EDB Backup and Recovery Tool

Version 2.5.3

Built at 2020-12-17T14:53:53

1 EDB Backup and Recovery Tool Installation Guide

Supported Platforms and Database Versions

- To view a complete list of platforms supported by BART, visit the EnterpriseDB website.
- BART supports the following database versions:

  - Advanced Server versions 9.5, 9.6, 10, 11, 12
  - PostgreSQL versions 9.5, 9.6, 10, 11, 12

Software Requirements

The following components are required for BART installation.

- BART Host Components - Use EnterpriseDB packages to add BART host components; see Installing BART using an RPM Package for installation instructions.
- Additional Components - In addition to the BART host components, the following components are required:

  - The Secure Shell (SSH) server daemon and Secure Copy (SCP) client programs <authorizing_ssh/scp_access> must be enabled and activated on the BART host as well as on the remote database server hosts on which BART will be managing backup and recovery.
  - BART uses the `pg_basebackup` utility program when taking full backups.

Limitation

BART supports taking only a full backup of standby servers; it does not support taking incremental or parallel backups of standby servers.

1.1 Installing BART

This section will walk you through performing a fresh installation of BART on a host. Installation instructions are organized into the following platform/installer specific sections:

Using an RPM Package to Install BART on a CentOS or RHEL Host

The following section demonstrates installing BART 2.5.3 on a CentOS 7 host using an RPM package. The tutorial assumes that the user has some knowledge of installation and system administration procedures, and administrative privileges on the host.

1. On a CentOS7 host, the *Extra Packages for Enterprise Linux (EPEL)* package, which contains supporting boost libraries required by BART is installed by default. On a CentOS6/RHEL6 OS, you need to install the EPEL package by assuming the superuser privileges and invoking the following command:

   ```
   yum install -y epel*
   ```

2. Use either yum or rpm to create the repository configuration file.

   ```
   yum install -y https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

   or

   ```
   rpm -Uvh https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

1. Visit this page at the EnterpriseDB website to request credentials to the EDB Yum Repository.

2. Use your choice of editor to open and modify the repository configuration file. The repository configuration file is named `edb.repo` and is located in the `/etc/yum.repos.d` directory.

3. After creating the `edb.repo` file,

   - ensure the value of the enabled parameter is `1`.
   - replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EnterpriseDB user.
   - save the configuration file and exit the editor.

   ```
   [edb]
   name=EnterpriseDB RPMs $releasever - $basearch
   baseurl=https://<username>:<password>@yum.enterprisedb.com/edb/redhat/
   rhel-$releasever-$basearch
   enabled=1
   gpgcheck=1
   gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
   ```

4. Optionally, install the `pg_basebackup` utility program using the server client package. If you do not already have the `pg_basebackup` program installed on the BART host, you can install a

limited number of files that include the pg_basebackup program by invoking the following command:

```
yum install edb-asxx-server-client
```

In the above command, replace xx with the required Advanced Server version. The pg_basebackup version must be the same or more recent than the database server to be backed up. For example, pg_basebackup version 10 can be used to back up database server version 10, but cannot be used to back up database server version 11.

1. Use the following command to install the BART RPM package:

   ```
   yum install edb-bart
   ```

2. Repeat the installation process described in this section to install BART on each remote host on which an incremental backup is to be restored.

3. To verify the BART installation, navigate to the /usr/bart/bin directory and execute the following command:

   ```
   bart --version
   ```

   The bart --version command should return the current BART version. If the bart --version command returns an error stating the PATH is not available after switching from the root user to another BART user account, adjust the setting of the PATH environment variable to include the directory location of the BART bin subdirectory in the ~/.bashrc or ~/.bash_profile files of the following user accounts:

   - The BART user account on the BART host. See Configuring BART for details.
   - The remote user account on the remote host to which incremental backups are to be restored. For details, see the *EDB Postgres Backup and Recovery User Guide* available at the EDB website.

   Upon successful installation, BART is installed in the BART_HOME directory:

   ```
   /usr/edb/bart
   ```

   The installation includes the following files:

File Name	Location	Description
bart	<BART_HOME>/bin	BART command line, executable program
bart-scanner	<BART_HOME>/bin	BART WAL scanner program
bart.cfg.sample	<BART_HOME>/etc	Sample BART configuration file
xlogreader_ident.so	<BART_HOME>/lib	Libraries supporting WAL versions
bart_license.txt	<BART_HOME>	License agreement

After BART is installed successfully, you need to configure the installation.

Installing BART on a Debian or Ubuntu Host

To install BART on a Debian or Ubuntu host, you must have credentials that allow access to the EnterpriseDB repository. To request credentials for the repository, visit this page.

Perform the following steps to install a Debian package using the EnterpriseDB apt repository.

1. Assume the superuser privileges.

   ```
   sudo su -
   ```

2. Configure the EnterpriseDB repository; substitute your EnterpriseDB credentials for the `username` and `password` placeholders in the following command:

   ```
   sh -c 'echo "deb https://username:password@apt.enterprisedb.com/$(lsb_release -cs)-edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -cs).list'
   ```

3. Add support to your system for secure APT repositories.

   ```
   apt-get install apt-transport-https
   ```

4. Add the EBD signing key; When invoking the command, replace the `username` and `password` with the credentials provided by EnterpriseDB.

   > wget -q -O - https://username:password@apt.enterprisedb.com/edb-deb.gpg.key | apt-key add –

5. Update the repository metadata.

   ```
   apt-get update
   ```

6. Install the Debian package.

   ```
   apt-get install edb-bart
   ```

Installing BART on an SLES 12 Host

This section provides instructions for installing BART on an SLES 12 SP4 host using the zypper package manager.

Note

BART is supported on SLES SP4 and SP5 versions.

1. Assume superuser privileges and use the following command to add the EnterpriseDB repository configuration file to your SLES host:

```
zypper addrepo https://zypp.enterprisedb.com/suse/edb-sles.repo
```

This command creates a repository configuration file named `edb.repo` in the `/etc/zypp/repos.d` directory.

1. Use the zypper utility to install BART.

```
zypper install edb-bart
```

1.2 Configuring BART

To configure BART, you must identify the BART user account, configure the BART host, and configure the database server that will be backed up.

Establishing the BART User Account

The BART user account is an operating system user that will run the BART command line program. The BART user account must:

- own the BART backup catalog.
- be able to run the `bart` program and the `bart-scanner` program.
- be able to establish a SSH/SCP connection to and from each database server managed by BART.

You can optionally use the `enterprisedb` database user as the BART user account for an Advanced Server database and use the `postgres` database user as the BART user account for a PostgreSQL server. If you do not wish to use an existing database user as the BART user account, you must create an operating system user to assume the role.

Configuring the BART Host

This section describes the configuration steps that must be performed on the BART host; these steps must be performed as a root user.

Step 1. Navigate to the `usr/edb/bart/etc` directory and make a copy of the `bart.cfg.sample` file to create the `bart.cfg` file that will contain the parameter settings.

Step 2. Confirm that the Postgres `pg_basebackup` utility program is installed on the BART host.

The pg_basebackup utility resides in the bin directory under your Postgres installation.

Step 3. Ensure the LD_LIBRARY_PATH environment variable includes the location of the libpq library. If your libpq library does not reside in the default location (POSTGRES_INSTALL_HOME/lib), you must add the library path to the LD_LIBRARY_PATH environment variable in the BART user account's profile (bash_profile) located in /home/<bart user account>.

```
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
. ~/.bashrc
fi
# User specific environment and startup programs
export LD_LIBRARY_PATH=/usr/edb/as11/lib:$LD_LIBRARY_PATH
```

Step 4. Create the BART backup catalog and ensure the BART user account holds privileges on the BART backup catalog. In the following example, the BART configuration file specifies /opt/backup as the parent directory for the BART backup catalog in the <backup_path> parameter:

```
[BART]

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

In the following example, bartuser is the BART user account. The example creates and sets the ownership and permissions on the BART backup catalog:

```
su root
mkdir /opt/backup
chown bartuser /opt/backup
chgrp bartuser /opt/backup
chmod 700 /opt/backup
```

If the subdirectory does not exist, BART creates a subdirectory for each database server listed in the configuration file when you invoke the bart command line program.

Step 5. Use your choice of editor to open the BART configuration file (located in the usr/edb/bart/etc directory) and edit the configuration as required. You must add the mandatory parameters to the [BART] and [ServerName] sections (for example, [EPAS11]). Default values may be used for optional parameters.

Step 6 Invoke the CHECK-CONFIG subcommand, omitting the -s option to check the parameter settings in the BART configuration file. The CHECK-CONFIG subcommand displays an error

message if the required configuration is not properly set.

For information about server section parameters, see database server parameters.

Configuration Parameter Reference

In the table that follows:

- Parameters set in the [BART] section are applicable to all BART managed database servers.
- Parameters set in the Server section are applicable only to the specific server; the Server parameter setting overrides the [BART] section setting.

Parameter	Type	Default	Server	[BART]
[BART]	Mandatory	N/A	N/A	yes
<bart_host>	Mandatory	N/A	N/A	Yes
<backup_path>	Mandatory	N/A	N/A	Yes
<pg_basebackup_path>	Mandatory	N/A	N/A	Yes
retention_policy	Optional	<max_number>BACKUPS	Yes	Yes
wal_compression	Optional	Disabled	Yes	Yes
copy_wals_during_restore	Optional	Disabled	Yes	Yes
xlog_method	Optional	fetch	Yes	Yes
logfile	Optional	/tmp/bart.log	N/A	Yes
scanner_logfile	Optional	/tmp/bart_scanner.log	N/A	Yes
<bart_socket_directory>	Optional	/tmp	N/A	Yes
<thread_count>	Optional	1	Yes	Yes
<batch_size>	Optional	49152	Yes	Yes
<scan_interval>	Optional	0	Yes	Yes
<mbm_scan_timeout>	Optional	20 seconds	Yes	Yes

Parameter	Type	Default	Server	[BART]
<workers>	Optional	1	Yes	Yes
[Server Name]	Mandatory	N/A	Yes	N/A
<backup_name>	Optional	N/A	Yes	N/A
host	Mandatory	N/A	Yes	N/A
port	Mandatory	5444 for EPAS; 5432 for Postgres	Yes	N/A
user	Mandatory	N/A	Yes	N/A
<archive_path>	Optional	BART backup catalog	Yes	N/A
<archive_command>	Optional	N/A	Yes	N/A
<cluster_owner>	Mandatory	enterprisedb for EPAS postgres for PostgreSQL	Yes	N/A
<remote_host>	Optional	N/A	Yes	N/A
<tablespace_path>	Optional	N/A	Yes	N/A
allow_incremental_backup	Optional	Disabled	Yes	N/A
description	Optional	N/A	Yes	N/A

The following table describes the [BART] host parameters.

Parameters/Placeholder	Description
[BART} (mandatory)	Identifies the global section of the configuration file (it must be named BART).

Parameters/Placeholder	Description
bart_host (mandatory)	Specify the bart user name and the IP address of the bart host on which the BART utility resides, in the form of <bart_user>@<bart_host_address>.
backup_path (mandatory)	Specify the path to the file system parent directory where all BART backups are stored.
pg_basebackup_path (mandatory)	Specify the path to the pg_basebackup program that you installed on the BART host. For information about pg_basebackup version-specific restrictions, see the EDB Postgres Backup and Recovery User Guide.
wal_compression (optional)	Set this parameter to enabled to compress the archived WAL files in gzip format in the BART backup catalog when the MANAGE subcommand is invoked. By default it is set to disabled. The gzip compression program must be in the BART user account's PATH and the WAL compression setting must not be enabled for those database servers where you need to take incremental backups.
copy_wals_during_restore (optional)	Set this parameter to enabled to copy the archived WAL files from the BART backup catalog to the restore_path/archived_wals directory prior to the database server archive recovery. Enabling this option helps you save time during the restore operation. Set this parameter to disabled (default) to retrieve the archived WAL files directly from the BART backup catalog during the database server archive recovery. During the restore operation, recovery settings will be saved in the postgresql.auto.conf file. The restore_command in the postgresql.auto.conf file will be determined by the value specified in the copy_wals_during_restore parameter. If the RESTORE subcommand is invoked with the -c option, the archived WAL files are copied from the BART backup catalog to the restore_path/archived_wals directory, thus overriding any setting of the copy_wals_during_restore parameter. If the RESTORE subcommand is invoked without the -c option, the value specified by the copy_wals_during_restore parameter is used.
xlog_method (optional)	Specify how the transaction log is collected during the execution of pg_basebackup through the BACKUP subcommand. Set xlog_method to fetch (default) to collect the transaction log files after the backup is completed. Set to stream to stream the transaction log in parallel with the full backup creation.

Parameters/Placeholder	Description
retention_policy (optional)	Set this parameter to determine when an active backup should be marked as obsolete when the MANAGE subcommand is used. You can specify the retention policy either in terms of number of backups or duration (days, weeks, or months). <max_number> BACKUPS (default), <max_number> DAYS, <max_number> WEEKS, or <max_number> MONTHS where <max_number> is a positive integer. For information about managing backups using a retention policy, see the EDB Postgres Backup and Recovery User Guide.
logfile (optional)	Use this parameter to specify the path to the BART log file. The default log file location is /tmp/bart.log. The log file will be created the first time you invoke the bart command line program using the sample configuration file value. To change the default setting, you must delete the bart.log file from the /tmp directory and create a new log file in another directory so that a new log file will be created and owned by the new BART user account. If no path to a log file is specified, BART does not create a log file.
scanner_logfile (optional)	Use this parameter to specify the path to the XLOG/WAL scanner log file. The default location is /tmp/bart_scanner.log. The scanner log file will be created the first time you invoke the bart_scanner program using the sample configuration file value. To change the default setting, you must delete the bart_scanner.log file from the /tmp directory and create a new log file in another directory so that a new log file will be created and owned by the new BART user account. If no path to a log file is specified, BART does not create a WAL scanner log file.
<bart_socket_directory> (optional)	Specify the socket directory path where all BART sockets will be stored. The default directory is /tmp. While specifying the bart_socket_directory path, you must ensure that the directory exists and the BART user has the required access permissions to the directory.

Parameters/Placeholder	Description
<thread_count> (optional)	Specify the number of worker threads for copying blocks (for incremental backups) or data files (for full backup) from the database server to the archive_path when the BACKUP subcommand is invoked. The default value is 1. The same set of worker threads are used for the compression operation when taking full backups in order to provide parallel, compressed backups when the BACKUP subcommand is specified with the -z or -c options. The compression operation does not apply to incremental backups. See thread count for more information.
<batch_size> (optional)	Specify the number of blocks of memory used for copying modified blocks from the database server to the archive_path when the BACKUP subcommand is invoked for incremental backups. The default value is 49152 blocks; each block is 8192 bytes. The maximum permitted value is 131072 blocks and the minimum permitted value is 1 block. Reduce the <batch_size> setting if the server runs out of memory while executing the pg_read_binary_file().
<scan_interval> (optional)	Specify the number of seconds after which the WAL scanner should scan the new WAL files. The default value is 0, which means no brute-force scanning will be started.
<mbm_scan_timeout> (optional)	Specify the number of seconds to wait for MBM files before timing out; This parameter is applicable only for incremental backup. The default value is 20 seconds. The mbm_scan_timeout parameter value must be greater than 0. If the value is 0 or negative, then an error will be displayed during an incremental backup.
<workers> (optional)	Specify the number of parallel worker processes required to stream the modified blocks of an incremental backup to the restore host. The default value is 1.

Thread Count

If the BACKUP subcommand is invoked with the --thread-count option, then the number of worker threads specified by this option overrides any setting of the thread_count parameter in the BART configuration file. If the BACKUP subcommand is invoked without the --thread-count option, then the following determines the number of worker threads used:

- The setting of the thread_count parameter in the server section of the BART configuration file overrides the setting of thread_count in the global section for that particular database server.
- If omitted in the server section, the setting of thread_count in the global section is used.

- If the thread_count parameter is not specified in either section, the default is 1.
- When taking a full backup, if the thread count in effect is only 1, then the pg_basebackup utility is used to take the full backup unless the --no-pg_basebackup option is specified with the BACKUP subcommand.

<thread_count> will not be effective if the backup is taken on a standby server.

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

Configuring the Database Server

This section describes the procedure for enabling BART backup and recovery management for a database server. To configure the database server, you need to:

- Authorize SSH/SCP access without a password prompt.
- Create and configure a replication database user.
- Update the BART configuration file (server section) <adding_a_database_server>.
- Enable WAL archiving of the server.
- Verify the server configuration settings.

Note

You must authorize SSH/SCP access and set up a replication database user before restarting the database server with WAL archiving enabled.

Authorizing SSH/SCP Access

BART uses the Secure Shell (ssh) and Secure Copy (scp) Linux utility programs to copy the backup and WAL files from the BART managed database servers to the BART host as well as to restore backups.

- The client/server ssh and scp commands must not prompt for a password when establishing a connection with the target server (the server to which a passwordless connection is being made).
- A passwordless connection uses *authorized public keys* (public key of a client user account) to authenticate with the target server.
- You must add the public key of each client user account to the target user account's authorized public keys list on the target server.

BART Connections that Require Authentication without a Password

For BART usage, there are two scenarios that require a passwordless SSH/SCP connection:

- When connecting from each BART managed database server (SSH/SCP client) to the BART host (target SSH/SCP server) to support WAL archiving as implemented by the archive_command parameter.
  - In this case, the database server user account should generate the public key file

(`id_rsa.pub`) with the `ssh-keygen –t rsa` command on the database server host.
    - The public key file name should be appended to the `~/.ssh/authorized_keys` file on the BART host. The `authorized_keys` file is in the BART user account's home directory.

- When connecting from the BART host (SSH/SCP client) to each BART managed database server (target SSH/SCP server) for taking incremental backups and for supporting restoration of the full backup, the archived WAL files, and the modified blocks, which occurs when the BART `RESTORE` subcommand is given.
    - In this case, the BART user account should generate the public key file (`id_rsa.pub`) with the `ssh-keygen –t rsa` command on the BART host.
    - The public key file name should be appended to the `~/.ssh/authorized_keys` file on the database server host. The `authorized_keys` file is in the home directory of the user account that owns the directory where the database backup is to be restored.

- If backups are to be taken from a given database server host, but restored to a different database server host, the passwordless SSH/SCP connections must be configured from the BART host to the database server host from which the backup is to be taken as well as from the BART host to the database server host to which the backup is to be restored.

See the EDB Postgres Backup and Recovery Reference Guide to view examples of creating a passwordless connection.

Enabling Public Key Authentication

The following example enables SSH/SCP access on a CentOS 6.x host; similar (platform-specific) steps will apply to other platforms/versions.

1. In the SSH server daemon configuration file (`sshd_config`) located in the `/etc/ssh`, set the `PubkeyAuthentication` parameter to `yes`.
2. Reload the configuration file:

```
service sshd reload
```

If you get any SSH or SCP errors, examine the `/var/log/secure` log file.

Creating a Passwordless Connection

The following general instructions will walk you through generating a client's public key file, creating the target server's authorized public keys file, and creating a passwordless connection.

Step 1. On the client system, log in as the user account that will be initiating the SSH or SCP connection.

Step 2. Navigate to the user account's home directory and check for an existing `.ssh` subdirectory. If the `.ssh` directory does not exist, create one and assign the required privileges to the user.

Step 3. Generate the public key file with the following command. Accept all prompted defaults and do not specify a passphrase when prompted for one.

```
ssh-keygen –t rsa
```

The public key file named `id_rsa.pub` is created in the `.ssh` subdirectory.

Step 4. While logged into the client where you just generated the public key file, use `SCP` to make a temporary copy of it on the target server:

```
scp ~/.ssh/id_rsa.pub <target_user>@<host_address>:tmp.pub
```

Step 5. Navigate into the target user account's home directory and check for an existing `.ssh` subdirectory. If it does not exist, create one and assign the required privileges to the user.

Step 6. Append the temporary, client's public key file, `tmp.pub`, to the `authorized_keys` file. If an `authorized keys` file does not exist, create a new file, but do not completely replace any existing `authorized keys` file.

```
cat tmp.pub >> ~/.ssh/authorized_keys
```

Make sure the `authorized_keys` file is only accessible by the file owner and not by groups or other users. If the `authorized_keys` file does not have the required permission setting or it was newly created, change the file permissions as follows:

```
chmod 600 ~/.ssh/authorized_keys
```

Step 7. Delete the temporary public key file:

```
rm tmp.pub
```

Now, when logged into the client system as `user` there should be no prompt for a password when commands such as the following is given:

```
ssh target_user@host_address
```

Setting up a Replication Database User

For each database server that is to be managed by BART, a database user must be chosen to serve as the *replication database user*. The replication database user sets the Postgres `archive_command` configuration parameter when the `INIT` subcommand in invoked and creates backups when the `BACKUP` subcommand is invoked. The replication database user must be a `superuser`.

When executed with the PSQL client, the following PostgreSQL command creates a superuser to be the replication database user:

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

The `pg_hba.conf` file must minimally permit the replication database user to have access to the database.

In the following example, the `pg_hba.conf` file permits the `repuser` (replication database user) to

have access to the `template1` database. The IP address from which `repuser` has access to `template1` database is the location of the BART host:

For pg_basebackup only: If `pg_basebackup` is to be used for taking any backups (such as for standby servers), the replication database user must also be included in the `pg_hba.conf` file as a `replication` database connection as shown by the last entry in the following example.

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

The replication database user must be specified for the `user` parameter in the BART configuration file for the database server as shown in the following example:

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

There must be no password prompt when connecting to the database server with the replication database user. There are several ways to permit this; one recommended method is to use a `.pgpass` file located in the BART user account's home directory.

For example, if `bartuser` is the BART user account, then the `.pgpass` file located in the `/home/bartuser` directory must contain the following entry:

```
192.168.2.24:5444::repuser:password
```

When `bartuser` invokes a BART backup, the password for the replication database user, `repuser`, is obtained from the `.pgpass` file of `bartuser` to connect to the database server running at `192.168.2.24` on `port 5444`.

The `.pgpass` file must contain an entry for each BART managed database server and its corresponding replication database user and password.

Adding a Database Server to the BART Configuration File

To manage the backup and recovery of a database server, you must add entries to the server section of the BART configuration file (located in <BART_HOME>/etc/bart.cfg). Settings in the server section will override the settings in the [BART] section for that particular database server. If omitted, default values will be used.

For each cluster serviced by BART, the following parameters are required:

```
[HR]

host = 192.168.2.24
port = 5432
user = postgres
cluster_owner = postgres
```

Note

The port parameter setting is required only if the database server listens on a port other than the default (for example if Postgres listens on a port other than 5432).

Database Server Parameter Reference

Set the following parameters in the server section of the BART configuration file. The parameter setting in the server section overrides the setting in the global [BART] section for that particular database server. If omitted, the default value will be used.

The following table describes the database server parameters.

Parameters/Placeholder	Description
[ServerName] (mandatory)	Specify the server name that you want to backup using BART. It is not case-sensitive when referenced with BART subcommand options. A lowercase conversion of this name is used to create a subdirectory in the BART backup catalog for storing the backups and WAL files for this database server (for eg., epas12).
<backup_name> (optional)	Specify a template for user-defined, friendly names that will be assigned to the backups of the database server. The maximum permitted length of backup name is 49 characters. The <backup_name> parameter can be overridden by the --backup-name option of the BACKUP subcommand. If this parameter is omitted from the BART configuration file, and the --backup-name option with a user-defined name is not specified with the BACKUP subcommand, then the backup can only be referenced in BART subcommands by the BART assigned, integer backup identifier.

Parameters/Placeholder	Description
host (mandatory)	Specify the IP address of the database server to be configured for backup.
port (mandatory)	Specify the port number identifying the database server instance (that is, the relevant database cluster) to be backed up. The default port number for EPAS is 5444 and for Postgres it is 5432. The port parameter setting is only required if the database server listens on a port other than the default value.
User (mandatory)	Specify the replication database user name used by BART to establish the connection to the database server for full backups. See Setting up a Replication Database User for more information.
<archive_path> (optional)	Specify the path where archived WAL files will be stored. The default location is the BART backup catalog (<backup_path>/<server_name>/archived_wals).
<archive_command> (optional)	When the INIT subcommand is used, the content and variables specified in the BART <archive_command> result in the archive command string to be generated into the Postgres archive_command parameter in the postgresql.auto.conf file. To configure the BART <archive_command> parameter, enclose the command string within single quotes ('). If you do not specify the <archive_command> parameter in the configuration file, the default setting is taken as 'scp %p %h:%a/%f'. See Archive Command Auto Configuration for information about variables. The BART <archive_command> parameter in the BART configuration file, and the Postgres <archive_command> parameter in the postgresql.conf file (or the postgresql.auto.conf file) refer to two different parameters that are to be set in different manners.
<cluster_owner> (required)	Specify the Linux operating system user account that owns the database cluster. This is typically enterprisedb for Advanced Server database clusters installed in the Oracle compatible mode, or postgres for Advanced Server database clusters installed in the PostgreSQL compatible mode and PostgreSQL database clusters.

Parameters/Placeholder	Description
<remote_host> (optional)	Specify the IP address of the remote server to which a backup is to be restored. Specify this parameter in the form of <remote_user>@<remote_host_address>. <remote_user> is the user account on the target database server host that accepts a passwordless SSH/SCP login connection and owns the directory where the backup is to be restored. <remote_host_address> is the IP address of the remote host. For restoring a backup to a remote host or for restoring any backup where <remote_user> and the BART user account are not the same users, either this parameter must be set or it may be specified with the -r option with the BART RESTORE subcommand.
<tablespace_path> (optional)	Specify path to which tablespaces are to be restored in the format OID = <tablespace_path>; If the backup is to be restored to a remote host specified by the <remote_host> parameter, then the tablespace paths must exist on the remote host.
allow_incremental_backups (optional)	Set this parameter to enabled to enable use of the WAL scanner and permit taking incremental backups when the BACKUP subcommand is invoked with the --parent option. Set it to disabled (default) to disallow incremental backups and thus permit only full backups. For information about using the BACKUP subcommand and running the WAL scanner, please see the EDB Postgres Backup and Recovery User Guide available at this page.
Description (optional)	Specify the description that will be used to identify the database server.

For information regarding how to configure the following parameters, see configuring the BART host.

- retention_policy
- xlog_method
- wal_compression
- copy_wals_during_restore.
- thread_count.
- batch_size.
- scan_interval.
- mbm_scan_timeout.
- workers

Backup Name Template

- The template is an alphanumeric string that may include the following variables that will be replaced with the timestamp values when the backup is taken:

  - %year to be replaced by 4-digit year
  - %month to be replaced by 2-digit month
  - %day to be replaced by 2-digit day
  - %hour to be replaced by 2-digit hour
  - %minute to be replaced by 2-digit minute
  - %second to be replaced by 2-digit second

- To include a percent sign (%) as a character in the backup name, specify %% in the template.
- Do not enclose the template string in quotes even if you want the template to include space characters, otherwise the enclosing quotes are stored as part of the backup name. However, when referenced with the -i option by BART subcommands use of space characters in the backup name requires enclosing the backup name in quotes.

The following example shows the configuration settings of three database servers:

```
[ACCTG]

host = 127.0.0.1
port = 5444
user = enterprisedb
cluster_owner = enterprisedb
backup_name = acctg_%year-%month-%dayT%hour:%minute:%second
archive_command = 'cp %p %a/%f'
allow_incremental_backups = enabled
retention_policy = 8 BACKUPS
description = "Accounting"

[MKTG]

host = 192.168.2.24
port = 5444
user = repuser
cluster_owner = enterprisedb
remote_host = enterprisedb@192.168.2.24
allow_incremental_backups = enabled
description = "Marketing"

[HR]

host = 127.0.0.1
port = 5432
user = postgres
cluster_owner = postgres
retention_policy = 4 DAYS
description = "Human Resources"
```

Enabling WAL Archiving

WAL archiving must be enabled for the database server for which BART is to perform backup and recovery management.

- The WAL Archiving Configuration section describes the manual WAL archiving configuration process.
- The Archive Command Auto Configuration section describes an automated WAL archiving process.

WAL Archiving Configuration

Set the following configuration parameters in the `postgresql.conf` file to enable WAL archiving

- Set `wal_level` to `archive` for Postgres 9.5 or to `replica` for Postgres 9.6 or later.
- Set `archive_mode` to `on`.
- Set the PostgreSQL `archive_command` parameter to copy the WAL files to the `archive_path`. The `archive_command` configuration parameter mentioned here is located in the `postgresql.conf` file; the PostgreSQL `archive_command` parameter is used in a different manner than the BART archive_command <archive_command>.
- Set `max_wal_senders` to a value high enough to leave at least one session available for the backup. If the `xlog_method=stream` parameter setting is to be used by this database server, the `max_wal_senders` setting must account for an additional session for the transaction log streaming (the setting must be a minimum of 2). See Configuring the BART host for information about the `xlog_method` parameter.

For detailed information about WAL archiving, see the PostgreSQL Core Documentation.

The `ARCHIVE PATH` field displayed by the BART `SHOW-SERVERS` subcommand displays the full directory path where the WAL files should be copied as specified in the `archive_command` configuration parameter in the `postgresql.conf` file:

```
-bash-4.1$ bart SHOW-SERVERS -s acctg
SERVER NAME : acctg
HOST NAME : 192.168.2.24
USER NAME : repuser
PORT : 5444
REMOTE HOST :
RETENTION POLICY : none
DISK UTILIZATION : 0.00 bytes
NUMBER OF ARCHIVES : 0
ARCHIVE PATH : /opt/backup/acctg/archived_wals
ARCHIVE COMMAND : (disabled)
XLOG METHOD : fetch
WAL COMPRESSION : disabled
TABLESPACE PATH(s) :
INCREMENTAL BACKUP : DISABLED
DESCRIPTION : "Accounting"
```

The parameter settings in the following example will copy the WAL files to a directory named `/opt/backup/acctg/archived_wals` on the BART host located at `192.168.2.22` as the `bartuser` user account. Using the `bartuser` account ensures that the operation will have sufficient permissions to copy to the BART backup catalog owned by `bartuser`.

```
wal_level = archive                     # minimal, archive, or hot_standby
                                # (change requires restart)
...

archive_mode = on                       # allows archiving to be done
                                # (change requires restart)
archive_command = 'scp %p bartuser@192.168.2.22:/opt/backup/acctg/archived_wals/%f'
                                # command to use to archive a logfile segment
                                # placeholders: %p = path of file to archive
                                # %f = file name only
...

max_wal_senders = 1                     # max number of walsender processes
                                # (change requires restart)
```

The database server must be restarted in order to initiate WAL archiving, but do not do so until you have verified that the full path of the BART backup catalog has been created by some prior BART subcommand or the archive operation will fail.

Start the WAL scanner by executing the following command:

```
./bart-scanner
```

Archive Command Auto Configuration

To enable WAL archiving:

- In the `postgresql.conf` file, set the `wal_level` to `archive` for Postgres 9.5 or to `replica` for Postgres 9.6 or later, set the `archive_mode` to `on`, and set `max_wal_senders` to a value high enough to leave at least one session available for the backup. If the `xlog_method=stream` parameter setting is to be used by this database server as determined in the BART configuration file, the `max_wal_senders` setting must account for an additional session for the transaction log streaming (that is, the setting must be a minimum of `2`). See Configuring the BART host for information on the `xlog_method` parameter.

- Configure the Postgres `archive_command` parameter automatically with the `INIT` subcommand and restart the database server when you are ready to initiate WAL archiving. The `INIT` subcommand invokes the Postgres `ALTER SYSTEM` command to set the Postgres `archive_command` configuration parameter in the `postgresql.auto.conf` file located in the managed database server's `POSTGRES_INSTALL_HOME data directory`. For additional information about the `INIT` subcommand, see the EDB Postgres Backup and Recovery User Guide.

  The archive command string that the `INIT` subcommand generates into the

postgresql.auto.conf file is determined by the parameter setting of the BART archive_command parameter in the server section of the BART configuration file. If the BART archive_command parameter is not set in the server section for a given database server, the command string that is configured uses the following default format:

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

The following table describes these variables:

Variable	Description
%p	The path of the file to archive used by the Postgres archiving process.
%h	Will be replaced by the <bart_user>@<bart_host_address> as specified in the <bart_host> parameter setting.
%a	Will be replaced by the BART archived_wals directory as specified in the archive path parameter setting. If the <archive_path> is not specified, then the default directory is <backup_path>/<server_name>/archived_wals. <server_name> is the lowercase conversion of the database server name.
%f	The archived file name used by the Postgres archiving process.

The placeholders %h and %a are replaced by the INIT subcommand when creating the archive command string. The placeholders %p and %f are not replaced by the INIT subcommand, but are kept as given to be used by the Postgres archiving process.

For example, to use the default archive command format, the BART configuration file contains the following settings where the BART archive_command parameter is omitted from the server section for ACCTG:

```
[BART]

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

[ACCTG]

host = 127.0.0.1
port = 5444
user = repuser
cluster_owner = enterprisedb
description = "Accounting"
```

The INIT subcommand is invoked by BART user account bartuser as follows:

```
[bartuser@localhost ~]$ bart INIT -s acctg -o
INFO: setting archive_command for server 'acctg'
WARNING: archive_command is set. server restart is required
```

If the BART backup catalog directory is not already complete, it will be completed.

The resulting archive command string in the `postgresql.auto.conf` file located in the managed database server's `POSTGRES_INSTALL_HOME/data directory` appears as follows:

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

Run the `INIT` subcommand with the `-o` option to override any existing `archive_command` setting in the `postgresql.conf` or the `postgresql.auto.conf` file. In addition, the `-o` option must be used to generate the command string if the `archive_mode` is set to off even if there are no existing settings of the `archive_command` in the `postgresql.conf` or `postgresql.auto.conf` files.

In this example, the following BART configuration file is used with an explicit setting of the BART `archive_command` parameter:

```
[BART]

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

[ACCTG]

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

The `INIT` subcommand is invoked by BART user account `enterprisedb` as follows:

```
-bash-4.1$ bart INIT -s acctg -o
INFO: setting archive_command for server 'acctg'
WARNING: archive_command is set. server restart is required
```

The resulting Postgres `archive_command` parameter in the `postgresql.auto.conf` file appears as follows:

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

When the database server has been restarted, the ARCHIVE COMMAND field of the SHOW-SERVERS subcommand displays the active Postgres archive command as shown by the following example:

```
-bash-4.1$ bart SHOW-SERVERS -s acctg
SERVER NAME : acctg
HOST NAME : 127.0.0.1
USER NAME : repuser
PORT : 5444
REMOTE HOST :
RETENTION POLICY : none
DISK UTILIZATION : 48.00 MB
NUMBER OF ARCHIVES : 0
ARCHIVE PATH : /opt/backup/acctg/archived_wals
ARCHIVE SCOMMAND : `cp %p /opt/backup/acctg/archived_wals/%f`
XLOG METHOD : fetch
WAL COMPRESSION : disabled
TABLESPACE PATH(s) :
INCREMENTAL BACKUP : DISABLED
DESCRIPTION : "Accounting"
```

Verifying Configuration Settings

To verify the parameter settings of the database server specified, execute tthe CHECK-CONFIG subcommand with the –s option:

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

The CHECK-CONFIG subcommand confirms the following:

- The cluster_owner parameter is set to the user account owning the database cluster directory.
- A passwordless SSH/SCP connection is set between the BART user and the user account specified by the cluster_owner parameter.
- The BART user parameter specifies a database superuser.
- The BART user has access to the backup directory catalog.
- The pg_hba.conf file contains a replication entry for the database superuser specified by the BART user parameter.
- The archive_mode parameter in the postgresql.conf file is enabled.
- The archive_command parameter in the postgresql.auto.conf or the postgresql.conf file is set.
- The allow_incremental_backups parameter in the BART configuration file is enabled for database servers for which incremental backups are to be taken.
- Archiving of WAL files to the archive_path is in process.

- The WAL scanner program is running.

After configuring the BART host and the database server(s), you can start using BART. For information about using BART, see the EDB Postgres Backup and Recovery User Guide.

1.3 Installation Troubleshooting

This section provides a workaround for the following installation issue that you may encounter:

> existing lock /var/run/yum.pid: another copy is running as pid 3104. Another app is currently holding the yum lock; waiting for it to exit

To fix this issue, execute the following command:

```
rm -f /var/run/yum.pid
```

1.4 Performing a BART Upgrade

This section outlines the process of upgrading BART from an existing version to the latest version.

- Upgrading from BART 2.0 describes the upgrade process from BART 2.0 to the latest version.
- Upgrading from Older Versions of BART (except 2.0) <upgrading_from_older_versions_(except_2.0)_to_latest_versions_of_bart> describes the upgrade process from previous BART versions (except 2.0) to the latest version.

Upgrade Restrictions

The following restrictions apply with regard to previous BART versions.

- You can take incremental backups using the latest version only when the parent backup (full or incremental backup) has also been taken with the latest version.
- Using the latest version, you can restore incremental backups that are taken only with the latest version of BART. However, using the latest version you can restore full backups that were taken with older versions.

Upgrading from Older Versions of BART (except 2.0)

Perform the following steps to upgrade from older versions of BART (except 2.0) to the latest

version:

Step 1: Assume the identity of the BART user account and invoke the following command to stop the BART WAL scanner program (`bart-scanner`):

```
bart-scanner STOP
```

Step 2: As the `root` user, upgrade to the latest BART version with the `yum upgrade` command.

- To upgrade the BART RPM package directly from the *EDB Yum Repository* website, specify only the package name:

  ```
  yum upgrade edb-bart
  ```

  You can also use a downloaded RPM package file to upgrade. To use a downloaded BART RPM package file to upgrade, use the `yum` command, specifying the complete RPM package file name:

  ```
  yum upgrade edb-bart-2.5.3 rhel7.x86_64.rpm
  ```

Step 3: Repeat the process described in this section to upgrade to BART 2.5.3 on each remote hosts where an incremental backup will be restored.

For additional information about restoration of incremental backups on remote hosts, see the *EDB Postgres Backup and Recovery User Guide* available at this page.

Step 4: If the `bart --version` command returns an error stating the `PATH` is not available after switching from `root` user to another BART user account, adjust the setting of the `PATH` environment variable to include the location of the BART 2.5.3 executable (the `bin` subdirectory) in the `~/.bashrc` or `~/.bash_profile` files of the following user accounts:

- The BART user account on the BART host.
- The remote user account on the remote host to which incremental backups are to be restored. For details, see the *EDB Postgres Backup and Recovery User Guide* available at this page.

The `PATH` setting should be the same as set for BART 2.5.3 since all versions use `/usr/edb/bart/bin`.

Note

After upgrading to BART 2.5.3, you must take a new full backup of your system before performing an incremental backup.

Upgrading from BART 2.0

Perform the following steps to upgrade BART 2.0 to the latest version of BART:

Step 1: Install the latest version of BART. For information about how to install, see Using an RPM Package to Install BART.

Step 2: Save a copy of your BART 2.0 configuration file. The default location of the BART 2.0 configuration file is `/usr/edb/bart2.0/etc/bart.cfg`.

Step 3: Invoke the following command to remove BART 2.0:

```
yum remove edb-bart20
```

Step 4: Place the BART 2.0 configuration file (`bart.cfg`) that you saved in Step 2 in the newly created `/usr/edb/bart/etc` directory. You can use many of the same configuration parameters for BART 2.5.3, but note that you must use a new directory for the BART backup catalog. A new set of full backups and incremental backups taken using BART 2.5.3 must be stored in a new BART backup catalog.

To specify an alternative configuration file name or location, use the `-c` option with BART subcommands. For more information about the `-c` option, see the EDB Postgres Backup and Recovery User Guide available at this page.

Note

The `bart.cfg` configuration file is only required on the BART 2.5.3 host from which you will invoke BART subcommands. BART does not require the `bart.cfg` file on hosts on which an incremental backup will be restored.

Step 5: Adjust the setting of the `PATH` environment variable to include the location of the BART 2.5.3 executable (the `bin` subdirectory) in the `~/.bashrc` or `~/.bash_profile` files for the following user accounts:

- The BART user account on the BART host.
- The user account on the remote host to which incremental backups will be restored. For details, see the *EDB Postgres Backup and Recovery User Guide* available at this page.

Step 6: Perform the BART 2.5.3 installation and BART 2.0 removal process on each remote host on which an incremental backup was restored using BART 2.0.

Note

After upgrading to BART 2.5.3, you must take a new full backup of your system before performing an incremental backup.

1.5 Uninstallation

To uninstall BART, assume the identity of the `root` user and invoke the following command:

```
yum remove edb-bart
```

Uninstalling BART does not delete the backup files and archived WAL files that reside in the BART backup catalog. To permanently delete the backup files and archived WAL files in the BART backup catalog (`/opt/backup`), use one of the follwing commands:

- `rm –rf /opt/backup`
- BART `DELETE` subcommand

For information about the BART `DELETE` subcommand, refer the EDB Postgres Backup and Recovery User Guide available at this page.

2 Backup and Recovery Quick Start Guide

This tutorial demonstrates installing Backup and Recovery Tool (BART) 2.5.3 using yum on a CentOS 7 host. The tutorial assumes that the user has some knowledge of installation and system administration procedures, and administrative privileges on the host. It provides shortcuts that allow you to install and configure BART with minimal configuration settings.

For detailed information about BART installation and configuration, see the BART Installation and Upgrade Guide.

- BART is tested with the following database versions:
  - Advanced Server - 9.5, 9.6, 10, 11, and 12.
  - PostgreSQL - 9.5, 9.6, 10, 11, and 12.

Installing BART

The following steps describe installing BART on CentOS 7.x OS.

1. Assume superuser privileges and use `yum` to create the repository configuration file:

   ```
   yum install -y https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

2. To request credentials to the EDB repository, visit the EnterpriseDB website.

3. Use your choice of editor to open the repository configuration file (named `edb.repo`, located in `/etc/yum.repos.d`) and set the value of the `enabled` parameter to `1`, and replace the `username` and `password` placeholders in the `baseurl` specification with the username and password of a registered EnterpriseDB user.

4. Then, install a Postgres (Advanced Server or PostgreSQL) database server. Initialize the cluster and start the server. Note that the BART host server is not required to host a Postgres installation, but must include a copy of the Postgres `libpq` library, the `pg_basebackup` utility program, and Boost Libraries version 1.48 and 1.53 (for RHEL/CentOS 6 and RHEL/CentOS 7 only).

5. Optionally, if you do not already have the pg_basebackup program installed on the BART host, you can install a limited number of files that include the pg_basebackup program using the following command:

```
yum install edb-asxx-server-client
```

6. Then, use the following command to install the BART RPM package:

```
yum install edb-bart
```

BART (bart program and bart-scanner) is installed in the /usr/edb/bart directory, referred to as <BART_HOME>. Repeat the installation process described in this section to install BART x.x on all remote hosts where incremental backups are to be restored.

Configuring BART

Before configuring BART, establish the BART user account (the operating system user) that will run the BART command line program.

To configure the BART host and each database server that is to be managed by BART, perform the following steps:

1. Assume superuser privileges, create the directory that will hold the BART backup catalog, and assign its ownership (with restrictive privileges) to the BART user account:

```
su root
mkdir /opt/backup
chown bartuser /opt/backup
chgrp bartuser /opt/backup
chmod 700 /opt/backup
```

In the example, bartuser is the BART user account and /opt/backup is the BART backup catalog.

1. Navigate to the /usr/edb/bart/etc directory and copy the bart.cfg.sample file to create the bart configuration file (bart.cfg):

```
cp bart.cfg.sample bart.cfg
```

1. Open the BART configuration file (bart.cfg) using an editor of your choice. Scroll through the BART configuration file and edit the sections as required; example settings are included for your reference. You must add the mandatory parameters to both the sections as described in the following table. Default values may be used for optional parameters; for detailed information about parameter settings, see the BART Installation and Upgrade Guide.

   Parameters set in the [BART] section are applicable to all BART managed database servers, and the parameters set in the [ServerName] section are applicable only to the specific server; the [ServerName] setting overrides the [BART] section setting.

Parameters/Placeholder	Section	Description

Parameters/Placeholder	Section	Description
bart_host	[BART]	Use this field to specify the BART user and the IP address of the host on which the BART utility is installed. Specify the value in the form of <bart_user>@<bart_host_address>
backup_path	[BART]	Use this field to specify the path where all BART backups and archived WAL files will be stored. Ensure the BART user account holds privileges to create subdirectories and files within the location specified in the backup_path parameter. The default backup_path is BART backup catalog (/opt/backup)
pg_basebackup_path	[BART]	Use this field to specify the path to the pg_basebackup utility (/usr/edb/as<xx>/bin/pg_basebackup)
[ServerName]	[ServerName]	Specify the name of the database server to be backed up (for example, [EPAS12]).
host	[ServerName]	Specify the IP address of the database server to be configured for backup.
port	[ServerName]	Specify the port number identifying the database server instance to be backed up. The default port number for EPAS is 5444 and for Postgres it is 5432. The port parameter setting is only required if the database server listens on a port other than the default value.
user	[ServerName]	Specify the replication database user name used by BART to establish the connection to the database server for full backups.
cluster_owner	[ServerName]	Specify the Linux operating system user account that owns the database cluster.

In the following example, only mandatory parameters are set:

```
[BART]
bart_host= bartuser@192.168.169.199
backup_path = /opt/backup
pg_basebackup_path = /usr/edb/as12/bin/pg_basebackup

[EPAS12]
host = 127.0.0.1
user = repuser
cluster_owner = enterprisedb
```

1. As the BART user, navigate to the /usr/edb/bart/bin directory and invoke the following subcommand (omitting the -s option) to verify the [BART] section parameter settings in the BART configuration file:

```
bart CHECK-CONFIG
```

2. Then, authorize SSH/SCP access <passwordless> between the server and the BART host without a password prompt.

3. Create a replication database user for each database server that BART manages.

4. To enable continuous WAL archiving for any database server for which BART is to perform a backup, in the postgresql.conf file set the wal_level to archive for Postgres 9.5 or to replica for Postgres 9.6 or later, set the archive_mode to on, set the archive_command (if it is not set in the bart.cfg file), and set the max_wal_senders to a value high enough to leave at least one session available for the backup, and restart the database server.

5. To start the WAL scanner, navigate to the /usr/edb/bart/bin directory as a BART user and execute the following command:

   ```
   ./bart-scanner
   ```

6. If you are using the default archive_command, then navigate to the /usr/edb/bart directory as a BART user, run the INIT subcommand without the -o option, and restart the database server:

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

   Where <server_name> is the name of the database server to be backed up.

   If you have customized the archive_command setting in the bart.cfg file, run the INIT subcommand with the -o option to override any existing Postgresql archive_command setting in the postgresql.conf or the postgresql.auto.conf file, and restart the database server.

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

7. To verify the parameter settings of the database server, as a BART user navigate to the /usr/edb/bart/bin directory and invoke the CHECK-CONFIG subcommand with the -s option:

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

BART is now configured successfully. For detailed information about using BART, see the EDB Backup and Recovery Tool User Guide.

Creating a Passwordless Connection

The following example enables SSH/SCP access on a CentOS 7.x host; similar (platform-specific) steps will apply to other platforms/versions. You must create a passwordless connection between the BART host (SSH/SCP client) and the database server (target SSH/SCP server), as well as a passwordless connection between the database server (SSH/SCP client) and the BART host (target SSH/SCP server).

1. Log in as the user account on the BART host that will be initiating the SSH or SCP connection and navigate to the user account's home directory and check for an existing .ssh subdirectory. If the .ssh directory does not exist, create one with the required privileges.

2. As a root user navigate to the `/usr/edb/bart` directory, open the `/etc/ssh/sshd_config` file and set the `PubkeyAuthentication` parameter to `yes`.

3. Reload the configuration file:

```
service sshd reload
```

If you get any SSH or SCP errors, examine the log file (`/var/log/secure`).

1. As a BART user, use the following command to generate the public key file; you can accept the default responses:

```
ssh-keygen -t rsa
```

The public key file named `id_rsa.pub` is created in the `.ssh` subdirectory.

2. Use `SCP` to make a temporary copy of the public key file on the target server:

```
scp ~/.ssh/id_rsa.pub target_user@host_address:tmp.pub
```

3. As a `target_user`, log into the target server using `ssh target_user@host_address` command and navigate to the user account's home directory to check if there is an existing `.ssh` subdirectory. If it does not exist, create one with the required privileges.

4. Append the temporary client's public key file, `tmp.pub`, to the authorized keys file named `authorized_keys`:

```
cat tmp.pub >> ~/.ssh/authorized_keys
```

If an authorized keys file does not exist, create a new file, but be careful not completely replace any existing authorized keys file.

5. Ensure the `authorized_keys` file is only accessible by the file owner, and not by groups or other users:

```
chmod 600 ~/.ssh/authorized_keys
```

6. Delete the temporary public key file:

```
rm tmp.pub
```

Now, when logged into the BART host as a user, there should be no prompt for a password when you are connecting to the target database server:

```
ssh target_user@database_server_address
```

Creating a Passwordless Connection Between the Database Server and the BART Host

1. On the database server, navigate into the target user account's home directory to check for an

existing `.ssh` subdirectory. If it does does not exist, create one in the user account's home directory with the required privileges.

2. As a database server user, generate the public key file:

```
ssh-keygen -t rsa
```

3. Create a temporary copy of the public key file:

```
scp ~/.ssh/id_rsa.pub target_user@host_address:tmp.pub
```

4. As a target user, log into the BART host and navigate to the user account's home directory to check if there is an existing `.ssh` subdirectory. If it does not exist, create one with the required privileges:

```
ssh target_user@host_address
```

5. Append the temporary, client's public key file to the `authorized_keys` file:

```
cat tmp.pub >> ~/.ssh/authorized_keys
```

If an authorized keys file does not exist, create a new file, but do not completely replace any existing authorized keys file.

1. Ensure the `authorized_keys` file is only accessible by the file owner and not by groups or other users (`chmod 600`):

```
chmod 600 ~/.ssh/authorized_keys
```

2. Delete the temporary public key file:

```
rm tmp.pub
```

Now, when logged into the database server as a user, there should be no prompt for a password when you are connecting to the BART host:

```
ssh bart_user@bartip_address
```

- If backups are to be taken from a given database server host, but restored to a different database server host, the passwordless SSH/SCP connections must be configured from the BART host to the database server host from which the backup is to be taken as well as from the BART host to the database server host to which the backup is to be restored.

> For examples of creating a passwordless connection, see the EDB Postgres Backup and Recovery Reference Guide

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

Creating a Replication Database User

1. To create a replication database user (a superuser), connect to the database server with the psql client, and invoke the following PostgreSQL command:

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

2. Specify this replication database user in the `user` parameter of the `bart.cfg` file.

3. The pg_hba.conf file must minimally permit the replication database user to have access to the database. The IP address from which the replication database user has access to the database is the BART host location. The replication database user must also be included in the `pg_hba.conf` file as a replication database connection if `pg_basebackup` is to be used for taking any backups.

4. To ensure there is no password prompt when connecting to the database server with the replication database user, a recommended method is to use the `.pgpass` file located in the BART user account's home directory (if it does not exist, you need to create the `.pgpass` file with the required privileges). The `.pgpass` file must contain an entry for each BART managed database server, and its corresponding replication database user and password.

   The following is an example of an entry in the `.pgpass` file (192.168.2.24 is the IP address of the database server):

   ```
   192.168.2.24:5444::repuser:password
   ```

3 Backup and Recovery Reference Guide

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

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

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

The document is organized as follows:

- See Subcommands for information and examples related to BART subcommands.

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

3.1 BART Subcommand Syntax and Examples

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

Invoking BART

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

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

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

To run BART from any current working directory:

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

Syntax for invoking BART

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

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

General Options

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

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

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

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

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

Usage:
bart BACKUP [OPTION]...

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

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

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

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

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

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

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

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

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

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

3.1.1 BACKUP

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

Syntax for a Full Backup:

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

[ -z ] [ –c <compression_level> ]

[ --backup-name <backup_name> ]

[ --thread-count <number_of_threads> ]

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

Syntax for an Incremental Backup:

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

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

[ --backup-name <backup_name> ]

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

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

The following table describes the BACKUP options:

Options	Description
-s or --server { <server_name> | all }	Use this option to specify the database server to be backed up. Specify <server_name> to take a backup of the database server (as specified in the BART configuration file). Specify all to take a backup of all servers.
-F or --format { p | t }	Use this option to specify the backup file format. Specify p option to take backup in plain text format and specify t option to take backup in tar format. If the p or t option is omitted, the default is tar format. Use p option with the BACKUP subcommand when streaming is used as a backup method. An incremental backup can only be taken in plain text format (p).
-z or --gzip (applicable only for full backup and tar format)	Use this option to enable gzip compression of tar files using the default compression level (typically 6).
-c or --compress-level <compression_level> (applicable only for full backup and tar format)	Use this option to specify the gzip compression level on the tar file output. <compression_level> is a digit from 1 through 9, with 9 being the best compression.
--backup-name <backup_name>	Use this option to assign a user-defined, alphanumeric friendly name to the backup. The maximum permitted length of backup name is 49 characters. For detailed information about this parameter, see the EDB Postgres Backup and Recovery User Guide. If the option --backup-name is not specified and the backup_name parameter is not set for this database server in the BART configuration file, then the backup can only be referenced in other BART subcommands by the BART assigned backup identifier.
--thread-count <number_of_threads>	Use this option to specify the number of worker threads to run in parallel to copy blocks for a backup. For detailed information about the --thread-count parameter, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide.

Options	Description
`--with-pg_basebackup` (applicable only for full backup)	Use this option to specify the use of `pg_basebackup` to take a full backup. The number of thread counts in effect is ignored as given by the `thread_count` parameter in the BART configuration file. When taking a full backup, if the thread count in effect is greater than `1`, then the `pg_basebackup` utility is not used to take the full backup (parallel worker threads are used) unless the `--with-pg_basebackup` option is specified with the `BACKUP` subcommand.
`--no-pg_basebackup` (applicable only for full backup)	Use this option to specify that `pg_basebackup` is not to be used to take a full backup. When taking a full backup, if the thread count in effect is only `1`, then the `pg_basebackup` utility is used to take the full backup unless the `--no-pg_basebackup` option is specified with the `BACKUP` subcommand.
`--parent { <backup_id> | <backup_name> }`	Use this option to take an incremental backup. The parent backup is a backup taken prior to the incremental backup; it can be either a full backup or an incremental backup. `<backup_id>` is the backup identifier of a parent backup and `<backup_name>` is the user-defined alphanumeric name of a parent backup.
`--check` (applicable only for incremental backup)	Use this option to verify if the required MBM files are present in the BART backup catalog before taking an incremental backup. However, an actual incremental backup is not taken when the `--check` option is specified. The `--parent` option must be used along with the `--check` option.

Examples

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

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

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

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

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

```
[edb@localhost bin]$ ./bart BACKUP -s hr -z
INFO:  DebugTarget - getVar(checkDiskSpace.bytesAvailable)
```

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

START WAL LOCATION: 000000010000000000000025
STOP WAL LOCATION: 000000010000000000000026
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2019-09-04 06:11:49 EDT
STOP TIME: 2019-09-04 06:11:53 EDT
TOTAL DURATION: 4 sec(s)
```

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

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

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

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

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

```
[edb@localhost bin]$ ./bart BACKUP -s acctg --backup-name acctg_%year-%month-%day
INFO: DebugTarget - getVar(checkDiskSpace.bytesAvailable)
INFO: new backup identifier generated 1566899004804
INFO: creating 5 harvester threads
NOTICE: pg_stop_backup complete, all required WAL segments have been archived
INFO: backup completed successfully
INFO:
BART VERSION: 2.5
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1566899004804
BACKUP NAME: acctg_2019-08-27
BACKUP PARENT: none
BACKUP LOCATION: /home/edb/bkup_new/acctg/1566899004804
BACKUP SIZE: 46.86 MB
```

```
BACKUP FORMAT: tar
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 0
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000001A
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2019-08-27 05:43:24 EDT
STOP TIME: 2019-08-27 05:43:24 EDT
TOTAL DURATION: 0 sec(s)
```

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

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

```
TOTAL DURATION: 1 sec(s)
```

3.1.2 CHECK-CONFIG

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

Syntax:

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

```
bart CHECK-CONFIG
```

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

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

The following table describes the `CHECK-CONFIG` option:

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

Example

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

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

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

```
[edb@localhost bin]$ ./bart check-config -s hr
INFO: Checking server hr
INFO: Verifying cluster_owner and ssh/scp connectivity
INFO: success
INFO: Verifying user, host, and replication connectivity
```

```
INFO: success
INFO: Verifying that user is a database superuser
INFO: success
INFO: Verifying that cluster_owner can read cluster data files
INFO: success
INFO: Verifying that you have permission to write to vault
INFO: success
INFO: /home/edb/bkup_new/hr
INFO: Verifying database server configuration
INFO: success
INFO: Verifying that WAL archiving is working
INFO: waiting 30 seconds for
/home/edb/bkup_new/hr/archived_wals/000000010000000000000001E
INFO: success
INFO: Verifying that bart-scanner is configured and running
INFO: success
```

3.1.3 DELETE

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

Syntax:

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

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

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

The following table describes the `DELETE` options:

Options	Description
`-s` (or `--server`) `<server_name>`	`<server_name>` is the name of the database server whose backups are to be deleted.

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

Example

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

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

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

```
[edb@localhost acctg]$ ls -l
total 16
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:03 1566900199604
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:03 1566900204377
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:03 1566900209087
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:05 1566900321228
drwxrwxr-x. 2 edb edb 6 Aug 27 06:01 archived_wals
```

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

```
[edb@localhost bin]$ ./bart DELETE -s acctg -i `1566988095633,1566988100760,
acctg_2019-08-28`
INFO: deleting backup `1566988095633` of server `acctg`
INFO: deleting backup `1566988095633`
```

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

Deleting Multiple Backups with Space Characters

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

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

```
`/home/edb/bkup_new/acctg/archived_wals/000000010000000000000028` is required,
yet not available in archived_wals directory
INFO: backup(s) deleted
INFO: deleting backup `1566900204377` of server `acctg`
INFO: deleting backup `1566900204377`
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or will
be marked unused
WARNING: not marking any WALs as unused WALs, the WAL file
`/home/edb/bkup_new/acctg/archived_wals/00000001000000000000002A` is required,
yet not available in archived_wals directory
INFO: backup(s) deleted
INFO: deleting backup `1566900209087` of server `acctg`
INFO: deleting backup `1566900209087`
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or will
be marked unused
WARNING: not marking any WALs as unused WALs, the WAL file
`/home/edb/bkup_new/acctg/archived_wals/00000001000000000000002C` is required,
yet not available in archived_wals directory
INFO: backup(s) deleted
[edb@localhost bin]$
[edb@localhost bin]$
[edb@localhost acctg]$ ls -l
total 4
drwxrwxr-x. 3 edb edb 4096 Aug 27 06:05 1566900321228
drwxrwxr-x. 2 edb edb 6 Aug 27 06:01 archived_wals
[edb@localhost acctg]$
```

3.1.4 INIT

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

Syntax:

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

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

[-- no-configure]
```

The following table describes the INIT options:

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

Examples

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

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

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

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

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

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

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

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

```
# It will be overwritten by the ALTER SYSTEM command.
archive_mode = 'on'
archive_command = 'scp %p
edb@127.0.0.1:/home/edb/bkup/ppas11/archived_wals/%f'
```

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

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

```
[BART]

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

[ACCTG]

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

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

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

(1 row)
```

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

```
-bash-4.1$ bart INIT -s acctg -o
```

```
INFO: setting archive_mode/archive_command for server 'acctg'
WARNING: archive_command is set. Reload the configuration in the
PostgreSQL server using pg_reload_conf() or 'pg_ctl reload'
```

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

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

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

```
edb=# SHOW archive_command;
              archive_command
---------------------------------------------
cp %p /opt/backup_edb/acctg/archived_wals/%f
(1 row)
```

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

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

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

Note

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3.1.5 MANAGE

The `MANAGE` subcommand can be invoked to:

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

Syntax:

> bart MANAGE [–s { <server_name> | all}]

> [-l] [-d]

```
[ -c { keep | nokeep }
-i { <backup_id> | <backup_name> | all } ]
[ -n ]
```

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

The following table describes the `MANAGE` options:

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

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

Example

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

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

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

```
-bash-4.2$ bart MANAGE -s acctg
INFO: processing server 'acctg', backup '1482770807519'
INFO: processing server 'acctg', backup '1482770803000'
INFO: marking backup '1482770803000' as obsolete
INFO: 1 WAL file(s) marked obsolete
INFO: processing server 'acctg', backup '1482770735155'
INFO: marking backup '1482770735155' as obsolete
```

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

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

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

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

```
 -bash-4.2$ bart MANAGE -s acctg -d
```

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

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

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

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

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

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

```
[ACCTG]

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

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

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

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

```
-bash-4.2$ pwd
/opt/backup/acctg
-bash-4.2$ ls -l archived_wals
total 160
```

```
-rw------- 1 enterprisedb enterprisedb 27089 Dec 26 12:16
000000010000000010000005B.gz
-rw------- 1 enterprisedb enterprisedb 305 Dec 26 12:17
000000010000000010000005C.00000028.backup
-rw------- 1 enterprisedb enterprisedb 27112 Dec 26 12:17
000000010000000010000005C.gz
-rw------- 1 enterprisedb enterprisedb 65995 Dec 26 12:18
000000010000000010000005D.gz
-rw------- 1 enterprisedb enterprisedb 305 Dec 26 12:18
000000010000000010000005E.00000028.backup
-rw------- 1 enterprisedb enterprisedb 27117 Dec 26 12:18
000000010000000010000005E.gz
```

3.1.6 RESTORE

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

Syntax for Restore:

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

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

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

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

The following table describes the `RESTORE` options:

Options	Description

Options	Description
-s or --server <server_name>	<server_name> is the name of the database server to be restored.
-p or --restore-path <restore_path>	<restore_path> is the directory path where the backup of the database server is to be restored. The directory must be empty and have the proper ownership and privileges assigned to it.
-i or --backupid { <backup_id> | <backup_name>}	backup_id is the backup identifier of the backup to be used for the restoration and <backup_name> is the user-defined alphanumeric name for the backup. If the option is omitted, the latest backup is restored by default.
-r or --remote-host <remote_user@remote_host_address>	<remote_user> is the user account on the remote database server host that accepts a passwordless SSH/SCP login connection and is the owner of the directory where the backup is to be restored. <remote_host_address> is the IP address of the remote host to which the backup is to be restored. This option must be specified if the remote_host parameter for this database server is not set in the BART configuration file. For information about the remote_host parameter, see the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* available at this page.
-w or --workers <number_of_workers>	<number_of_workers> is the number of worker processes to run in parallel to stream the modified blocks of an incremental backup to the restore location. If the -w option is omitted, the default is 1 worker process. For example, if four worker processes are specified, four receiver processes on the restore host and four streamer processes on the BART host are used. The output of each streamer process is connected to the input of a receiver process. When the receiver gets to the point where it needs a modified block file, it obtains those modified blocks from its input. With this method, the modified block files are never written to the restore host disk.
-t or --target-tli <timeline_id>	<timeline_id> is the integer identifier of the timeline to be used for replaying the archived WAL files for point-in-time recovery.

Options	Description
-x or --target-xid <target_xid>	<target_xid> is the integer identifier of the transaction ID that determines the transaction up to and including, which point-in-time recovery encompasses.
-g or --target-timestamp <target_timestamp>	<target_timestamp> is the timestamp that determines the point in time up to and including, which point-in-time recovery encompasses.
-c or --copy-wals	Specify this option to copy archived WAL files from the BART backup catalog to <restore_path>/archived_wals directory. The restore_command retrieves the WAL files from <restore_path>/archived_wals for the database server archive recovery. If the -c option is omitted and the copy_wals_during_restore parameter in the BART configuration file is not enabled in a manner applicable to this database server, then the restore_command in the postgresql.conf retrieves the archived WAL files directly from the BART backup catalog. For information about the copy_wals_during_restore parameter, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide.

Examples

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

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

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

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

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

```
[root@localhost restore]# pwd
/opt/restore
[root@localhost restore]# ls -l
total 108
-rw------- 1 enterprisedb enterprisedb 208 Dec 15 10:43 backup_label
drwx------ 6 enterprisedb enterprisedb 4096 Dec 2 10:38 base
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 10:42 dbms_pipe
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 11:00 global
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_clog\
-rw------- 1 enterprisedb enterprisedb 4438 Dec 2 10:38 pg_hba.conf
-rw------- 1 enterprisedb enterprisedb 1636 Nov 10 15:38 pg_ident.conf
drwxr-xr-x 2 enterprisedb enterprisedb 4096 Dec 15 10:42 pg_log
drwx------ 4 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_multixact
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 10:42 pg_notify
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_serial
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_snapshots
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 10:42 pg_stat
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 10:43 pg_stat_tmp
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_subtrans
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 11:00 pg_tblspc
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_twophase
-rw------- 1 enterprisedb enterprisedb 4 Nov 10 15:38 PG_VERSION
drwx------ 2 enterprisedb enterprisedb 4096 Dec 15 11:00 pg_xlog
-rw------- 1 enterprisedb enterprisedb 23906 Dec 15 11:00
postgresql.conf
-rw-r--r-- 1 enterprisedb enterprisedb 217 Dec 15 11:00
postgresql.auto.conf
```

Example

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

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

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

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

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

```
-bash-4.1$ pwd
/opt/restore_pg95
-bash-4.1$ ls -l
total 128
drwxr-xr-x 2 postgres postgres 4096 Mar 29 14:27 archived_wals
-rw------- 1 postgres postgres 206 Mar 29 13:50 backup_label
drwx------ 5 postgres postgres 4096 Mar 29 12:25 base
drwx------ 2 postgres postgres 4096 Mar 29 14:27 global
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_clog
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_commit_ts
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_dynshmem
-rw------- 1 postgres postgres 4212 Mar 29 13:18 pg_hba.conf
-rw------- 1 postgres postgres 1636 Mar 29 12:25 pg_ident.conf
drwxr-xr-x 2 postgres postgres 4096 Mar 29 13:45 pg_log
drwx------ 4 postgres postgres 4096 Mar 29 12:25 pg_logical
drwx------ 4 postgres postgres 4096 Mar 29 12:25 pg_multixact
drwx------ 2 postgres postgres 4096 Mar 29 13:43 pg_notify
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_replslot
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_serial
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_snapshots
drwx------ 2 postgres postgres 4096 Mar 29 13:43 pg_stat
drwx------ 2 postgres postgres 4096 Mar 29 13:50 pg_stat_tmp
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_subtrans
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_tblspc
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_twophase
-rw------- 1 postgres postgres 4 Mar 29 12:25 PG_VERSION
drwx------ 3 postgres postgres 4096 Mar 29 14:27 pg_xlog
-rw------- 1 postgres postgres 169 Mar 29 13:24 postgresql.auto.conf
-rw-r--r-- 1 postgres postgres 21458 Mar 29 14:27 postgresql.conf
-rw-r--r-- 1 postgres postgres 118 Mar 29 14:27 postgresql.auto.conf
```

3.1.7 SHOW-SERVERS

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

Syntax:

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

The following table describes the SHOW-SERVERS option:

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

Example

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

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

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

3.1.8 SHOW-BACKUPS

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

Syntax:

> bart SHOW-BACKUPS [–s { <server_name> | all }]

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

The following table describes the `SHOW-BACKUPS` options:

Options	Description

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

Example

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

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

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

```
-bash-4.2$ bart SHOW-BACKUPS -s dev -i 1477579596637 -t
SERVER NAME : dev
BACKUP ID : 1477579596637
BACKUP NAME : dev_2016-10-27T10:46:36
BACKUP PARENT : none
BACKUP STATUS : active
BACKUP TIME : 2016-10-27 10:46:37 EDT
BACKUP SIZE : 54.50 MB
WAL(S) SIZE : 80.00 MB
NO. OF WALS : 5
FIRST WAL FILE : 000000010000000100000EC
CREATION TIME : 2016-10-27 10:46:37 EDT
LAST WAL FILE : 000000010000000100000F0
CREATION TIME : 2016-10-27 11:22:01 EDT
```

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

```
-bash-4.2$ bart SHOW-BACKUPS
SERVER NAME             BACKUP ID       BACKUP NAME       BACKUP PARENT
```

```
BACKUP TIME              BACKUP SIZE       WAL(s) SIZE       WAL FILES       STATUS
acctg                    1477580293193     acctg_2016-10-27  none
2016-10-27 10:58:13 EDT  16.45 MB          16.00 MB          1               active
acctg 1477580111358 acctg_2016-10-27 none 2016-10-27 10:55:11 EDT 59.71
MB 16.00 MB 1 active
```

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

```
 -bash-4.2$ bart SHOW-BACKUPS -t
SERVER NAME : acctg
BACKUP ID : 1477580293193
BACKUP NAME : none
BACKUP PARENT : acctg_2016-10-27
BACKUP STATUS : active
BACKUP TIME : 2016-10-27 10:58:13 EDT
BACKUP SIZE : 16.45 MB
WAL(S) SIZE : 16.00 MB
NO. OF WALS : 1
FIRST WAL FILE : 0000000100000002000000D9
CREATION TIME : 2016-10-27 10:58:13 EDT
LAST WAL FILE : 0000000100000002000000D9
CREATION TIME : 2016-10-27 10:58:13 EDT
SERVER NAME : acctg
BACKUP ID : 1477580111358
BACKUP NAME : acctg_2016-10-27
BACKUP PARENT : none
BACKUP STATUS : active
BACKUP TIME : 2016-10-27 10:55:11 EDT
BACKUP SIZE : 59.71 MB
WAL(S) SIZE : 16.00 MB
NO. OF WALS : 1
FIRST WAL FILE : 0000000100000002000000D8
CREATION TIME : 2016-10-27 10:55:12 EDT
LAST WAL FILE : 0000000100000002000000D8
CREATION TIME : 2016-10-27 10:55:12 EDT
```

3.1.9 VERIFY-CHKSUM

The VERIFY-CHKSUM subcommand verifies the MD5 checksums of the full backups and any user-defined tablespaces for the specified database server or for all database servers. The

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

Note

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

Syntax:

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

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

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

Example

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

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

3.1.10 Running the BART WAL Scanner

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

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

Syntax:

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

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

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

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

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

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

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

Example

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

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

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

```
[root@localhost archived_wals]# pwd
/opt/backup/acctg/archived_wals
[root@localhost archived_wals]# ls -l
total 81944
-rw------- 1 ... ... 16777216 Dec 20 09:10 0000000100000000000000ED
-rw------- 1 ... ... 16777216 Dec 20 09:06 0000000100000000000000EE
-rw------- 1 ... ... 16777216 Dec 20 09:11 0000000100000000000000EF
-rw------- 1 ... ... 16777216 Dec 20 09:15 0000000100000000000000F0
```

```
-rw------- 1 ... ... 16777216 Dec 20 09:16 000000010000000000000F1
-rw------- 1 ... ... 305      Dec 20 09:16 000000010000000000000F1.00000028.backup
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
0000000100000000ED00002800000000EE000000.mbm
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
0000000100000000EE00002800000000EF000000.mbm
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
0000000100000000EF00002800000000F0000000.mbm
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
0000000100000000F000002800000000F1000000.mbm
-rw-rw-r-- 1 ... ... 161      Dec 20 09:18
0000000100000000F100002800000000F2000000.mbm
```

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

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

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

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

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

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

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

```
-bash-4.2$ ps -ef | grep bart
enterpr+   4340  1   0   09:48   ? 00:00:00 bart-scanner --daemon
enterpr+   4341 4340 0   09:48   ? 00:00:00 bart-scanner --daemon
```

```
enterpr+   4415 3673 0   09:50     pts/0 00:00:00 grep   --color=auto bart
```

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

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

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

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

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

```
-bash-4.2$ bart-scanner -p
/opt/backup/acctg/archived_wals/0000000100000000FF000028000000100000000.mbm

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

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

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

3.2 Additional Examples

This section lists examples of the following BART operations.

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

Restoring a Database Cluster with Tablespaces

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

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

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

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

edb=# \d tbl_tblspc_2
Table "enterprisedb.tbl_tblspc_2"
Column | Type | Modifiers
-------+------+-----------
```

```
c1     | text |
Tablespace: "tblspc_2"
```

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

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

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

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

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

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

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

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

 INFO: backup completed successfully
 INFO: backup checksum: 594f69fe7d26af991d4173d3823e174f of 16587.tar
 INFO: backup checksum: 7a5507567729a21c98a15c948ff6c015 of base.tar
 INFO: backup checksum: ae8c62604c409635c9d9e82b29cc0399 of 16588.tar
 INFO:
```

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

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

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

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

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

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

restored.

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

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

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

```
[ACCTG]

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

description = "Accounting"
```

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

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

```
INFO: archiving is disabled
INFO: tablespace(s) restored
```

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

```
bash-4.1$ pwd
/opt
-bash-4.1$ ls -l restore
total 104
-rw------- 1 enterprisedb enterprisedb 206 Nov 16 16:36 backup_label.old
drwx------ 6 enterprisedb enterprisedb 4096 Nov 10 15:38 base
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:46 global
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_clog
-rw------- 1 enterprisedb enterprisedb 4438 Nov 10 16:23 pg_hba.conf
-rw------- 1 enterprisedb enterprisedb 1636 Nov 10 15:38 pg_ident.conf
drwxr-xr-x 2 enterprisedb enterprisedb 4096 Nov 16 17:45 pg_log
drwx------ 4 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_multixact
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:45 pg_notify
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_serial
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_snapshots
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:47 pg_stat
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:47 pg_stat_tmp
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_subtrans
drwx------ 2 enterprisedb enterprisedb 4096 Nov 16 17:42 pg_tblspc
drwx------ 2 enterprisedb enterprisedb 4096 Nov 10 15:38 pg_twophase
-rw------- 1 enterprisedb enterprisedb 4 Nov 10 15:38 PG_VERSION
drwx------ 3 enterprisedb enterprisedb 4096 Nov 16 17:47 pg_xlog
-rw------- 1 enterprisedb enterprisedb 23906 Nov 16 17:42 postgresql.conf
-rw------- 1 enterprisedb enterprisedb 61 Nov 16 17:45 postmaster.opts
-bash-4.1$
-bash-4.1$ ls -l restore_tblspc_1
total 4
drwx------ 3 enterprisedb enterprisedb 4096 Nov 16 16:18
PG_9.5_201306121
-bash-4.1$ ls -l restore_tblspc_2
total 4
drwx------ 3 enterprisedb enterprisedb 4096 Nov 16 16:18
PG_9.5_201306121
```

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

```
bash-4.1$ pwd
/opt/restore/pg_tblspc
-bash-4.1$ ls -l
total 0
```

```
lrwxrwxrwx 1 enterprisedb enterprisedb 21 Nov 16 17:42 16587 ->
/opt/restore_tblspc_1
lrwxrwxrwx 1 enterprisedb enterprisedb 21 Nov 16 17:42 16588 ->
/opt/restore_tblspc_2
```

`psql` queries also show the restored tablespaces:

```
edb=# \db

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

Restoring an Incremental Backup

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

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

Creating a Backup Chain

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

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

```
[BART]

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

[ACCTG]
```

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

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

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

First, a full backup is taken.

```
-bash-4.2$ bart BACKUP -s acctg --backup-name full_1
INFO: creating backup for server 'acctg'
INFO: backup identifier: '1490649204327'\
63364/63364 kB (100%), 1/1 tablespace
INFO: backup completed successfully
INFO: backup checksum: aae27d4a7c09dffc82f423221154db7e of base.tar
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1490649204327
BACKUP NAME: full_1
BACKUP PARENT: none
BACKUP LOCATION: /opt/backup/acctg/1490649204327
BACKUP SIZE: 61.88 MB
BACKUP FORMAT: tar
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
ChkSum File
aae27d4a7c09dffc82f423221154db7e base.tar
TABLESPACE(s): 0
START WAL LOCATION: 000000010000000000000000E
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2017-03-27 17:13:24 EDT
STOP TIME: 2017-03-27 17:13:25 EDT
TOTAL DURATION: 1 sec(s)
```

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

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

```
incr_1-c
INFO: creating incremental backup for server 'acctg'
INFO: checking mbm files /opt/backup/acctg/archived_wals
INFO: new backup identifier generated 1490649414316
INFO: reading directory /opt/backup/acctg/archived_wals
INFO: all files processed
NOTICE: pg_stop_backup complete, all required WAL segments have been
archived
INFO: incremental backup completed successfully
.
.
.
```

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

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

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

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

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

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

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
```

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

Restoring an Incremental Backup

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

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

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

Managing Backups

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

Using a Redundancy Retention Policy

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

```
[ACCTG]
host = 127.0.0.1
port = 5444
user = enterprisedb
```

```
archive_command = 'cp %p %a/%f'
retention_policy = 3 BACKUPS
description = "Accounting"
```

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

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

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

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

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

Note

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

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

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

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

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

```
-bash-4.1$ bart MANAGE -s acctg -l
INFO: 6 WAL file(s) will be removed
SERVER NAME: acctg
BACKUP ID: 1428502049836
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-08 10:07:30 EDT
BACKUP SIZE: 55.25 MB
WAL FILE(s): 6
WAL FILE: 000000010000000100000003
WAL FILE: 000000010000000100000002
WAL FILE: 000000010000000100000001
```

```
WAL FILE: 000000010000000100000000
WAL FILE: 0000000100000000000000E3
WAL FILE: 0000000100000000000000E2
INFO: 2 WAL file(s) will be removed
SERVER NAME: acctg
BACKUP ID: 1428422324880
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-07 11:58:45 EDT
BACKUP SIZE: 54.53 MB
WAL FILE(s): 2
WAL FILE: 0000000100000000000000E1
WAL FILE: 0000000100000000000000E0
```

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

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

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

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

```
LAST WAL FILE : 000000010000000100000003
CREATION TIME : 2015-04-09 10:25:46 EDT
```

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

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

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

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

Using a Recovery Window Retention Policy

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

This section provides examples of the following:

- How to view the calculated recovery window.

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

Viewing the Recovery Window

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

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

Viewing the Recovery Window Using the Manage Subcommand

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

For example, using the following `retention_policy` settings:

```
[ACCTG]

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

[DEV]

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

[HR]

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

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

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

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

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

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

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

Viewing the Recovery Window Using the Show-Servers Subcommand

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

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

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

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

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

```
-bash-4.1$ date
Fri Apr 10 14:57:33 EDT 2015
-bash-4.1$
-bash-4.1$ bart SHOW-SERVERS -s acctg
SERVER NAME        : acctg
HOST NAME          : 127.0.0.1
USER NAME          : enterprisedb
PORT               : 5444
REMOTE HOST        :
RETENTION POLICY   : 2015-04-07 14:57:36 EDT
DISK UTILIZATION   : 824.77 MB
NUMBER OF ARCHIVES : 37
ARCHIVE PATH       : /opt/backup/acctg/archived_wals
ARCHIVE COMMAND    : cp %p /opt/backup/acctg/archived_wals/%f
XLOG METHOD        : fetch
WAL COMPRESSION    : disabled
TABLESPACE PATH(s) :
DESCRIPTION : "Accounting"
```

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

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

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

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

```
-bash-4.1$ date
Fri Apr 10 14:59:39 EDT 2015
-bash-4.1$
-bash-4.1$ bart SHOW-SERVERS -s dev
SERVER NAME : dev
HOST NAME : 127.0.0.1
USER NAME : enterprisedb
PORT : 5445
REMOTE HOST :
```

```
RETENTION POLICY : 2015-03-20 14:59:42 EDT
DISK UTILIZATION : 434.53 MB
NUMBER OF ARCHIVES : 22
ARCHIVE PATH : /opt/backup/dev/archived_wals
ARCHIVE COMMAND : cp %p /opt/backup/dev/archived_wals/%f
XLOG METHOD : fetch
WAL COMPRESSION : disabled
TABLESPACE PATH(s) :
DESCRIPTION : "Development"
```

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

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

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

At this current point in time, backups taken on or after `2015-01-10 14:04:23 EST` would be considered `active`. Backups taken prior to `2015-01-10 14:04:23 EST` would be considered `obsolete`, except for backups marked as `keep`.

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

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

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

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

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

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

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

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

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

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

```
-bash-4.1$ bart MANAGE -s dev -n
INFO: processing server 'dev', backup '1428933278236'
INFO: processing server 'dev', backup '1428862187757'
```

```
INFO: processing server 'dev', backup '1428768351638'
INFO: processing server 'dev', backup '1428684544008'
INFO: marking backup '1428684544008' as obsolete
INFO: 14 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: processing server 'dev', backup '1428590536488'
INFO: marking backup '1428590536488' as obsolete
INFO: 3 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: processing server 'dev', backup '1428502171990'
```

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

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

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

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

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

```
INFO: 3 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: processing server 'dev', backup '1428502171990'
```

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

```
-bash-4.1$ bart MANAGE -s dev -l
INFO: 14 WAL file(s) will be removed
INFO: 1 Unused WAL file(s) will be removed
SERVER NAME: dev
BACKUP ID: 1428684544008
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-10 12:49:06 EDT
BACKUP SIZE: 5.65 MB
WAL FILE(s): 14
UNUSED WAL FILE(s): 1
WAL FILE: 0000000100000000000002E
WAL FILE: 0000000100000000000002D
WAL FILE: 0000000100000000000002C
WAL FILE: 0000000100000000000002B
WAL FILE: 0000000100000000000002A
WAL FILE: 00000001000000000000029
WAL FILE: 00000001000000000000028
WAL FILE: 00000001000000000000027
WAL FILE: 00000001000000000000026
WAL FILE: 00000001000000000000025
WAL FILE: 00000001000000000000024
WAL FILE: 00000001000000000000023
WAL FILE: 00000001000000000000022
WAL FILE: 00000001000000000000021
UNUSED WAL FILE: 0000000100000000000000F.00000028
INFO: 3 WAL file(s) will be removed
INFO: 1 Unused WAL file(s) will be removed
SERVER NAME: dev
BACKUP ID: 1428590536488
BACKUP STATUS: obsolete
BACKUP TIME: 2015-04-09 10:42:18 EDT\
BACKUP SIZE: 5.65 MB
WAL FILE(s): 3
UNUSED WAL FILE(s): 1
WAL FILE: 00000001000000000000020
WAL FILE: 0000000100000000000001F
WAL FILE: 0000000100000000000001E
UNUSED WAL FILE: 0000000100000000000000F.00000028
```

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

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

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

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

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

```
-bash-4.1$ bart MANAGE -s dev -d
INFO: removing all obsolete backups of server 'dev'
INFO: removing obsolete backup '1428684544008'
INFO: 14 WAL file(s) will be removed
INFO: 1 Unused WAL file(s) will be removed
INFO: removing WAL file '00000001000000000000002E'
INFO: removing WAL file '00000001000000000000002D'
```

```
INFO: removing WAL file '00000001000000000000002C'
INFO: removing WAL file '00000001000000000000002B'
INFO: removing WAL file '00000001000000000000002A'
INFO: removing WAL file '000000010000000000000029'
INFO: removing WAL file '000000010000000000000028'
INFO: removing WAL file '000000010000000000000027'
INFO: removing WAL file '000000010000000000000026'
INFO: removing WAL file '000000010000000000000025'
INFO: removing WAL file '000000010000000000000024'
INFO: removing WAL file '000000010000000000000023'
INFO: removing WAL file '000000010000000000000022'
INFO: removing WAL file '000000010000000000000021'
INFO: removing (unused) WAL file '00000001000000000000000F.00000028'
INFO: removing obsolete backup '1428590536488'
INFO: 3 WAL file(s) will be removed
INFO: removing WAL file '000000010000000000000020'
INFO: removing WAL file '00000001000000000000001F'
INFO: removing WAL file '00000001000000000000001E'
```

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

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

Managing Incremental Backups

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

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

DELETE subcommands when a 1-day recovery window retention policy is in effect.

Using a Redundancy Retention Policy

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

```
[ACCTG]

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

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

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

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

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

The MANAGE subcommand is invoked as shown by the following:

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

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

```
-bash-4.2$ bart SHOW-BACKUPS -s acctg
```

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

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

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

Second backup chain: `1481749997807 => 1481750098924`

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

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

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

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

```
acctg      1481750098924   ... 1481749997807   2016-12-14 16:14:59 EST ... active
acctg      1481749997807   ... none           2016-12-14 16:13:18 EST ... active
acctg      1481749992003   ... none           2016-12-14 16:13:12 EST ... active
acctg      1481749696905   ... 1481749673603   2016-12-14 16:08:17 EST ... obsolete
acctg      1481749673603   ... 1481749651927   2016-12-14 16:07:53 EST ... obsolete
acctg      1481749651927   ... 1481749619582   2016-12-14 16:07:32 EST ... obsolete
acctg      1481749619582   ... none           2016-12-14 16:07:00 EST ... obsolete
```

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

```
 -bash-4.2$ bart MANAGE -s acctg -d
INFO: removing all obsolete backups of server 'acctg'
INFO: removing obsolete backup '1481749619582'
INFO: 4 WAL file(s) will be removed
INFO: 3 incremental(s) of backup '1481749619582' will be removed
INFO: removing obsolete incremental backup '1481749696905'
INFO: removing obsolete incremental backup '1481749673603'
INFO: removing obsolete incremental backup '1481749651927'
INFO: removing WAL file '000000010000000100000000'
INFO: removing WAL file '0000000100000000000000FF'
INFO: removing WAL file '0000000100000000000000FE'
INFO: removing WAL file '0000000100000000000000FD'
INFO: 16 Unused file(s) will be removed
INFO: removing (unused) file '00000001000000010000004.00000028.backup'
.
.
.
INFO: removing (unused) file
'0000000100000000FB00002800000000FC000000.mbm'
```

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

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

Using a Recovery Window Retention Policy

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

```
[ACCTG]

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

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

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

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

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

Second backup chain: `1481553651165 => 1481554203288 => 1481559303348`

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

```
-bash-4.2$ date
Tue Dec 13 09:20:03 EST 2016
-bash-4.2$ bart MANAGE -s acctg
INFO: processing server 'acctg', backup '1481553651165'
INFO: processing server 'acctg', backup '1481552078404'
INFO: marking backup '1481552078404' as obsolete
INFO: 4 incremental(s) of backup '1481552078404' will be marked obsolete
```

```
INFO: marking incremental backup '1481559014359' as obsolete
INFO: marking incremental backup '1481554802918' as obsolete
INFO: marking incremental backup '1481553914533' as obsolete
INFO: marking incremental backup '1481553088053' as obsolete
INFO: 7 WAL file(s) marked obsolete
INFO: 1 Unused WAL file(s) present
INFO: 2 Unused file(s) (WALs included) present, use 'MANAGE -l' for the list
```

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

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

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

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

```
'acctg' from 'obsolete' to 'active'
INFO: changing status of incremental backup '1481553088053' of server
'acctg' from 'obsolete' to 'active'
INFO: 7 WAL file(s) changed
```

The backup chain has now been reset to active status.

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

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

```
 -bash-4.2$ bart DELETE -s acctg -i 1481554802918
INFO: deleting backup '1481554802918' of server 'acctg'
INFO: deleting backup '1481554802918'
INFO: 1 incremental backup(s) will be deleted
INFO: deleting incremental backup '1481559014359'
INFO: WALs of deleted backup(s) will belong to prior backup(if any), or
will be marked unused
INFO: 2 Unused file(s) will be removed
INFO: removing (unused) file '000000010000000000000BA'
INFO: removing (unused) file
'0000000100000000BA00002800000000BB000000.mbm'
INFO: backup(s) deleted
```

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

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

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

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

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

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

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

```
-bash-4.2$ bart MANAGE -s acctg -c keep -i 1481553651165
INFO: changing status of backup '1481553651165' of server 'acctg' from
'obsolete' to 'keep'
```

```
INFO: status of 2 incremental(s) of backup '1481553651165' will be
changed
INFO: changing status of incremental backup '1481559303348' of server
'acctg' from 'obsolete' to 'keep'
INFO: changing status of incremental backup '1481554203288' of server
'acctg' from 'obsolete' to 'keep'
INFO: 38 WAL file(s) changed
```

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

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

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

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

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

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

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

3.3 Sample BART System with Local and Remote Database Servers

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

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

The environment for this sample system is as follows:

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

Passwordless SSH/SCP connections are required between the following:

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

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

- The BART Configuration File shows the settings used in the BART configuration file.
- Establishing SSH/SCP Passwordless Connections <establishing_ssh/scp_passwordless_connections> provides an example of how to establish an SSH/SCP passwordless connection.
- Configuring a Replication Database User provides an example of how to configure the replication database user.
- WAL Archiving Configuration Parameters provides an example of how to configure WAL archiving.
- Creating the BART Backup Catalog provides information about creating a BART Backup Catalog.
- Starting the Database Servers with WAL Archiving provides example of starting the database servers with WAL archiving.
- Taking a Full Backup illustrates taking the first full backup of the database servers.
- Using Point-In-Time Recovery demonstrates the point-in-time recovery operation on the remote PostgreSQL database server.

The BART Configuration File

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

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

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

[MKTG]

host = 192.168.2.24
port = 5444
```

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

[HR]

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

Establishing SSH/SCP Passwordless Connections

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

Generating a Public Key File for the BART User Account

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

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

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

Now, generate the public key file:

```
[user@localhost ~]$ su - enterprisedb
Password:
```

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

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

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

Configuring Access between Local Advanced Server and the BART Host

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

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

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

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

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

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

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

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

Configuring Access from Remote Advanced Server to BART Host

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

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

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

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

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

```
+----------------+
```

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

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

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

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

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

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

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

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

Configuring Access from the BART Host to a Remote Advanced Server

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

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

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

The public key file, `id_rsa.pub`, for BART user account `enterprisedb` on the BART host that was earlier generated in Generating a Public Key File for the BART User Account, is now copied to the remote Advanced Server host on `192.168.2.24`:

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

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

```
-bash-4.1$ ssh enterprisedb@192.168.2.24
enterprisedb@192.168.2.24's password:
Last login: Tue Apr 21 09:53:18 2015 from 192.168.2.22
-bash-4.1$ pwd
```

```
/opt/PostgresPlus/9.5AS
-bash-4.1$ ls -l .ssh
total 12
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:11 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:11 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 394 Apr 23 13:12 known_hosts
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
```

Adjust the file permission on `authorized_keys`:

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

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

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

Configuring Access from a Remote PostgreSQL Server to a BART Host

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

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

```
[root@localhost 9.5]# cd /opt/PostgreSQL/9.5
[root@localhost 9.5]# mkdir .ssh
[root@localhost 9.5]# chown postgres .ssh
[root@localhost 9.5]# chgrp postgres .ssh
[root@localhost 9.5]# chmod 700 .ssh
```

```
[root@localhost 9.5]# ls -la | grep ssh
drwx------ 2 postgres postgres 4096 Apr 23 13:32 .ssh
```

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

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

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

```
-bash-4.1$ ssh enterprisedb@192.168.2.22
enterprisedb@192.168.2.22's password:
Last login: Thu Apr 23 13:19:25 2015 from 192.168.2.24
-bash-4.1$ pwd
/opt/PostgresPlus/9.5AS
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 enterprisedb enterprisedb 828 Apr 23 13:40 authorized_keys
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:04 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 394 Apr 23 13:24 known_hosts
-bash-4.1$ rm /tmp/tmp.pub
-bash-4.1$ exit
logout
Connection to 192.168.2.22 closed.
```

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

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

Configuring Access from the BART Host to Remote PostgreSQL

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

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

```
[user@localhost ~]$ su - enterprisedb
Password:
-bash-4.1$ ls -l .ssh
total 16
-rw------- 1 enterprisedb enterprisedb 828 Apr 23 13:40 authorized_keys
```

```
-rw------- 1 enterprisedb enterprisedb 1675 Apr 23 13:04 id_rsa
-rw-r--r-- 1 enterprisedb enterprisedb 416 Apr 23 13:04 id_rsa.pub
-rw-r--r-- 1 enterprisedb enterprisedb 394 Apr 23 13:24 known_hosts
```

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

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

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

```
-bash-4.1$ ssh postgres@192.168.2.24
postgres@192.168.2.24's password:
Last login: Mon Jan 26 18:08:36 2015 from 192.168.2.19
-bash-4.1$ pwd
/opt/PostgreSQL/9.5
-bash-4.1$ cat /tmp/tmp.pub >> ~/.ssh/authorized_keys
```

Adjust the file permissions on `authorized_keys`:

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

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

```
[user@localhost ~]$ su - enterprisedb
Password:
```

```
-bash-4.1$ ssh postgres@192.168.2.24
Last login: Thu Apr 23 13:52:25 2015 from 192.168.2.22
-bash-4.1$ exit
logout
Connection to 192.168.2.24 closed.
```

Configuring a Replication Database User

This section demonstrates how a replication database user is established.

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

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

```
[ACCTG]

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

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

[HR]

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

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

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

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

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

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

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

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

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

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

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

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

WAL Archiving Configuration Parameters

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

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

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

Note

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

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

```
scanner_logfile = /tmp/bart_scanner.log

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

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

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

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

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

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

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

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

```
.
.
.
[MKTG]

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

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

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

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

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

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

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

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

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

```
[BART]
```

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

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

The default, the BART `archive_command` format is:

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

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

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

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

Creating the BART Backup Catalog (backup_path)

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

```
[BART]

bart_host= enterprisedb@192.168.2.22
```

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

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

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

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

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

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

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

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

```
-bash-4.1$ bart SHOW-SERVERS
SERVER NAME :              acctg
BACKUP FRIENDLY NAME:      acctg_%year-%month-%dayT%hour:%minute
HOST NAME :                127.0.0.1
USER NAME :                enterprisedb
PORT :                     5444
```

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

```
-bash-4.1$ pwd
/opt/backup
-bash-4.1$ ls -l
total 12
drwxrwxr-x 3 enterprisedb enterprisedb 4096 Mar 29 13:16 acctg
drwxrwxr-x 3 enterprisedb enterprisedb 4096 Mar 29 13:16 hr
drwxrwxr-x 3 enterprisedb enterprisedb 4096 Mar 29 13:16 mktg
-bash-4.1$ ls -l acctg
total 4
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Mar 29 13:16 archived_wals
-bash-4.1$ ls -l hr
total 4
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Mar 29 13:16 archived_wals
-bash-4.1$ ls -l mktg
total 4
drwxrwxr-x 2 enterprisedb enterprisedb 4096 Mar 29 13:16 archived_wals
```

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

Starting the Database Servers with WAL Archiving

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

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

On remote host 192.168.2.24:

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

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

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

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

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

Taking a Full Backup

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

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

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

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

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

INFO: backup completed successfully
INFO: backup checksum: 8b010e130a105e76d01346bb56dfcf14 of base.tar.gz
INFO:
BACKUP DETAILS:
BACKUP STATUS: active
BACKUP IDENTIFIER: 1490809751193
BACKUP NAME: mktg_2017-03-29T13:49
```

```
BACKUP PARENT: none
BACKUP LOCATION: /opt/backup/mktg/1490809751193
BACKUP SIZE: 6.13 MB
BACKUP FORMAT: tar.gz
BACKUP TIMEZONE: US/Eastern
XLOG METHOD: fetch
BACKUP CHECKSUM(s): 1
ChkSum File
8b010e130a105e76d01346bb56dfcf14 base.tar.gz

TABLESPACE(s): 0
START WAL LOCATION: 000000010000000100000085
BACKUP METHOD: streamed
BACKUP FROM: master
START TIME: 2017-03-29 13:49:11 EDT
STOP TIME: 2017-03-29 13:49:14 EDT
TOTAL DURATION: 3 sec(s)

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

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

```
STOP TIME: 2017-03-29 13:50:26 EDT
TOTAL DURATION: 1 sec(s)
```

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

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

Using Point-In-Time Recovery

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

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

```
 postgres=# \dt

            List of relations
Schema  |      Name      |  Type   | Owner

--------+----------------+---------+---------
public  | hr_rmt_t1_1356 | table   | postgres
public  | hr_rmt_t1_1358 | table   | postgres
public  | hr_rmt_t1_1400 | table   | postgres
public  | hr_rmt_t1_1402 | table   | postgres
public  | hr_rmt_t1_1404 | table   | postgres
public  | hr_rmt_t1_1406 | table   | postgres
```

```
(6 rows)
```

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

The PostgreSQL database server was then stopped.

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

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

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

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

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

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

```
-bash-4.1$ mkdir /tmp/unarchived_pg95_wals
-bash-4.1$ pwd
/opt/PostgreSQL/9.5/data/pg_xlog
bash-4.1$ cp -p 000000010000000000000004 /tmp/unarchived_pg95_wals
```

```
bash-4.1$ cp -p 000000010000000000000005 /tmp/unarchived_pg95_wals
bash-4.1$ ls -l /tmp/unarchived_pg95_wals
total 32768
-rw------- 1 postgres postgres 16777216 Mar 29 14:07 000000010000000000000004
-rw------- 1 postgres postgres 16777216 Mar 29 13:50 000000010000000000000005
```

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

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

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

```
[HR]

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

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

```
SERVER NAME        BACKUP ID        BACKUP NAME          BACKUP PARENT
BACKUP TIME
BACKUP SIZE        WAL(s) SIZE      WAL FILES            STATUS
```

```
acctg              1490809695281     acctg_2017-03-29T13:48  none
2017-03-29 13:48:17 EDT
6.10 MB            32.00 MB          2                       active
hr                 1490809824946     hr_2017-03-29T13:50     none
2017-03-29 13:50:26 EDT
2.59 MB            32.00 MB          2                       active
mktg               1490809751193     mktg_2017-03-29T13:49   none
2017-03-29 13:49:14 EDT
6.13 MB            64.00 MB          4                       active
```

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

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

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

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

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

```
-bash-4.1$ pwd
/opt/restore_pg95
```

```
-bash-4.1$ ls -l
total 128
drwxr-xr-x 2 postgres postgres 4096 Mar 29 14:27 archived_wals
-rw------- 1 postgres postgres 206 Mar 29 13:50 backup_label
drwx------ 5 postgres postgres 4096 Mar 29 12:25 base
drwx------ 2 postgres postgres 4096 Mar 29 14:27 global
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_clog
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_commit_ts
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_dynshmem
-rw------- 1 postgres postgres 4212 Mar 29 13:18 pg_hba.conf
-rw------- 1 postgres postgres 1636 Mar 29 12:25 pg_ident.conf
drwxr-xr-x 2 postgres postgres 4096 Mar 29 13:45 pg_log
drwx------ 4 postgres postgres 4096 Mar 29 12:25 pg_logical
drwx------ 4 postgres postgres 4096 Mar 29 12:25 pg_multixact
drwx------ 2 postgres postgres 4096 Mar 29 13:43 pg_notify
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_replslot
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_serial
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_snapshots
drwx------ 2 postgres postgres 4096 Mar 29 13:43 pg_stat
drwx------ 2 postgres postgres 4096 Mar 29 13:50 pg_stat_tmp
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_subtrans
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_tblspc
drwx------ 2 postgres postgres 4096 Mar 29 12:25 pg_twophase
-rw------- 1 postgres postgres 4 Mar 29 12:25 PG_VERSION
drwx------ 3 postgres postgres 4096 Mar 29 14:27 pg_xlog
-rw------- 1 postgres postgres 169 Mar 29 13:24 postgresql.auto.conf
-rw-r--r-- 1 postgres postgres 21458 Mar 29 14:27 postgresql.conf
-rw-r--r-- 1 postgres postgres 118 Mar 29 14:27 postgresql.auto.conf
```

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

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

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

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

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

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

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

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

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

```
2017-03-29 14:33:23 EDT LOG: database system was interrupted; last known
up at 2017-03-29 13:50:25 EDT
2017-03-29 14:33:23 EDT LOG: starting point-in-time recovery to
2017-03-29 14:01:00-04
2017-03-29 14:33:23 EDT LOG: restored log file
"000000010000000000000002" from archive
2017-03-29 14:33:23 EDT LOG: redo starts at 0/2000098
2017-03-29 14:33:23 EDT LOG: consistent recovery state reached at
0/20000C0
2017-03-29 14:33:23 EDT LOG: restored log file
"000000010000000000000003" from archive
2017-03-29 14:33:23 EDT LOG: recovery stopping before commit of
transaction 1762, time 2017-03-29 14:02:28.100072-04
```

```
2017-03-29 14:33:23 EDT LOG: redo done at 0/303F390
2017-03-29 14:33:23 EDT LOG: last completed transaction was at log time
2017-03-29 14:00:43.351333-04
cp: cannot stat `archived_wals/00000002.history': No such file or
directory
2017-03-29 14:33:23 EDT LOG: selected new timeline ID: 2
cp: cannot stat `archived_wals/00000001.history': No such file or
directory
2017-03-29 14:33:23 EDT LOG: archive recovery complete
2017-03-29 14:33:23 EDT LOG: MultiXact member wraparound protections are
now enabled
2017-03-29 14:33:23 EDT LOG: database system is ready to accept
connections
2017-03-29 14:33:23 EDT LOG: autovacuum launcher started
```

The tables that exist in the recovered database cluster are:

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

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

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

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

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

4 Backup and Recovery User Guide

4.1 Introduction

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.
- information about the backup and recovery management process.
- information about using tablespaces.

To view information about BART installation and upgrade, see the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* and to view examples of BART operations and subcommands, see the *EDB Postgres Backup and Recovery Reference Guide*. These guides are available at the EnterpriseDB documentation page.

4.1.1 What's New

BART supports installation on a SLES 12 host. For information about how to install BART on SLES, please see the EDB Postgres Backup and Recovery Installation and Upgrade Guide.

4.1.2 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`.
    - For Advanced Server Linux installations performed using the interactive installer for version 10 and earlier, this defaults to `/opt/PostgresPlus/x.xAS` or `/opt/edb/asx.x`. For Advanced Server Linux installations performed with an RPM package, this defaults to `/usr/ppas-x.x` or `/usr/edb/asx.x`. For Advanced Server Linux installations performed with an RPM package for version 11 or later, this defaults to `/usr/edb/asxx`.

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 this page

For information about `pg_basebackup`, see the PostgreSQL Core Documentation.

4.2 Overview

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

- Capturing a complete image of a database cluster as a full base backup or referred to simply as

a *full backup*.
- Capturing a modified image of a database cluster called a *block-level incremental backup* 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 <block-level_incremental_backup> 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 incremental and parallel backups. For information about standby servers, see the PostgreSQL Documentation.

BART simplifies the management process by use of a centralized backup catalog, a single configuration file, and a command line interface controlling the necessary operations. Reasonable defaults are automatically used for various backup and restore options. BART also performs the necessary recovery file configuration required for point-in-time recovery 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 used by BART.
- **BART Backup Catalog.** File system directory structure containing all of the backups for the database servers managed by BART. 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 managed by BART, a database superuser must be selected to act as the replication database user. This database user is used to connect to the database server when backups are taken. The database superusers created with an initial Postgres database server installation (`enterprisedb` or `postgres`) may be used for this purpose.

  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.

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).

	-Fp-xlog-method=fetch	-Fp-xlog-method=stream	-Ft-xlog-method=fetch	-Ft-xlog-method=stream
Master Database Server/Full backup	Supported	Supported	Supported	Supported

	-Fp-xlog-method=fetch	-Fp-xlog-method=stream	-Ft-xlog-method=fetch	-Ft-xlog-method=stream
Master 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

Backup

	Wal compression by BART	WAL scanner
Master Database Server/Full backup	Supported	Not required
Master Database Server/Incremental backup	Not Supported	required
Standby Database Server/Full backup	Supported	Not required
Standby Database Server/Incremental backup	Not Supported	Not supported

Wal Archiving

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

Restore

4.2.1 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 information.

4.2.1.1 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.

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

4.2.1.2 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.

2. Take an initial full backup with the `BACKUP` subcommand. This full backup establishes the parent of the first incremental backup.

3. 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.

4. 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.

5. 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* 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.

6. 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.

4.2.1.3 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 000000010000000178000028000000179000000.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 00000001000000017F000028000000180000000.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 `rsync` utility 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* for more information about the `scan_interval` parameter.

See Running the BART WAL Scanner for information about using the WAL scanner.

4.2.1.4 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 } ]
```

Note

While a BACKUP subcommand is in progress, no other processes must run in parallel.

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 ]
```

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.

4.2.1.5 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.

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.

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.

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.

Restoring an Incremental Backup on a Remote Host

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.

When restoring an incremental backup, specify the RESTORE subcommand and the backup identifier or name of the incremental backup that will be restored. To view an example of restoring an incremental backup, see the EDB Postgres Backup and Recovery Reference Guide.

4.2.2 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 for examples of creating a backup chain and restoring an incremental backup.

4.3 Using BART

After installing and configuring the BART host and the database servers, you can start using BART. For detailed information about installation and configuration, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide.

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

4.3.1 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.

4.3.1.1 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 9.5 or 9.6 versions) 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 9.5 or 9.6 versions) 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* 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.

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.

4.3.1.2 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:
   1. restore the database cluster and tablespace files to new, empty directories, create the new directories with the appropriate directory ownership and permissions.
   2. 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 9.5 or 9.6 versions) 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
  ```

1. 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 9.5 or 9.6 versions).

2. 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.

3. 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.

4. 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.

5. 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 this page. See Restore for more information about using the `Restore` subcommand.

4.3.2 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.

4.3.2.1 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.

4.3.2.2 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.

4.3.2.3 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.

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.

4.3.2.4 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 <evaluating_marking_and_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 Po	licy	
	Active	Obsolete	Keep	Active	Obsolete	Keep
MANAGE –d	No	Yes	No	No	Yes	No
DELETE –i *backup*	Yes (see Note1*)*	Yes (see Note1)	Yes (see Note1*)*	Yes (see Note2	Yes) | (see Note2*)*	Yes (see Note2)

DELETE –i all	Yes	Yes	Yes	Yes	Yes	Yes

Note

Redundancy Retention Policy (Note1) : 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 (Note2): 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* 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.

4.3.2.5 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* 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* for examples demonstrating use of the `MANAGE` and `DELETE` subcommands when a recovery window retention policy is in effect.

4.3.3 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 this page.

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
`-h` or `–help`	Displays general syntax and information on BART usage. All subcommands support a help option (`-h`, `--help`). If the help option is specified, information is displayed regarding that particular subcommand. The subcommand, itself, is not executed.
`-v` or `–version`	Displays the BART version information.
`-d` or `–debug`	Displays debugging output while executing BART subcommands.
`-c` or `--config-path config_file_path`	Specifies `config_file_path` as the full directory path to a BART configuration file. Use this option if you do not want to use the default BART configuration file `<BART_HOME>/etc/bart.cfg`.

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* 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.

4.3.3.1 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
`-s` or `--server <server_name>`	`server_name` is the name of the database server to be checked for proper configuration. If the option is omitted, the settings of the global section of the BART configuration file are checked.

- 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.

4.3.3.2 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 or --server {<server_name> | all }	server_name is the name of the database server to which the INIT actions are to be applied. If all is specified or if the option is omitted, the actions are applied to all servers.
-o or –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 or –rebuild	Rebuilds the backupinfo file located in each backup subdirectory. If all is specified or if the option is omitted, the backupinfo files of all backups for the database servers specified by the -s option are recreated. This option is only intended for recovering from a situation where the backupinfo file has become corrupt. If the backup was initially created with a user-defined backup name, and then the INIT -r option is invoked to rebuild that backupinfo file, the user-defined backup name is no longer available. Thus, future references to the backup must use the backup identifier.
-i or --backupid { <backup_id> | <backup_name> | all }	<backup_id> is an integer, backup identifier and <backup_name> is the user-defined alphanumeric name for the backup. 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.

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.

4.3.3.3 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 ]
```

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.
- If in the BART configuration file, parameter setting `xlog_method=stream` applies to a given database server, streaming of the transaction log in parallel with creation of the backup is performed for that database server, otherwise the transaction log files are collected upon completion of the backup. See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* 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 version 9.5 or prior, only one backup per server may be in progress at any given time and if a backup is interrupted, you must manually run the `pg_stop_backup()` command to terminate the backup mode.
  - PostgreSQL server version 9.6 or higher, 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

Options	Description
-s or --server { server_name | all }	server_name is the database server name to be backed up as specified in the BART configuration file. If all is specified, all servers are backed up. This option is mandatory. If all is specified, and a connection to a database server listed in the BART configuration file cannot be opened, the backup for that database server is skipped, but the backup operation continues for the other database servers.

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
-F { p | t } or --format { p | t }	Specify this option to provide the backup file format. Use p for plain text or t for tar. If the option is omitted, the default is tar format. For taking incremental backups, the option -Fp must be specified.
-z or --gzip (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.
-c compression_level or --compress-level compression_level (applicable only for full backup)	Specify this option to use the gzip compression level on the tar file output. compression_level is a digit from 1 through 9, with 9 being the best compression. This option is applicable only for the tar format.
--parent { backup_id | backup_name }	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. The parent is a backup taken prior to the incremental backup. The parent backup can be either a full backup or an incremental backup. The option –Fp must be specified since an incremental backup can only be taken in plain text format. An incremental backup cannot be taken on a standby database server. See Block-Level Incremental Backup for additional information on incremental backups.

Options	Description
--backup-name <backup_name>	Specify this option to assign a user-defined, alphanumeric friendly name to the backup. The maximum permitted length of backup name is 49 characters. The backup name may include the following variables to be substituted by the timestamp values when the backup is taken: 1) %year – 4-digit year, 2) %month – 2-digit month, 3) %day – 2-digit day, 4) %hour 2-digit hour, 5) %minute – 2-digit minute, and 6) %second – 2-digit second. To include the percent sign (%) as a character in the backup name, specify %% in the alphanumeric string. If the backup name contains space characters (i.e. more than one word) or when referenced with the option -i by other subcommands (such as restore), enclose the string in single quotes or double quotes. See backup name examples. If the --backup-name option is not specified, and the backup_name parameter is not set for this database server in the BART configuration file, then the backup can only be referenced in other BART subcommands by the BART assigned backup identifier.
--thread-count <number_of_threads>	Use this option to use the number of worker threads to run in parallel to copy blocks for a backup. If the option --thread-count is omitted, then the thread_count parameter in the BART configuration file applicable to this database server is used. If the option --thread-count is not enabled for this database server, then the thread_count setting in the global section of the BART configuration file is used. If the option --thread-count is not set in the global section as well, the default number of threads is 1. If parallel backup is run with N number of worker threads, then it will initiate N+ 1 concurrent connections with the server. Thread count will not be effective if backup is taken on a standby server. For more information about the --thread-count parameter, see the EDB Postgres Backup and Recovery Installation and Upgrade Guide'
--with-pg_basebackup (applicable only for full backup)	Specify this option to use pg_basebackup to take a full backup. The number of thread counts in effect is ignored as given by the thread_count parameter in the BART configuration file. When taking a full backup, if the thread count in effect is greater than 1, then the pg_basebackup utility is not used to take the full backup (parallel worker threads are used) unless the option --with-pg_basebackup is specified with the BACKUP subcommand.

Options	Description
--no-pg_basebackup (applicable only for full backup)	Specify this option if you do not want pg_basebackup to be used to take a full backup. When taking a full backup, if the thread count in effect is only 1, then the pg_basebackup utility is used to take the full backup unless the option --no-pg_basebackup is specified with the BACKUP subcommand.
--check (applicable only for incremental backup)	Specify this option to verify if the required MBM files are present in the archived_wals directory as specified in the archive_path parameter in the bart.cfg file before taking an incremental backup. The option --parent must be specified when the option --check is used. An actual incremental backup is not taken when the option --check is specified.

--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
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	Insufficient free disk space.

error message	Cause
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 000000010000000D50000006B has already been removed	The wal_keep_segments configuration parameter is not set to a sufficiently large value in the postgresql.conf file.
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?	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

4.3.3.4 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
`-s` or `--server { <server_name>	all }`

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

4.3.3.6 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
`-s or --server { <server_name>	all }`
`-i or --backupid { <backup_id>	<backup_name>
<backup_name> is the user-defined alphanumeric name for the full backup.	If all is specified or if the -i option is omitted, the checksums of all tar backups for the relevant database server are verified.

4.3.3.7 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* 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
s or --server { <server_name> | all }	<server_name> is the name of the database server to which the actions are to be applied. If all is specified or if the -s option is omitted, the actions are applied to all database servers.
-l or --list-obsolete	Lists the backups marked as obsolete.
-d or --delete-obsolete	Delete the backups marked as obsolete. This action physically deletes the backup along with its archived WAL files and any MBM files for incremental backups.
-c or --change-status { keep | nokeep }	Specify keep to change the status of a backup to keep to retain it indefinitely. Specify nokeep to change the status of any backup back to active status. The backup can then be re-evaluated and possibly be marked to obsolete according to the retention policy by subsequent usage of the MANAGE subcommand. The –i option must be included when using the -c option.
-i or --backupid { <backup_id> | <backup_name> | all }	<backup_id> is a backup identifier and <backup_name> is the user-defined alphanumeric name for the backup. If all is specified, then actions are applied to all backups. The –c option must be included when using the -i option.
-n, --dry-run	Performs the test run and displays the results prior to actually implementing the actions as if the operation was performed, however, no changes are actually made. If -n is specified with the -d option, it displays which backups would be deleted, but does not actually delete the backups. If -n is specified with the -c option, it displays the keep or nokeep action, but does not actually change the backup from its current status. If -n is specified alone with no other options, or with only the -s option, it displays which active backups would be marked as obsolete, but does not actually change the backup status. In addition, no compression is performed on uncompressed, archived WAL files even if WAL compression is enabled for the database server.

4.3.3.8 RESTORE

The RESTORE subcommand restores the backup and its archived WAL files for the designated database server to the specified directory location. If the appropriate RESTORE options are specified, 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 ]
```

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 this page.

The following table describes the command options:

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

Options	Description
-r or --remote-host <remote_user@remote_host_address>	<remote_user> is the user account on the remote database server host that accepts a passwordless SSH/SCP login connection and is the owner of the directory where the backup is to be restored and <remote_host_address> is the IP address of the remote host to which the backup is to be restored. This option must be specified if the <remote_host> parameter for this database server is not set in the BART configuration file. If the BART user account is not the same as the operating system account owning the <restore_path> directory given with the -p option, use the <remote_host> BART configuration parameter or the RESTORE subcommand -r option to specify the <restore_path> directory owner even when restoring to a directory on the same host as the BART host. See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about the <remote_host> parameter.
-w or --workers <number_of_workers>	<number_of_workers> is the specification of the number of worker processes to run in parallel to stream the modified blocks of an incremental backup to the restore location. For example, if 4 worker processes are specified, 4 receiver processes on the restore host and 4 streamer processes on the BART host are used. The output of each streamer process is connected to the input of a receiver process. When the receiver gets to the point where it needs a modified block file, it obtains those modified blocks from its input. With this method, the modified block files are never written to the restore host disk. If the -w option is omitted, the default is 1 | worker process.
-t or --target-tli <timeline_id>	<timeline_id> is the integer identifier of the timeline to be used for replaying the archived WAL files for point-in-time recovery.
-x or --target-xid <target_xid>	<target_xid> is the integer identifier of the transaction ID that determines the transaction up to and including, which point-in-time recovery encompasses. Include either the -x <target_xid> or the --target-xid <target_xid> option if point-in-time recovery is desired.

Options	Description
-g or --target-timestamp <target_timestamp>	<target_timestamp> is the timestamp that determines the point in time up to and including, which point-in-time recovery encompasses. Include either the --target-timestamp <target_timestamp> or the -g <target_timestamp> option if point-in-time recovery is desired.
-c or --copy-wals	Specify this option to copy archived WAL files from the BART backup catalog to <restore_path>/archived_wals directory. If recovery settings are saved in the postgresql.auto.conf file for point-in-time recovery, the restore_command retrieves the WAL files from <restore_path>/archived_wals for the database server archive recovery. If the -c option is omitted and the copy_wals_during_restore parameter in the BART configuration file is not enabled in a manner applicable to this database server, the restore_command in the postgresql.auto.conf file is generated by default to retrieve the archived WAL files directly from the BART backup catalog. See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for information about the copy_wals_during_restore parameter.

4.3.3.9 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
-s or --server <server_name>	<server_name> is the name of the database server whose backups are to be deleted.
-i or --backupid { all | ['']{ <backup_id> | <backup_name> },... }['] }	<backup_id> is the backup identifier of the backup to be deleted 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. If all is specified, all of the backups and their archived WAL files for the specified database server are deleted.
-n or --dry-run	Displays the results as if the deletions were done, however, no physical removal of the files are actually performed. In other words, a test run is performed so that you can see the potential results prior to actually initiating the action. 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.

4.3.4 Running the BART WAL Scanner

Use the BART WAL scanner to invoke the bart-scanner program located in the BART_HOME/bin directory.

Syntax:

```
bart-scanner
  [ -d ]
  [ -c <config_file_path> ]
  { –h |
    -v |
    --daemon |
    -p mbm_file |
```

```
  wal_file |
  RELOAD |
  STOP }
```

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 `rsync` utility and also in case of some specific platforms. This results in the WAL files being copied to the `archived_wals` directory, but the WAL scanner 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 scan the new WAL files.

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

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

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

See the *EDB Postgres Backup and Recovery Installation and Upgrade Guide* for additional information about WAL scanning, `scan_interval`, `allow_incremental_backups`, and `logfile` parameters.

When invoking the WAL scanner, the current user must be the BART user account.

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

Options	Description
-h or —help	Displays general syntax and information on WAL scanner usage.
-v or —version	Displays the WAL scanner version information.
-d or –debug	Displays debugging output while executing the WAL scanner with any of its options.
-c or --config-path config_file_path	Specifies config_file_path as the full directory path to a BART configuration file. Use this option if you do not want to use the default BART configuration file BART_HOME/etc/bart.cfg.
--daemon	Runs the WAL scanner as a background process.
-p or --print mbm_file	Specifies the full directory path to an MBM file whose content is to be printed. The directory specified in the archive_path parameter in the bart.cfg file contains the MBM files.
wal_file	Specifies the full directory path to a WAL file to be scanned. The directory specified in the archive_path parameter in the bart.cfg file contains the WAL files. Use it if a WAL file in the archive path is missing its MBM file. This option is to be used for assisting the EnterpriseDB support team for debugging problems that may have been encountered.
RELOAD	Reloads the BART configuration file. The keyword RELOAD is not case-sensitive. The RELOAD option is useful if you make changes to the configuration file after the WAL scanner has been started. It will reload the configuration file and adjust the WAL scanners accordingly. For example, if a server section allowing incremental backups is removed from the BART configuration file, then the process attached to that server will stop. Similarly, if a server allowing incremental backups is added, a new WAL scanner process will be launched to scan the WAL files of that server.
STOP	Stops the WAL scanner. The keyword STOP is not case-sensitive.

4.4 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/9.5AS/data/pg_tblspc
[root@localhost pg_tblspc]# ls -l
total 0
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Aug 22 16:38 16644 -> /mnt/tablespace_1
lrwxrwxrwx 1 enterprisedb enterprisedb 17 Aug 22 16:38 16645 -> /mnt/tablespace_2
```

The OIDs are `16644` and `16645` to directories `/mnt/tablespace_1` and `/mnt/tablespace_2`, respectively.

- If you later wish to restore the tablespaces to the same locations as indicated in the preceding example, the BART configuration file must contain the following entry:

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

- If you 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.

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*.