done and 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.
2. Use the `MANAGE` subcommand to categorize and manage backups according to the retention policy.
3. Create a cron job to periodically run the `MANAGE` subcommand to evaluate the backups and then list and/or delete the obsolete backups.

Retention policy management applies differently to incremental backups than to full backups. See *Managing Incremental Backups* for information about how retention policy management is applied to each backup type.

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

3.2.1 Overview - Managing Backups Using a Retention Policy

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 and recovery window retention policy.

- **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).
- **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).
 - 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 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. Use the `MANAGE` subcommand to evaluate if the backup status should be changed to obsolete in accordance with the retention policy. You can review the current status of your backups with the `SHOW-BACKUPS` subcommand.

Active backups are evaluated and also marked (that is, internally recorded by BART) as obsolete only when the `MANAGE` subcommand is invoked either with no options or with only the `-s` option.

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

- Use the `MANAGE` subcommand with the `-c` option along with the backup identifier or name with the `-i` option. 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.

After setting the `retention_policy` parameter and running the `MANAGE` subcommand if you change the `retention_policy` parameter, the current, marked status of the backups are probably inconsistent with the new `retention_policy` setting. To modify the backup status to be consistent with the new `retention_policy` setting, you need to run the `MANAGE` subcommand with:

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

See [MANAGE](#) for usage information for the `MANAGE` subcommand.

3.2.3 Setting the Retention Policy

The retention policy is determined by the `retention_policy` parameter in the BART configuration file. It can be applied globally to all servers, but each server can override the global retention policy with its own. For information about creating a global retention policy and an individual database server retention policy, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide available at the [EDB website](#).

There are two types of retention policies - redundancy retention policy and the recovery window retention policy as described in the following sections.

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.

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 Backups` 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"
```

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.

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 is equal to or greater than the start of the recovery window date/time.

You can view the actual, calculated recovery window by:

- Invoking the `MANAGE` subcommand in debug mode, along with the `-n` option.
- Using the `SHOW-SERVERS` subcommand.

3.2.4 Managing the Backups Based on the Retention Policy

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 `active`. You can use the `MANAGE` subcommand to change the status of an active backup to `obsolete`. Obsolete backups can then be deleted.

This section covers following aspects of backup management:

- The rules for *deleting backups* depending upon the backup status and the subcommand used.
- The process to retain a backup indefinitely by *marking it as keep*. This section also provides information about resetting backups status (that are marked as `obsolete` and `keep`) back to active status.
- The general process for *evaluating, marking, and then deleting obsolete backups*.

Deletions Permitted Under a Retention Policy

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

You must use the `MANAGE` subcommand to delete obsolete backups. Use the `DELETE` subcommand 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:

- MANAGE subcommand: 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. The deletion of backups relies on the last occasion when the backups have been marked.
- DELETE subcommand:
 - 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.

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

Note: Redundancy Retention Policy (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: Recovery Window Retention Policy (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.

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.

Evaluating, Marking, and Deleting Obsolete Backups

When the `MANAGE` subcommand is invoked, BART evaluates active backups:

- If you include the `-s` option when invoking the `MANAGE` subcommand, BART evaluates backups for the database server.
- If you include the `-s all` option when invoking the `MANAGE` subcommand, BART evaluates backups for all database servers.
- If the `-s` option is omitted, the command evaluates the current number of backups for the database server based on the redundancy retention policy or the current date/time for a recovery window retention policy.

Note: The status of backups currently marked as `obsolete` or `keep` is not changed. To re-evaluate such backups and then classify them, their status must first be reset to `active` with the `MANAGE -c nokeep` option. See [Marking the Backup Status](#) for more information.

See the *EDB Postgres Backup and Recovery Reference Guide* available at the [EDB website](#) to review examples of how to evaluate, mark, and delete backups using a redundancy retention policy and recovery window retention policy, as well as examples of `MANAGE` subcommand.

3.2.5 Managing Incremental Backups

The following section summarizes how retention policy management affects incremental backups.

- The retention policy rules are applied to full backups.
 - A redundancy retention policy uses the number of full backups to determine if a backup is obsolete. Incremental backups are excluded from the comparison count against the `retention_policy` setting for the maximum number of backups.
 - A recovery window retention policy uses the backup date/time of any full backups to determine if a backup is obsolete. 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.

- **MANAGE** subcommand:
 - 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.
- **DELETE** subcommand:
 - 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.

Using a Redundancy Retention Policy with Incremental Backups

When a *redundancy retention policy* is used and the **MANAGE** subcommand is invoked, the status of the oldest `active` full backup 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.

Note: When a full backup is changed from `active` to `obsolete`, all successive incremental backups in the chain of the 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. Incremental backups are not included in the count for the comparison against the `retention_policy` parameter setting.

See the *EDB Postgres Backup and Recovery Reference Guide* available at the [EDB website](#) for examples demonstrating use of the **MANAGE** and **DELETE** subcommands when a redundancy retention policy is in effect.

Using a Recovery Window Retention Policy with Incremental Backups

If the **MANAGE** command is invoked when BART is configured to use a *recovery window retention policy*, the status of `active` full backups are changed to `obsolete` if the date/time of the full backup is outside of the recovery window.

Note: If a full backup is changed from `active` to `obsolete`, all successive incremental backups in the chain of the 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.

See the *EDB Postgres Backup and Recovery Reference Guide* available at the [EDB website](#) for examples demonstrating use of the **MANAGE** and **DELETE** subcommands when a recovery window retention policy is in effect.

3.3 Basic BART Subcommand Usage

This section briefly describes the BART subcommands and options. You can invoke the `bart` program (located in the `<BART_HOME>/bin` directory) with the desired options and subcommands to manage your BART installation.

To view examples of BART subcommands, see the *EDB Postgres Backup and Recovery Reference Guide* available at the [EDB website](#).

Syntax for invoking BART:

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

- When invoking a subcommand, the subcommand name is not case-sensitive (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.
- Keywords are case-sensitive; options are generally specified in lowercase unless specified otherwise in this section.
- When invoking BART, the current user must be the BART user account (operating system user account used to run the BART command line program). 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 chosen operating system user account must own the BART backup catalog directory, be able to run the `bart` program and the `bart_scanner` program, and have a passwordless SSH/SCP connection established between database servers managed by BART.

You can specify one or more of the following general options:

Options	Description
<code>-h</code> <code>--help</code>	Displays general syntax and information on BART usage. All subcommands support a help option (<code>-h</code> , <code>--help</code>). If the help option is specified, information is displayed regarding that particular subcommand. The subcommand, itself, is not executed.
<code>-v</code> <code>--version</code>	Displays the BART version information.
<code>-d</code> <code>--debug</code>	Displays debugging output while executing BART subcommands.
<code>-c <config_file_path></code> <code>--config-path</code> <code><config_file_path></code>	Specifies <code>config_file_path</code> 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 <code><BART_HOME>/etc/bart.cfg</code> .

Troubleshooting: 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
```

Workaround: 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`.

It is suggested that the `PATH` and the `LD_LIBRARY_PATH` environment variable settings be placed in the BART user account's profile. See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at the [EDB website](#) for details.

In the following sections, the `help` option is omitted from the syntax diagrams for the purpose of providing readability for the subcommand options.

3.3.1 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.

Syntax:

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

The following table describes the option.

Options	Description
<code>-s <server_name></code> <code>--server</code> <code><server_name></code>	<code>server_name</code> 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.

- When the `-s` option is omitted, the global section [BART] parameters including `bart_host`, `backup_path`, and `pg_basebackup_path` are checked.
- When the `-s` option is specified, the server section parameters are checked. In addition, certain database server `postgresql.conf` parameters are also checked, which include the following:
 - The `cluster_owner` parameter must be set to the user account owning the database cluster directory.
 - A passwordless 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 `archive_path` 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.

3.3.2 INIT

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

Note: If the archive_mode configuration parameter is set to off, then the -o option must be used to set the Postgres archive_command using the BART archive_command parameter in the BART configuration file even if the archive_command is not currently set in postgresql.conf nor in postgresql.auto.conf file.

Syntax:

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

All subcommand options are generally specified in lowercase. The following table describes the command options:

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

Archive Command Setting

After the `archive_command` is set, you need to either restart the PostgreSQL server or reload the configuration file in the PostgreSQL server based on the following conditions.

- 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`
- 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 configuration in the PostgreSQL server using `pg_reload_conf()` or `pg_ctl reload`.
- 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`
- 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.

3.3.3 BACKUP

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

Syntax for full backup:

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

Note: While taking a backup, if a file (for example, database server log file) exceeding 1 GB size is stored in the \$PGDATA directory, the backup will fail. To avoid such backup failure, you need to store large files (exceeding 1 GB) outside the \$PGDATA directory.

Syntax for incremental Backup:

```
bart BACKUP -s { <server_name> | all } [ -F p ]
  [ --parent { <backup_id> | <backup_name> } ]
  [ --backup-name <backup_name> ]
  [ --thread-count <number_of_threads> ]
  [ --check ]
  [--checksum-algorithm ]
```

Note: To take an *incremental backup*, you must take a full backup first followed by incremental backup.

Please Note:

- While a BACKUP subcommand is in progress, no other subcommands must be invoked. Any subcommands invoked while a backup is in progress will skip and ignore the backups.
- For full backup, the target default format is a tar file, whereas for incremental backup, only plain format must be specified.
- The backup is saved in the <backup_path>/<server_name>/<backup_id> directory, 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.
- Before performing the backup, BART checks to ensure if there is enough disk space to completely store the backup in the BART backup catalog.
- In the postgresql.conf file, ensure the wal_keep_segments configuration parameter is set to a sufficiently large value. A low setting of the wal_keep_segments configuration parameter may result in the deletion of some WAL files before the BART BACKUP subcommand saves them to the archive_path. For information about the wal_keep_segments parameter, see the [PostgreSQL Core Documentation](#).
- In the BART configuration file, setting xlog_method=stream will instruct the server to stream the transaction log in parallel with creation of the backup for a specific database server; otherwise the transaction log files are collected upon completion of the backup. See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at the [EDB website](#) for details about database server setting.

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

- When you use BART to take a backup of PostgreSQL server, multiple backups can be taken simultaneously and if a backup is interrupted, the backup mode is terminated automatically without the need to run `pg_stop_backup()` command manually to terminate the backup.

Options

Along with the `BACKUP` subcommand, specify the following option:

Options	Description
<pre>-s { server_name all } --server { server_name all }</pre>	<p><code>server_name</code> is the database server name to be backed up as specified in the BART configuration file. If <code>all</code> is specified, all servers are backed up. This option is mandatory.</p> <p>If <code>all</code> 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.</p>

Specify the following options as required. If you do not specify any of the following options, the backup is created using default settings.

Options	Description
<pre>-F { p t } --format { p t }</pre>	<p>Specify this option to provide the backup file format. Use <code>p</code> for plain text or <code>t</code> for tar. If the option is omitted, the default is tar format.</p> <p>For taking incremental backups, the option <code>-Fp</code> must be specified.</p>
<pre>-z --gzip</pre>	<p>This is applicable only for full backup. Specify this option to use gzip compression on the tar file output using the default compression level. This option is applicable only for the tar format.</p>
<pre>-c <compression_level> --compress-level <compression_level></pre>	<p>This is applicable only for full backup. Specify this option to use the gzip compression level on the tar file output. <code>compression_level</code> is a digit from 1 through 9, with 9 being the best compression. This option is applicable only for the tar format.</p>

continues on next page

Table 5 – continued from previous page

Options	Description
<pre>--parent { backup_id backup_name }</pre>	<p>Specify this option to take an incremental backup. <backup_id> is the backup identifier of a parent backup. <backup_name> is the user-defined alphanumeric name of a parent backup.</p> <p>The parent is a backup taken prior to the incremental backup. The parent backup can be either a full backup or an incremental backup.</p> <p>The option <code>-Fp</code> must be specified since an incremental backup can only be taken in plain text format.</p> <p>An incremental backup cannot be taken on a standby database server. See Block-Level Incremental Backup for additional information on incremental backups.</p>
<pre>--backup-name <backup_name></pre>	<p>Specify this option to assign a user-defined, alphanumeric friendly name to the backup. The maximum permitted length of backup name is 49 characters.</p> <p>The backup name may include the following variables to be substituted by the timestamp values when the backup is taken: 1) <code>%year</code> – 4-digit year, 2) <code>%month</code> – 2-digit month, 3) <code>%day</code> – 2-digit day, 4) <code>%hour</code> 2-digit hour, 5) <code>%minute</code> – 2-digit minute, and 6) <code>%second</code> – 2-digit second.</p> <p>To include the percent sign (<code>%</code>) as a character in the backup name, specify <code>%%</code> in the alphanumeric string.</p> <p>If the backup name contains space characters (i.e. more than one word) or when referenced with the option <code>-i</code> by other subcommands (such as <code>restore</code>), enclose the string in single quotes or double quotes. See backup name examples.</p> <p>If the <code>--backup-name</code> option is not specified, and the <code>backup_name</code> 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.</p>
<pre>--thread-count <number_of_threads></pre>	<p>Use this option to use the number of worker threads to run in parallel to copy blocks for a backup.</p> <p>If the option <code>--thread-count</code> is omitted, then the <code>thread_count</code> parameter in the BART configuration file applicable to this database server is used.</p> <p>If the option <code>--thread-count</code> is not enabled for this database server, then the <code>thread_count</code> setting in the global section of the BART configuration file is used.</p> <p>If the option <code>--thread-count</code> is not set in the global section as well, the default number of threads is 1.</p> <p>If parallel backup is run with N number of worker threads, then it will initiate N+ 1 concurrent connections with the server.</p> <p>Thread count will not be effective if backup is taken on a standby server.</p> <p>For more information about the <code>--thread-count</code> parameter, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide available at the EDB website</p>

continues on next page

Table 5 – continued from previous page

Options	Description
<code>--with-pg_basebackup</code>	<p>This is applicable only for full backup. Specify this option to use <code>pg_basebackup</code> to take a full backup. The number of thread counts in effect is ignored as given by the <code>thread_count</code> parameter in the BART configuration file.</p> <p>When taking a full backup, if the thread count in effect is greater than 1, then the <code>pg_basebackup</code> utility is not used to take the full backup (parallel worker threads are used) unless the option <code>--with-pg_basebackup</code> is specified with the <code>BACKUP</code> subcommand.</p>
<code>--no-pg_basebackup</code>	<p>This is applicable only for full backup. Specify this option if you do not want <code>pg_basebackup</code> to be used to take a full backup.</p> <p>When taking a full backup, if the thread count in effect is only 1, then the <code>pg_basebackup</code> utility is used to take the full backup unless the option <code>--no-pg_basebackup</code> is specified with the <code>BACKUP</code> subcommand.</p>
<code>--check</code>	<p>This is applicable only for incremental backup. Specify this option to verify if the required MBM files are present in the <code>archived_wals</code> directory as specified in the <code>archive_path</code> parameter in the <code>bart.cfg</code> file before taking an incremental backup. The option <code>--parent</code> must be specified when the option <code>--check</code> is used. An actual incremental backup is not taken when the option <code>--check</code> is specified.</p>
<code>--checksum-algorithm</code>	<p>While taking a backup, you can specify one of the following values with the <code>--checksum-algorithm</code> option:</p> <ul style="list-style-type: none"> <code>--checksum-algorithm=MD5</code> (default) to generate MD5 checksum files. <code>--checksum-algorithm=SHA256</code> to generate SHA256 checksum files. <code>--checksum-algorithm=NONE</code> to skip generating checksum files.

–backup-name Examples

The following examples demonstrate using the `–backup-name` clause:

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

Error messages

The following table lists the error messages that may be encountered when using the `BACKUP` subcommand:

error message	Cause
<pre>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</pre>	<p>Insufficient free disk space.</p>
<pre>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</pre>	<p>The <code>wal_keep_segments</code> configuration parameter is not set to a sufficiently large value in the <code>postgresql.conf</code> file.</p>
<pre>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?</pre>	<p>A connection to a database server listed in the BART configuration file fails. As a result the backup for that database server is skipped, but the backup operation continues for other database servers</p>

3.3.4 SHOW-SERVERS

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

Syntax:

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

The following table describes the command options.

Options	Description
<pre>-s { <server_name> all } --server { <server_name> all }</pre>	<p><server_name> is the name of the database server whose BART configuration information is to be displayed. If <code>all</code> is specified or if the option is omitted, information for all database servers is displayed.</p>

3.3.5 SHOW-BACKUPS

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

Syntax:

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

The following table describes the command options:

Options	Description
-s { <server_name> all } --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 with the exception as described by the following 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.
-i { <backup_id> <backup_name> all } --backupid { <backup_id> <backup_name> all }	<backup_id> is a backup identifier and <backup_name> is the user-defined alphanumeric name for the backup. If all is specified or if the option is omitted, all backup information for the relevant database server is displayed.
-t --toggle	Displays more backup information in a list format. If the option is omitted, the default is a tabular format.

3.3.6 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. The checksum is verified by comparing the current checksum of the backup against the checksum when the backup was taken.

Note: The VERIFY-CHKSUM subcommand is only used for tar format backups. It is not applicable to plain format backups.

Syntax:

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

The following table describes the command options:

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

3.3.7 MANAGE

The `MANAGE` subcommand can be invoked to:

- Evaluate backups, mark their status, and delete obsolete backups based on the `retention_policy` parameter in the BART configuration file (See *Managing Backups Using a Retention Policy* for information about retention policy management).
- Compress the archived WAL files based on the `wal_compression` parameter in the BART configuration file (See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at the [EDB website](#) for information about setting this parameter).

Syntax:

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

The following summarizes the actions performed when the `MANAGE` subcommand is invoked:

- When the `MANAGE` subcommand is invoked with no options or with only the `-s <server_name>` or `-s all` 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), 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, 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.
- 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 `copy_wals_during_restore` BART configuration parameter is enabled 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:
 - * the archived WAL files are transmitted across the network to the remote host in compressed format 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 or the `RESTORE -r` option (see *RESTORE*).
 - * 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 `copy_wals_during_restore` BART configuration parameter is disabled 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 saved to the remote host by the `restore_command` in the `postgresql.auto.conf` file when the database server archive recovery begins.

The following table describes the command options:

Options	Description
<code>s { <server_name> all }</code> <code>--server { <server_name> all }</code>	<code><server_name></code> is the name of the database server to which the actions are to be applied. If <code>all</code> is specified or if the <code>-s</code> option is omitted, the actions are applied to all database servers.
<code>-l</code> <code>--list-obsolete</code>	Lists the backups marked as <code>obsolete</code> .
<code>-d</code> <code>--delete-obsolete</code>	Delete the backups marked as <code>obsolete</code> . This action physically deletes the backup along with its archived WAL files and any MBM files for incremental backups.
<code>-c { keep nokeep }</code> <code>--change-status { keep nokeep }</code>	Specify <code>keep</code> to change the status of a backup to <code>keep</code> to retain it indefinitely. Specify <code>nokeep</code> to change the status of any backup back to active status. The backup can then be re-evaluated and possibly be marked to <code>obsolete</code> according to the retention policy by subsequent usage of the <code>MANAGE</code> subcommand. The <code>-i</code> option must be included when using the <code>-c</code> option.

continues on next page

Table 9 – continued from previous page

Options	Description
<pre>-i { <backup_id> <backup_name> all } --backupid { <backup_id> <backup_name> all }</pre>	<p><backup_id> is a backup identifier and <backup_name> is the user-defined alphanumeric name for the backup.</p> <p>If <code>all</code> is specified, then actions are applied to all backups.</p> <p>The <code>-c</code> option must be included when using the <code>-i</code> option.</p>
<pre>-n --dry-run</pre>	<p>Performs the test run and displays the results prior to actually implementing the actions as if the operation was performed, however, no changes are actually made.</p> <p>If <code>-n</code> is specified with the <code>-d</code> option, it displays which backups would be deleted, but does not actually delete the backups.</p> <p>If <code>-n</code> is specified with the <code>-c</code> option, it displays the keep or nokeep action, but does not actually change the backup from its current status.</p> <p>If <code>-n</code> is specified alone with no other options, or with only the <code>-s</code> option, it displays which active backups would be marked as obsolete, but does not actually change the backup status. In addition, no compression is performed on uncompressed, archived WAL files even if WAL compression is enabled for the database server.</p>

3.3.8 RESTORE

The RESTORE subcommand restores a backup and its archived WAL files for the designated database server to the specified directory location. If the appropriate RESTORE options are specified, all recovery settings will be saved in the `postgresql.auto.conf` file.

Syntax:

```
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 ]
[ --disable-checksum ]
```

For information about using a continuous archive backup for recovery, see the [PostgreSQL Core Documentation](#). 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 `postgresql.auto.conf` file by BART.

Please note:

- For special requirements when restoring an incremental backup to a remote database server, see [Restoring an Incremental Backup on a Remote Host](#).
- 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 may result in an error while copying files if there is not enough disk space available.
- See [Performing a Restore Operation](#) to view steps on how to perform a restore operation and see [Point-In-Time Recovery Operation](#) to view steps on how to perform a point-in-time recovery operation.
- If the backup is restored to a different database cluster directory than where the original database cluster resided, certain operations dependent upon the database cluster location may fail. This happens if their supporting service scripts are not updated to reflect the new directory location of restored backup. For information about the usage and modification of service scripts, see the *EDB Postgres Advanced Server Installation Guide* available at the [EDB website](#).

The following table describes the command options:

Options	Description
<code>-s <server_name></code> <code>--server <server_name></code>	<code><server_name></code> is the name of the database server to be restored.
<code>-p <restore_path></code> <code>--restore-path <restore_path></code>	<code><restore_path></code> 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.
<code>-i {<backup_id> <backup_name>}</code> <code>--backupid {<backup_id> <backup_name>}</code>	<code><backup_id></code> is the backup identifier of the backup to be used for the restoration and <code><backup_name></code> is the user-defined alphanumeric name for the backup. If the option is omitted, the default is to use the latest backup.

continues on next page

Table 10 – continued from previous page

Options	Description
<pre>-r <remote_user@remote_host_address> --remote-host <remote_user@remote_host_address></pre>	<p><code><remote_user></code> is the user account on the remote database server host that accepts a passwordless SSH/SCP login connection and is the owner of the directory where the backup is to be restored and <code><remote_host_address></code> is the IP address of the remote host to which the backup is to be restored. This option must be specified if the <code><remote_host></code> parameter for this database server is not set in the BART configuration file.</p> <p>If the BART user account is not the same as the operating system account owning the <code><restore_path></code> directory given with the <code>-p</code> option, use the <code><remote_host></code> BART configuration parameter or the RESTORE subcommand <code>-r</code> option to specify the <code><restore_path></code> directory owner even when restoring to a directory on the same host as the BART host.</p> <p>See the <i>EDB Postgres Backup and Recovery Installation and Upgrade Guide</i> available at the EDB website for information about the <code><remote_host></code> parameter.</p>
<pre>-w <number_of_workers> --workers <number_of_workers></pre>	<p><code><number_of_workers></code> 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.</p> <p>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 <code>-w</code> option is omitted, the default is 11 worker process.</p>
<pre>-t <timeline_id> --target-tli <timeline_id></pre>	<p><code><timeline_id></code> is the integer identifier of the timeline to be used for replaying the archived WAL files for point-in-time recovery.</p>
<pre>-x <target_xid> --target-xid <target_xid></pre>	<p><code><target_xid></code> is the integer identifier of the transaction ID that determines the transaction up to and including, which point-in-time recovery encompasses. Include either the <code>-x <target_xid></code> or the <code>--target-xid <target_xid></code> option if point-in-time recovery is desired.</p>
<pre>-g <target_timestamp> --target-timestamp <target_timestamp></pre>	<p><code><target_timestamp></code> is the timestamp that determines the point in time up to and including, which point-in-time recovery encompasses. Include either the <code>--target-timestamp <target_timestamp></code> or the <code>-g <target_timestamp></code> option if point-in-time recovery is desired.</p>

continues on next page

Table 10 – continued from previous page

Options	Description
<p><code>-c</code> <code>--copy-wals</code></p>	<p>Specify this option to copy archived WAL files from the BART backup catalog to <code><restore_path>/archived_wals</code> directory.</p> <p>If recovery settings are saved in the <code>postgresql.auto.conf</code> file for point-in-time recovery, the <code>restore_command</code> retrieves the WAL files from <code><restore_path>/archived_wals</code> for the database server archive recovery.</p> <p>If the <code>-c</code> option is omitted and the <code>copy_wals_during_restore</code> parameter in the BART configuration file is not enabled in a manner applicable to this database server, the <code>restore_command</code> in the <code>postgresql.auto.conf</code> file is generated by default to retrieve the archived WAL files directly from the BART backup catalog. See the <i>EDB Postgres Backup and Recovery Installation and Upgrade Guide</i> available at the EDB website for information about the <code>copy_wals_during_restore</code> parameter.</p>
<p><code>--disable-checksum</code></p>	<p>While restoring a backup, specify this option to skip verifying the MD5 or SHA256 checksum files.</p> <p>If you set the <code>--checksum-algorithm=NONE</code> option with the <i>BART scanner</i> or while taking a <i>backup</i>, you must specify the <code>--disable-checksum</code> option while restoring an incremental backup.</p>

3.3.9 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.

Syntax:

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

Note: While invoking the DELETE subcommand, you must specify a specific database server.

For database servers under a retention policy, there are conditions where certain backups may not be deleted. See *Deletions Permitted Under a Retention Policy* for information about permitted backup deletions.

The following table describes the command options:

Options	Description
<pre>-s <server_name> --server <server_name></pre>	<p><server_name> is the name of the database server whose backups are to be deleted.</p>
<pre>-i { all ['']{ <backup_id> <backup_name> },... }[''] } --backupid { all ['']{ <backup_id> <backup_name> },... }[''] }</pre>	<p><backup_id> is the backup identifier of the backup to be deleted and <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.</p> <p>If all is specified, all of the backups and their archived WAL files for the specified database server are deleted.</p>
<pre>-n --dry-run</pre>	<p>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.</p> <p>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.</p>

3.4 Running the BART WAL Scanner

Use the BART WAL scanner to invoke the `bart-scanner` program located in the `BART_HOME/bin` directory. When invoking the WAL scanner, the current user must be the BART user account.

Syntax:

```
bart-scanner
[ -d ]
[ -c <config_file_path> ]
{ -h |
  -v |
  --daemon |
  -p <mbm_file_path> |
  <wal_file_path> |
  RELOAD |
  STOP
  --checksum-algorithm }
```

Note: For clarity, the syntax diagram shows only the single-character option form (for example, `-d`), but the multi-character option form (for example, `--debug`) is supported as well.

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 is added to the `archived_wals` directory specified in the `archive_path` parameter of the configuration file. It then scans the WAL file and produces the MBM file.

The default approach does not work in some cases; for example when the WAL files are shipped to the `archive_path` using the Network File System (NFS) 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 does not scan them (as WAL scanner is not aware of WAL file) 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. The value for `scan_interval` is the number of seconds after which the WAL scanner will initiate force scanning of the new WAL files. See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at the [EDB website](#) for more information about `scan_interval` parameter.

Note: After upgrading to BART 2.6, users who have set this parameter to a non-default value may see increased CPU consumption on the part of `bart-scanner`. If this is an issue, consider increasing the configured value of `scan_interval` parameter, or removing the setting if it is not required.

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

The WAL scanner processes can run in either the foreground or background depending upon usage of the `--daemon` option. Use the `--daemon` option to run the WAL scanner process in the background so that all output messages can be viewed in the BART log file. If the `--daemon` option is omitted, the WAL scanner process runs in the foreground and all output messages can be viewed from the terminal running the program as well as in the BART log file.

See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at the [EDB website](#) for additional information about WAL scanning, `allow_incremental_backups`, and `logfile` parameters.

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 *Basic BART Subcommand Usage* for information about setting the `LD_LIBRARY_PATH` environment variable.

The following table describes the scanner options:

Options	Description
<code>-h</code> <code>--help</code>	Displays general syntax and information on WAL scanner usage.
<code>-v</code> <code>--version</code>	Displays the WAL scanner version information.
<code>-d</code> <code>--debug</code>	Displays debugging output while executing the WAL scanner with any of its options.
<code>-c <config_file_path></code> <code>--config-path</code> <code><config_file_path></code>	Use this option to specify the <code>config_file_path</code> of a BART configuration file if you do not want to use the default BART configuration file path <code>BART_HOME/etc/bart.cfg</code> .
<code>--daemon</code>	Runs the WAL scanner as a background process.
<code>-p <mbm_file_path></code> <code>--print</code> <code><mbm_file_path></code>	Use this option to specify the full directory path to an MBM file whose content is to be printed. The directory specified in the <code>archive_path</code> parameter in the <code>bart.cfg</code> file contains the MBM files.
<code><wal_file_path></code>	Specify the full directory path to a WAL file to be scanned. The directory specified in the <code>archive_path</code> parameter in the <code>bart.cfg</code> file contains the WAL files. Use this option if a WAL file in the archive path is missing its MBM file. This option is to be used for assisting the EnterpriseDB support team for debugging problems that may have been encountered.
RELOAD	Reloads the BART configuration file. The keyword <code>RELOAD</code> is not case-sensitive. The <code>RELOAD</code> 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.

continues on next page

Table 12 – continued from previous page

Options	Description
STOP	Stops the WAL scanner. The keyword STOP is not case-sensitive.
--checksum-algorithm	While invoking the WAL scanner, you can specify one of the following values with the --checksum-algorithm option: --checksum-algorithm=MD5 (default) to generate MD5 checksum files. --checksum-algorithm=SHA256 to generate SHA256 checksum files. --checksum-algorithm=NONE to skip generating checksum files.

Using Tablespaces

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

- You can take full backups with the `BACKUP` subcommand in either tar (`-Ft`) or plain text (`-Fp`) backup file format.
- You must take incremental backups in the plain text (`-Fp`) 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 (`-Ft`) or plain text (`-Fp`) backup file format.

Note: If the particular database cluster you plan to back up contains tablespaces created by the `CREATE TABLESPACE` command, then you must 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. It takes the following format:

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

Where `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/9.6AS/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 later 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.

To view example of backing up and restoring a database cluster on a remote host containing tablespaces, see the *EDB Postgres Backup and Recovery Reference Guide* available at the [EDB website](#).

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.

To view a sample BART managed backup and recovery system consisting of both local and remote database servers, see the *EDB Postgres Backup and Recovery Reference Guide* available at the [EDB website](#).

EDB Postgres™ Backup and Recovery User Guide

Copyright © 2014 - 2020 EnterpriseDB Corporation.

All rights reserved.

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

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

B

BACKUP Subcommand, 33
BACKUP Subcommand Error Messages, 36
BART Management Overview, 15
Basic BART Subcommand Usage, 28
Block-Level Incremental Backup, 8

C

CHECK-CONFIG Subcommand, 30
Concept Overview, 9
Conclusion, 53
Conventions Used in this Guide, 2
Creating a Backup Chain, 14

D

DELETE Subcommand, 47
Deleting obsolete backups, 26
Deletions permitted under retention policy, 24

E

Evaluating obsolete backups, 26

I

Incremental Backup Limitations and Requirements, 8
INIT Subcommand, 31
Introduction, 1

M

MANAGE Subcommand, 41
Managing Backups Using a Retention Policy, 21
Managing Incremental Backups, 26
Managing the Backups Based on the Retention Policy, 24
Marking backups for indefinite keep status, 26
Marking obsolete backups, 26

Marking the Backup Status, 22

O

Overview, 4
Overview - Managing Backups Using a Retention Policy, 21

P

Performing a Restore Operation, 17
Performing an Incremental Backup, 11
Point-In-Time Recovery Operation, 19

R

Recovery window retention policy, 24
Recovery window retention with incremental backups, 27
Redundancy retention policy, 23
Redundancy retention with incremental backups, 27
RESTORE Subcommand, 44
Restoring an Incremental Backup, 12
Restoring incremental backup on BART host, 12
Restoring incremental backup on remote host, 13
Restrictions on pg_basebackup, 3
Running the BART WAL Scanner, 48

S

Setting the Retention Policy, 23
SHOW-BACKUPS Subcommand, 39
SHOW-SERVERS Subcommand, 38

U

Using BART, 15
Using Tablespace, 51

V

VERIFY-CHKSUM Subcommand, 40

W

WAL compression, 41

WAL Scanning - Preparation for an
Incremental Backup, 10

What's New, 2