Fujitsu
Enterprise Postgres 16

Operation Guide

Linux

J2UL-2952-01PEZ0(00)
June 2024

Preface

Purpose of this document

The Fujitsu Enterprise Postgres database system extends the PostgreSQL features and runs on the Linux platform.

This document is the Fujitsu Enterprise Postgres Operation Guide.

Intended readers

This document is intended for those who install and operate Fujitsu Enterprise Postgres.

Readers of this document are assumed to have general knowledge of:

- PostgreSQL
- SQL
- Linux

Structure of this document

This document is structured as follows:

Chapter 1 Operating Fujitsu Enterprise Postgres

Describes how to operate Fujitsu Enterprise Postgres.

Chapter 2 Starting an Instance and Creating a Database

Describes how to start a Fujitsu Enterprise Postgres instance, and how to create a database.

Chapter 3 Backing Up the Database

Describes how to back up the database.

Chapter 4 Configuring Secure Communication Using Secure Sockets Layer

Describes communication data encryption between the client and the server.

Chapter 5 Protecting Storage Data Using Transparent Data Encryption

Describes how to encrypt the data to be stored in the database.

Chapter 6 Using Transparent Data Encryption with Key Management Systems as Keystores

Describes the operation of transparent data encryption when a key management system is used as a keystore.

Chapter 7 Policy-based Password Management

Describes the password opereation based on policy in client authentication.

Chapter 8 Data Masking

Describes the data masking feature.

Chapter 9 Periodic Operations

Describes the periodic database operations that must be performed on Fujitsu Enterprise Postgres.

Chapter 10 Streaming Replication Using WebAdmin

Describes how to create a streaming replication cluster using WebAdmin.

Chapter 11 Installing and Operating the In-memory Feature

Describes how to install and operate the in-memory feature.

Chapter 12 Parallel Query

Describes the factors taken into consideration by Fujitsu Enterprise Postgres when performing parallel queries.

Chapter 13 High-Speed Data Load

Describes how to install and operate high-speed data load.

Chapter 14 Global Meta Cache

Describes how to use Grobal Meta Cache feature.

Chapter 15 Local Meta Cache Limit

Describes how to use Local Meta Cache Limit feature.

Chapter 16 Backup/Recovery Using the Copy Command

Describes backup and recovery using the copy command created by the user.

Chapter 17 Actions when an Error Occurs

Describes how to perform recovery when disk failure or data corruption occurs.

Appendix A Parameters

Describes the Fujitsu Enterprise Postgres parameters.

Appendix B System Administration Functions

Describes the system administration functions of Fujitsu Enterprise Postgres.

Appendix C System Catalogs

Describes the system catalog used by Fujitsu Enterprise Postgres.

Appendix D System Views

Describes the system view used by Fujitsu Enterprise Postgres.

Appendix E Tables Used by Transparent Data Encryption

Describes the tables used by the transparent data encryption feature.

Appendix F Tables Used by Data Masking

Describes the tables used by the data masking feature.

Appendix G Tables Used by High-Speed Data Load

Describes the tables used by high-speed data load.

Appendix H Starting and Stopping the Web Server Feature of WebAdmin

Describes how to start and stop WebAdmin (Web server feature).

Appendix I WebAdmin Wallet

Describes how to use the Wallet feature of WebAdmin.

Appendix J WebAdmin Disallow User Inputs Containing Hazardous Characters

Describes characters not allowed in WebAdmin.

Appendix K Collecting Failure Investigation Data

Describes how to collect information for initial investigation.

Export restrictions

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Issue date and version

```
Edition 1.0: June  2024
```

Copyright

Contents

Chapter 1 Operating Fujitsu Enterprise Postgres

This chapter describes how to operate Fujitsu Enterprise Postgres.

1.1 Operating Methods

There are two methods of managing Fujitsu Enterprise Postgres operations:

- Operation management using GUI tools
- Operation management using commands

See

Before performing database multiplexing using database multiplexing, refer to "Database Multiplexing Mode" in the Cluster Operation Guide (Database Multiplexing).

Operation management using GUI tools

This involves managing operations using the WebAdmin.

- Management using WebAdmin

  This removes the requirement for complex environment settings and operational design for backup and recovery that is usually required for running a database. It enables you to easily and reliably monitor the state of the database, create a streaming replication cluster, back up the database, and restore it even if you do not have expert knowledge of databases.

Operation management using commands

You can use commands for configuring and operating the database and managing operations.

Points to consider when choosing an operation method

- You cannot combine WebAdmin and server commands to perform the following operations:
  - Use commands to operate an instance created using WebAdmin.
  - Use WebAdmin to recover a database backed up using commands.

  For instances created with WebAdmin, however, backup can be obtained with the pgx_dmpall command. Also, WebAdmin can perform recovery by using the backup obtained with the pgx_dmpall command.
- To operate an instance created using the initdb command in WebAdmin, the instance needs to be imported using WebAdmin.

Features used in each phase

The following table lists the features used in each phase for GUI-based operations and command-based operations.

Operation		Operation with the GUI	Operation with commands
Setup	Creating an instance	WebAdmin is used. The server machine capacity, and the optimum parameter for operations using WebAdmin, are set automatically.	The configuration file is edited directly using the initdb command.
	Creating a standby instance	WebAdmin is used. WebAdmin performs a base backup of the source instance and creates a standby instance.	A standby instance is created using the pg_basebackup command.

Operation		Operation with the GUI	Operation with commands
	Changing the configuration files	WebAdmin is used.	The configuration file is edited directly.
Starting and stopping an instance		WebAdmin is used.	The pg_ctl command is used.
Creating a database		None.	This is defined using the psql command or the application after specifying the DDL statement.
Backing up the database		WebAdmin, or the pgx_dmpall command, is used.	It is recommended that the pgx_dmpall command be used. Recovery to the latest database can be performed.
Database recovery		WebAdmin is used.	To use the backup that was performed using the pgx_dmpall command, the pgx_rcvall command is used.
Monitoring	Database errors	The status in the WebAdmin window can be checked.	The messages that are output to the database server log are monitored.
	Disk space	The status in the WebAdmin window can be checked. A warning will be displayed if the free space falls below 20%.	This is monitored using the df command of the operating system, for example.
	Connection status	None.	This can be checked referencing pg_stat_activity of the standard statistics view from psql or the application.

1.2 Starting WebAdmin

This section describes how to start and log in to WebAdmin.

About using WebAdmin

- It is recommended to use the following browsers with WebAdmin:
    - Microsoft Edge (Build41 or later)

      WebAdmin will work with other browsers, such as Firefox and Chrome, however, the look and feel may be slightly different.
- You must start the Web server feature of WebAdmin before using WebAdmin. Refer to "Appendix H Starting and Stopping the Web Server Feature of WebAdmin" for information on how to start the Web server feature of WebAdmin.

1.2.1 Logging in to WebAdmin

This section describes how to log in to WebAdmin.

Startup URL for WebAdmin

In the browser address bar, type the startup URL of the WebAdmin window in the following format:

```
http://hostNameOrIpAddress:portNumber/
```

- *hostNameOrIpAddress*: The host name or IP address of the server where WebAdmin is installed.
- *portNumber*: The port number of WebAdmin. The default port number is 27515.

Example

For a server with IP address "192.0.2.0" and port number "27515"

```
http://192.0.2.0:27515/
```

Display the startup windows. From this window you can log in to WebAdmin or access the product documentation.

Log in to WebAdmin

Click [Launch WebAdmin] in the startup URL window to start WebAdmin and display the login window.

To log in, specify the following values:

- [User name]: User name (OS user account) of the instance administrator
- [Password]: Password corresponding to the user name

1.3 Operations Using Commands

You can operate and manage the database using the following commands:

- Server commands

  This group of commands includes commands for creating a database cluster and controlling the database. You can run these commands on the server where the database is operating.

  To use these commands, you must configure the environment variables.

  See

  - Refer to "PostgreSQL Server Applications" under "Reference" in the PostgreSQL Documentation, or "Reference" for information on server commands.
  - Refer to "Configure the environment variables" in the procedure to create instances in "Using the initdb Command" in the Installation and Setup Guide for Server for information on configuring the environment variables.

- Client commands

  This group of commands includes the psql command and commands for extracting the database cluster to a script file. These commands can be executed on the client that can connect to the database, or on the server on which the database is running.

  To use these commands, you must configure the environment variables.

  See

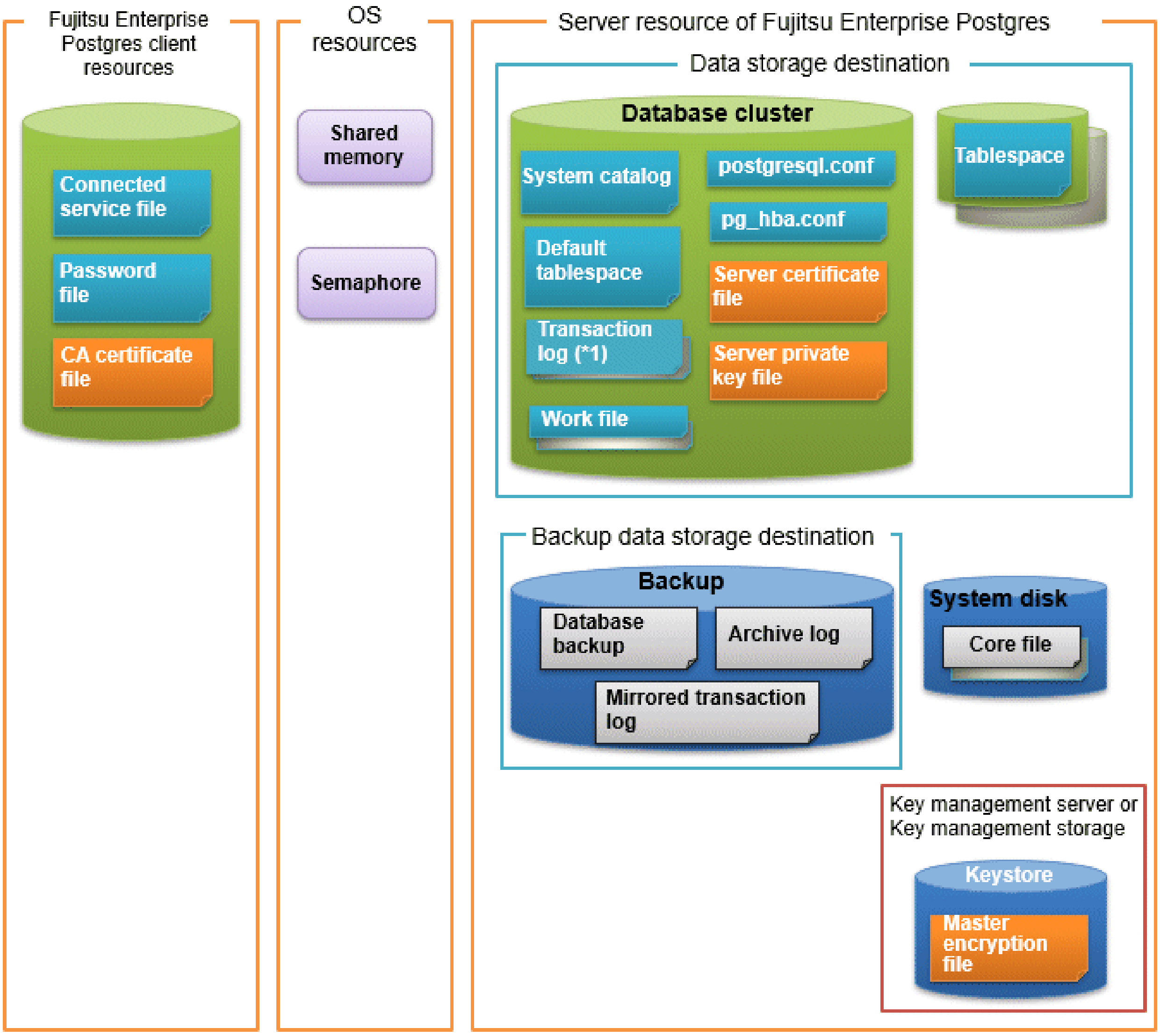
  - Refer to "PostgreSQL Client Applications" under "Reference" in the PostgreSQL Documentation, or "Reference" for information on client commands.
  - Refer to "Configuring Environment Variables" in the Installation and Setup Guide for Client for information on the values to be set in the environment variables.

1.4 Operating Environment of Fujitsu Enterprise Postgres

This section describes the operating environment and the file composition of Fujitsu Enterprise Postgres.

1.4.1 Operating Environment

The following figure shows the configuration of the Fujitsu Enterprise Postgres operating environment. The tables given below list the roles of the OS resources and Fujitsu Enterprise Postgres resources.



*1: To distribute the I/O load, place the transaction log on a different disk from the data storage destination.

Table 1.1 OS resources

Type	Role
Shared memory	Used when a database process exchanges information with an external process.
Semaphore	

Table 1.2 Fujitsu Enterprise Postgres client resources

Type	Role
Connection service file	Specifies information, such as the host name, user name, and password, for connecting to Fujitsu Enterprise Postgres.
Password file	Securely manages the password for connecting to Fujitsu Enterprise Postgres.
CA certificate file	CA (certificate authority) certificate used for server authentication when encrypting communication data.

Table 1.3 Server resources of Fujitsu Enterprise Postgres

Type	Role
Database cluster	Database storage area on the database storage disk. It is a collection of databases managed by an instance.
System catalog	Contains information required for the system to run, including the database definition information and the operation information created by the user.
Default tablespace	Contains table files and index files stored by default.
Transaction log	Contains log information in case of a crash recovery or rollback. This is the same as the WAL (Write Ahead Log).
Work file	Work file used when executing applications or commands.
postgresql.conf	Contains information that defines the operating environment of Fujitsu Enterprise Postgres.
pg_hba.conf	Fujitsu Enterprise Postgres uses this file to authenticate individual client hosts.
Server certificate file	Contains information about the server certificate to be used when encrypting communication data and authenticating a server.
Server private key file	Contains information about the server private key to be used when encrypting communication data and authenticating a server
Tablespace	Stores table files and index files in a separate area from the database cluster. Specify a space other than that under the database cluster.
Backup	Stores the data required for recovering the database when an error, such as disk failure, occurs.
Database backup	Contains the backup data for the database.
Archive log	Contains the log information for recovery.
Mirrored transaction log (mirrored WAL)	Enables a database cluster to be restored to the state immediately before an error even if both the database cluster and transaction log fail when performing backup/recovery operations using the pgx_dmpall command or WebAdmin. Mirrored WALs can be used only for backup/recovery using the pgx_dmpall command or WebAdmin.
Core file	Fujitsu Enterprise Postgres process core file that is output when an error occurs during a Fujitsu Enterprise Postgres process.
Key management server or key management storage	Server or storage where the master encryption key file is located.
Master encryption key file	Contains the master encryption key to be used when encrypting storage data. The master encryption key file is managed on the key management server or key management storage.

1.4.2 File Composition

Fujitsu Enterprise Postgres consists of the following files for controlling and storing the database. The table below shows the relationship between the number of such files and their location within a single instance.

Table 1.4 Number of files within a single instance and how to specify their location

File type	Required	Quantity	How to specify the location
Program files	Y	Multiple	Note that "<*x*>" indicates the product version. /opt/fsepv<*x*>server64
Database cluster	Y	1	Specify using WebAdmin or server commands.

File type	Required	Quantity	How to specify the location
Tablespace	Y	Multiple	Specify a space other than that under the database cluster, using the DDL statement.
Backup	Y	Multiple	Specify using WebAdmin or server commands.
Core file	Y	Multiple	Specify using WebAdmin, server commands, or postgresql.conf.
Server certificate file (*1)	N	1	Specify using postgresql.conf.
Server private key file (*1)	N	1	Specify using postgresql.conf.
Master encryption key file (*1)	N	1	Specify the directory created as the key store using postgresql.conf.
Connection service file (*1)	N	1	Specify using environment variables.
Password file (*1)	N	1	Specify using environment variables.
CA certificate file (*1)	N	1	Specify using environment variables.

Y: Mandatory
N: Optional

*1: Set manually when using the applicable feature.

Note

- Do not place files for use with Fujitsu Enterprise Postgres in a directory mounted over the network except when creating a database space in a storage device on a network.
  Examples include NFS (Network File System) and CIFS (Common Internet File System).
  This is because the database might hang if the network fails.

- If anti-virus software is used, set scan exception settings for directories so that none of the files that comprise Fujitsu Enterprise Postgres are scanned for viruses. Alternatively, if the files that comprise Fujitsu Enterprise Postgres are to be scanned for viruses, stop Fujitsu Enterprise Postgres and perform the scan when tasks that use Fujitsu Enterprise Postgres are not operating.

1.5 Notes on Compatibility of Applications Used for Operations

When you upgrade Fujitsu Enterprise Postgres to a newer version, there may be some effect on applications due to improvements or enhancements in functionality.

Take this into account when creating applications so that you can maintain compatibility after upgrading to a newer version of Fujitsu Enterprise Postgres.

See

Refer to "Notes on Application Compatibility" in the Application Development Guide for details.

Chapter 2 Starting an Instance and Creating a Database

This chapter describes basic operations, from starting an instance to creating a database.

2.1 Starting and Stopping an Instance

This section describes how to start and stop an instance.

- 2.1.1 Using WebAdmin
- 2.1.2 Using Server Commands

Point

- To automatically start or stop an instance when the operating system on the database server is started or stopped, refer to "Configuring Automatic Start and Stop of an Instance" in the Installation and Setup Guide for Server and configure the settings.
- The collected statistics are initialized if an instance is stopped in the "Immediate" mode or if it is abnormally terminated. To prepare for such initialization of statistics, consider regular collection of the statistics by using the SELECT statement. Refer to "The Statistics Collector" in "Server Administration" in the PostgreSQL Documentation for information on the statistics.

2.1.1 Using WebAdmin

WebAdmin enables you to start or stop an instance and check its operating status.

Starting an instance

Start an instance by using the [Instances] tab in WebAdmin.

▶ is displayed when an instance is stopped.

To start a stopped instance, click ▶.

Stopping an instance

Stop an instance by using the [Instances] tab in WebAdmin.

■ is displayed when an instance is active.

To stop an active instance, click ■.

Stop mode

Select the mode in which to stop the instance. The following describes the operations of the modes:

Stop mode	Connected clients	Backup being executed using the command
Smart mode (*1)	Waits for all connected clients to be disconnected.	Waits for backups being executed using the command to finish.
Fast mode	Rolls back all transactions being executed and forcibly disconnects clients.	Terminates backups being executed using the command.
Immediate mode	All server processes are terminated immediately. Crash recovery is executed the next time the instance is started.	
Kill process mode	Send SIGKILL to the process and abort all active transactions. This will lead to a crash-recovery run at the next restart.	

*1: When the processing to stop the instance in the Smart mode has started and you want to stop immediately, use the following procedure:

1. Restart the Web server feature of WebAdmin.

2. In the [Instances] tab, click ⟳.

3. In the [Instances] tab, click ⏹, and select the Immediate mode to stop the instance.

Checking the operating status of an instance

You can check the operating status of an instance by using the [Instances] tab. The following indicators are used to show the status of a resource.

Status indicator	Explanation
🟢	The resource is operating normally.
🔵	The resource is stopped.
🔴	There is an error in the resource.
🟠	An operation is in progress on this resource or the status is being checked.
⚠	The resource is not operating optimally and needs intervention.

If an instance stops abnormally, remove the cause of the stoppage and start the instance by using WebAdmin.

When operating WebAdmin, click ⟳ to update the status. WebAdmin will reflect the latest status of the operation or the instance resources from the server.

If an error occurs while communicating with the server, there may be no response from WebAdmin. When this happens, close the browser and then log in again. If this does not resolve the issue, check the system log of the server and confirm whether a communication error has occurred.

The following message is output during startup of an instance when the startup process is operating normally, therefore, the user does not need to be aware of this message:

```
FATAL:  the database system is starting up
```

2.1.2 Using Server Commands

Server commands enable you to start or stop an instance and check its operating status.

To use sever commands, configure the environment variables.

See

Refer to " Configure the environment variables" in the procedure to create instances in " Using the initdb Command" in the Installation and Setup Guide for Server for information on configuring the environment variables.

Starting an instance

Use the pg_ctl command to start an instance.

Specify the following values in the pg_ctl command:

- Specify "start" as the mode.

- Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

If an application, command, or process tries to connect to the database while the instance is starting up, the message "FATAL:the database system is starting up(11189)" is output. However, this message may also be output if the instance is started without the -W option specified.

This message is output by the pg_ctl command to check if the instance has started successfully. Therefore, ignore this message if there are no other applications, commands, or processes that connect to the database.

Example

```
> pg_ctl start -D /database/inst1
```

Note

If the -W option is specified, the command will return without waiting for the instance to start. Therefore, it may be unclear as to whether the instance startup was successful or failed.

Stopping an instance

Use the pg_ctl command to stop an instance.

Specify the following values in the pg_ctl command:

- Specify "stop" as the mode.
- Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

Example

```
> pg_ctl stop -D /database/inst1
```

Checking the operating status of an instance

Use the pg_ctl command to check the operating status of an instance.

Specify the following values in the pg_ctl command:

- Specify "status" as the mode.
- Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

Example

When the instance is active:

```
> pg_ctl status -D /database/inst1
pg_ctl: server is running (PID: 1234)
```

When the instance is inactive:

```
> pg_ctl status -D /database/inst1
pg_ctl: no server running.
```

See

Refer to "pg_ctl" under "Reference" in the PostgreSQL Documentation for information on pg_ctl command.

2.2 Creating a Database

This section explains how to create a database.

Follow the procedure below to define a database using client commands.

An example of operations on the server is shown below.

1. Use psql command to connect to the postgres database.
   Execute psql postgres.

   ```
   > psql postgres
   psql (<x>) (*1)
   Type "help" for help.
   ```

   *1: <*x*> indicates the PostgreSQL version on which this product is based.

2. Create the database.
   To create the database, execute the CREATE DATABASE databaseName; statement.

   ```
   postgres=# CREATE DATABASE db01;
   CREATE DATABASE
   ```

3. Confirm that the database is created.
   Execute \l+, and confirm that the name of the database created in step 2 is displayed.

   ```
   postgres=# \l+
   ```

4. Disconnect from the postgres database.
   Execute \q to terminate the psql command.

   ```
   postgres=# \q
   ```

You can create a database using the createdb command.

See

Refer to "Creating a Database" in "Tutorial" in the PostgreSQL Documentation for information on creating a database using the createdb command.

Chapter 3 Backing Up the Database

This chapter describes how to back up the database.

Backup methods

The following backup methods enable you to recover data to a backup point or to the state immediately preceding disk physical breakdown or data logical failure.

- Backup using WebAdmin

  This method enables you to back up data through intuitive window operations using the GUI.

  WebAdmin is used for recovery.

- Backup using the pgx_dmpall command

  Execute the pgx_dmpall command with a script to perform automatic backup.To back up data automatically, you must register the process in the automation software of the operating system. Follow the procedure given in the documentation for your operating system.

  Additionally, you can also take backups periodically by using any external scheduler.

  The pgx_rcvall command is used for recovery.

Use the selected backup method continuously.There are several differences, such as the data format, across the backup methods. For this reason, the following restrictions apply:

- It is not possible to use one method for backup and another for recovery.
- It is not possible to convert one type of backup data to a different type of backup data.

Information

By using a copy command created by the user, the pgx_dmpall command and the pgx_rcvall command can back up database clusters and tablespaces to any destination and recover them from any destination using any copy method. Refer to "Chapter 16 Backup/Recovery Using the Copy Command" for details.

Approximate backup time

The formula for deriving the approximate backup time when you use WebAdmin or the pgx_dmpall command is as follows:

```
backupTime = dataStorageDestinationUsage / diskWritePerformance x 1.5
```

- *dataStorageDestinationUsage*: Disk usage at the data storage destination
- *diskWritePerformance*: Maximum data volume (bytes/second) that can be written per second in the system environment where operation is performed
- 1.5: Coefficient to factor in tasks other than disk write (which is the most time-consuming step)

If using the copy command with the pgx_dmpall command, the backup time will depend on the implementation of the copy command.

When defining a tablespace

If you have defined a tablespace, back it up. If you do not back it up, directories for the tablespace are not created during recovery, which may cause the recovery to fail. If the recovery fails, refer to the system log, create the tablespace, and then perform the recovery process again.

When encrypting data stored in the database

There are several considerations for the backup of the keystore and backup of the database in case the data stored in the database is encrypted. Refer to the following for details:

- 5.6.4 Backing Up and Recovering the Keystore
- 5.7 Backing Up and Restoring/Recovering the Database

Information

The following methods can also be used to perform backup. Performing a backup using these methods allows you to restore to the point when the backup was performed.

- Backup using an SQL-based dump

  Dump the data by using SQL. This backup method also enables data migration.

- File system level backup

  This backup method requires you to stop the instance and use OS commands to backup database resources as files.

- Backup by continuous archiving

  This is the standard backup method for PostgreSQL.

Refer to "Backup and Restore" in "Server Administration" in the PostgreSQL Documentation for information on these backup methods.

The following backup methods are available for the features provided by Enterprise Postgres:

Category	Backup method	Backup target	Enterprise Postgres					
			Transparent Data Encryption	Policy-based Password Management	Data Masking	Confidentiality Management	Audit Log Feature	Database Multiplexing
Physical Backup	WebAdmin	Database cluster	Y	Y	Y	Y	Y	Y
	pgx_dmpall command		Y	Y	Y	Y	Y	Y
	Backup by continuous archiving		Y	Y	Y	Y	Y	Y
	File system level backup		Y	Y	Y	Y	Y	Y
Logical backup	pg_dumpall command		N	Y	N	Y	N	Y
	pg_dump command	Database	N	N	N	Y	N	Y
	COPY command	Table	N	N	N	N	N	N

Y: Can be used

N: Cannot be used

Some features are important to keep in mind when backing up. For information about the features, see the following:

Transparent Data Encryption : "5.7 Backing Up and Restoring/Recovering the Database"

Policy-based Password Management:"7.5 Backup and Recovery"

Confidentiality Management:"Backup/Restore" in the Security Operation Guide

Database Multiplexing:"Backup Operation" in the Cluster Operation Guide(Database Multiplexing)

3.1 Periodic Backup

It is recommended that you perform backup periodically.

Backing up data periodically using WebAdmin or the pgx_dmpall command has the following advantages:

- This method reduces disk usage, because obsolete archive logs (transaction logs copied to the backup data storage destination) are deleted. It also minimizes the recovery time when an error occurs.

Backup cycle

The time interval when backup is performed periodically is called the backup cycle. For example, if backup is performed every morning, the backup cycle is 1 day.
The backup cycle depends on the jobs being run, but on Fujitsu Enterprise Postgres it is recommended that operations are run with a backup cycle of at least once per day.

3.2 Backup Methods

This section describes the methods for backing up the database.

- 3.2.1 Using WebAdmin
- 3.2.2 Using Server Commands

3.2.1 Using WebAdmin

You can use WebAdmin to perform backup and check the backup status.

Backup operation

Follow the procedure below to back up the database.

1. Select the database to back up

   In the [Instances] tab, select the instance to be backed up and click .

2. Back up the database

   The [Backup] dialog box is displayed. To perform backup, click [Yes].
   An instance is automatically started when backup is performed.

Backup status

If an error occurs and backup fails, [Error] is displayed adjacent to [Data storage status] or [Backup storage status] in the [Instances] tab. An error message is also displayed in the message list.

In this case, the backup data is not optimized. Ensure that you check the backup result whenever you perform backup. If backup fails, [Solution] appears to the right of the error message. Clicking this button displays information explaining how to resolve the cause of the error. Remove the cause of failure, and perform backup again.

When encrypting data stored in the database

If the data to be stored in the database is to be encrypted, it is necessary to enable the automatic opening of the keystore before doing so. Refer to "5.6.3 Enabling Automatic Opening of the Keystore" for details.

3.2.2 Using Server Commands

Use the pgx_dmpall command and pgx_rcvall command to perform backup and check the backup result.

Preparing for backup

You must prepare for backup before actually starting the backup process.

Follow the procedure below.

See

Refer to "Preparing Directories to Deploy Resources" in the Installation and Setup Guide for Server for information on the location of directories required for backup and for points to take into account.

1. Prepare the backup data storage disk

   For backup, prepare a separate disk unit from the database storage disk and mount it using the operating system commands.

2. Create a directory where the backup data will be stored

   Create an empty directory.

   Set appropriate permissions so that only the instance administrator can access the directory.

   Example

   ```
   # mkdir /backup/inst1
   # chown fsepuser:fsepuser /backup/inst1
   # chmod 700 /backup/inst1
   ```

3. Specify the settings required for backup

   Stop the instance, and set the following parameters in the postgresql.conf file.

   Start the instance after editing the postgresql.conf file.

Parameter name	Setting	Description
backup_destination	Name of the directory where the backup data will be stored	Specify the name of the directory where the backup data will be stored. Appropriate privileges that allow only the instance administrator to access the directory must already be set. Place the backup data storage destination directory outside the data storage destination directory, the tablespace directory, and the transaction log storage destination directory.
archive_mode	on	Specify the archive log mode. Specify [on] (execute).
archive_command	'*installationDirectory*/bin/pgx_walcopy.cmd "%p" "*backupDataStorageDestinationDirectory*/archived_wal/%f"'	Specify the path name of the command that will save the transaction log and the storage destination.
archive_library	''(default)	Specify the archive library. Specify [''](default).

   Refer to "Appendix A Parameters" and "Write Ahead Log" under "Server Administration" in the PostgreSQL Documentation for information on the parameters.

Backup operation (file backup)

Use the pgx_dmpall command to perform file backup. You can even embed the pgx_dmpall command in OS automation software to perform backup.

The backup data is stored in the directory specified in the backup_destination parameter of postgresql.conf.

Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

Example

```
> pgx_dmpall -D /database/inst1
```

Note

Backup stores the data obtained during the backup and the backup data of the data obtained during previous backup.

If the data to be stored in the database is encrypted, refer to the following and back up the keystore:

- 5.6.4 Backing Up and Recovering the Keystore

Backup status

Use the pgx_rcvall command to check the backup status.

Specify the following values in the pgx_rcvall command:

- The -l option indicates backup data information.
- Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

```
> pgx_rcvall -l -D /database/inst1
Date                    Status          Dir
2022-03-01 13:30:40     COMPLETE        /backup/inst1/2022-03-01_13-30-40
```

If an error occurs and backup fails, a message is output to the system log.

In this case, the backup data is not optimized. Ensure that you check the backup result whenever you perform backup. If backup fails, remove the cause of failure and perform backup again.

See

Refer to "pgx_dmpall" and "pgx_rcvall" in the Reference for information on the pgx_dmpall command and pgx_rcvall command.

Setting a restore point

In case you want to recover your database to a certain point in time, you can name this particular point in time, which is referred to as the restore point, by using the psql command.

By setting a restore point before executing an application, it becomes easy to identify up to which point in time the data will be reverted.

A restore point can be set to any point in time after a backup is executed. However, if a restore point is set before a backup is executed, the database cannot be recovered to that point in time. This is because restore points are recorded in the archive logs, and the archive logs are discarded when backups are executed.

Example

The following example uses the psql command to connect to the database and execute the SQL statement to set a restore point.

However, when considering continued compatibility of applications, do not use functions directly in SQL statements. Refer to "Notes on Application Compatibility" in the Application Development Guide for details.

```
postgres=# SELECT pg_create_restore_point('batch_20220303_1');
LOG:  restore point "batch_20220303_1" created at 0/20000E8
```

```
STATEMENT:  select pg_create_restore_point('batch_20220303_1');
 pg_create_restore_point
-------------------------
 0/20000E8
(1 row)
```

Refer to "17.3.2 Using the pgx_rcvall Command" for information on using a restore point to recover the database.

Note

- Name restore points so that they are unique within the database. Add the date and time of setting a restore point to distinguish it from other restore points, as shown below:
    - YYMMDD_HHMMSS
        - YYMMDD: Indicates the date
        - HHMMSS: Indicates the time
- There is no way to check restore points you have set. Keep a record in, for example, a file.

See

Refer to "System Administration Functions" under "Functions and Operators" in the PostgreSQL Documentation for information on pg_create_restore_point.

Chapter 4 Configuring Secure Communication Using Secure Sockets Layer

If communication data transferred between a client and a server contains confidential information, encrypting the communication data can protect it against threats, such as eavesdropping on the network.

4.1 Configuring Communication Data Encryption

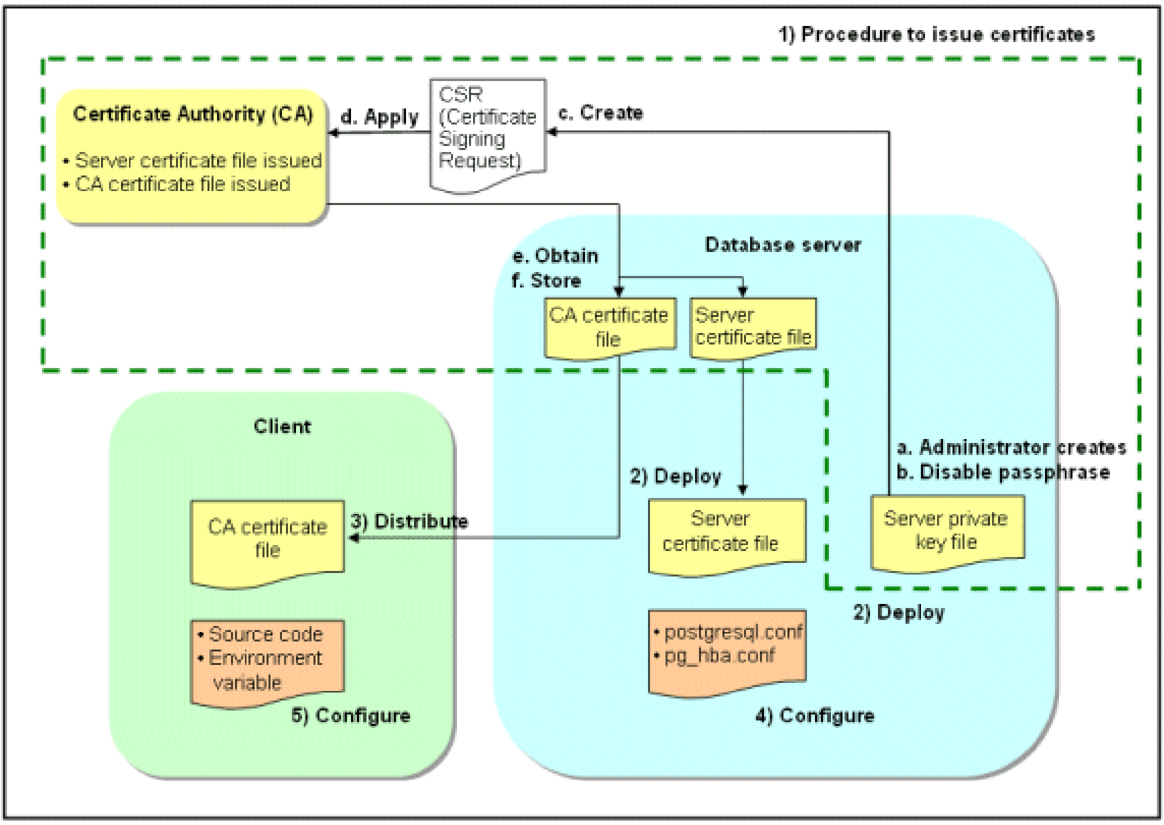
To encrypt communication data transferred between a client and a server, configure communication data encryption as described below. Communication data encryption not only protects the communication content, but it also guards against man-in-the-middle (MITM) attacks (for example, data and password theft through server impersonation).

Table 4.1 Configuration procedure

Configuration procedure
1) Issue a certificate
2) Deploy a server certificate file and a server private key file
3) Distribute a CA certificate file to the client
4) Configure the operating environment for the database server
5) Configure the operating environment for the client

The following figure illustrates the environment for communication data encryption.

Figure 4.1 Environment for communication data encryption



4.1.1 Issuing a Certificate

For authenticating servers, you must acquire a certificate issued by the certificate authority (CA).
Fujitsu Enterprise Postgres supports X.509 standard PEM format files. If the certificate authority issues a file in DER format, use a tool such as the openssl command to convert the DER format file to PEM format.

The following provides an overview of the procedure. Refer to the procedure published by the public or independent certificate authority (CA) that provides the certificate file for details.

a. Create a server private key file

b. Disable the passphrase for the server private key file

c. Create a CSR (signing request for obtaining a server certificate) from the server private key file

d. Apply to the certificate authority (CA) for a server certificate

e. Obtain a server certificate file and a CA certificate file from the certificate authority (CA)

f. Store the server certificate file and the CA certificate file
   Note: If you lose or destroy the certificates, you will need to have them re-issued.

The above procedure enables you to prepare the following files:

- Server private key file
- Server certificate file
- CA certificate file

4.1.2 Deploying a Server Certificate File and a Server Private Key File

Create a directory on the local disk of the database server and store the server certificate file and the server private key file in it.
Use the operating system features to set access privileges for the server certificate file and the server private key file so that only the database administrator has load privileges.
Back up the server certificate file and the server private key file in the event that data corruption occurs and store them securely.

4.1.3 Distributing a CA Certificate File to the Client

Create a directory on the local disk of the client and place the distributed CA certificate file there. Use the operating system features to set load privileges to protect the CA certificate file against accidental deletion.

4.1.4 Configuring the Operating Environment for the Database Server

See

Refer to "Secure TCP/IP Connections with SSL" under "Server Administration" in the PostgreSQL Documentation for details.

4.1.5 Configuring the Operating Environment for the Client

See

Refer to the following sections in the Application Development Guide for details, depending on your application development environment:

- "Settings for Encrypting Communication Data" under "Setup" in "JDBC Driver"
- "Settings for Encrypting Communication Data" under "Setup" in "C Library (libpq)"
- "Settings for Encrypting Communication Data" under "Setup" in "Embedded SQL in C"

4.1.6 Performing Database Multiplexing

When you perform communication that uses database multiplexing and a Secure Socket Layer server certificate, take one of the following actions:

- Create one server certificate, replicate it, and place a copy on each server used for database multiplexing.
  If sslmode is set to verify-full, add all domain names in subjectAltName.
- Create server certificate for each server used for database multiplexing.

See

Refer to "Using the Application Connection Switch Feature" in the Application Development Guide for information on how to specify applications on the client.

Chapter 5 Protecting Storage Data Using Transparent Data Encryption

This chapter describes how to encrypt data to be stored in the database.

See

If you want to use an external key management system as the storage location for the encryption key, refer to "Chapter 6 Using Transparent Data Encryption with Key Management Systems as Keystores".

5.1 Protecting Data Using Encryption

With PostgreSQL, data in a database is protected from access by unauthorized database users through the use of authentication and access controls. However, the OS file is not protected from attackers who bypass the database server's authentication and access controls.

With Fujitsu Enterprise Postgres, data inside the OS file is encrypted, so valuable information is protected even if the file or disk is stolen.

Data to be stored in a database is encrypted when it is written to the data file, and decrypted when it is read.

This is performed automatically by the instance, so the user and the application need not be aware of key management and encryption or decryption. This process is called TDE (Transparent Data Encryption).

The characteristics of TDE are described below.

Encryption mechanisms

Two-layer encryption key and the keystore

In each tablespace, there is a tablespace encryption key that encrypts and decrypts all the data within. The tablespace encryption key is encrypted by the master encryption key and saved.
Only one master encryption key exists in a database cluster. It is encrypted based on a passphrase specified by the user and stored in a keystore. Fujitsu Enterprise Postgres provides a file-based keystore. Attackers who do not know the passphrase cannot read the master encryption key from the keystore.

Strong encryption algorithms

TDE uses the Advanced Encryption Standard (AES) as its encryption algorithm. AES was adopted as a standard in 2002 by the United States Federal Government, and is used throughout the world.

Zero overhead storage areas

Encryption does not change the size of data stored in tables, indexes, or WAL. There is, therefore, no need for additional estimates or disks.

Scope of encryption

All user data within the specified tablespace

The tablespace is the unit for specifying encryption. All tables, indexes, temporary tables, and temporary indexes created in the encrypted tablespace are encrypted. There is no need for the user to consider which tables and strings to encrypt.

Refer to "5.4 Encrypting a Tablespace" for details.

Backup data

The pgx_dmpall command and pg_basebackup command create backup data by copying the OS file. Backups of the encrypted data are, therefore, also encrypted. Information is protected from leakage even if the backup medium is stolen.

Refer to "5.7 Backing Up and Restoring/Recovering the Database" after enabling or reconfiguring encryption to back up the database.

WAL and temporary files

WAL, which is created by updating encrypted tables and indexes, is encrypted with the same security strength as the update target. When large merges and sorts are performed, the encrypted data is written to a temporary file in encrypted format.

Streaming replication support

You can combine streaming replication and transparent data encryption. The data and WAL encrypted on the primary server is transferred to the standby server in its encrypted format and stored.

Note

The following are not encrypted:

- pg_dump and pg_dumpall output files
- Files output by the COPY command
- Notification event payloads that communicate using the LISTEN or NOTIFY command
- Checksum validation is not performed on encrypted tablespaces during backup and when using the pg_checksum utility.

5.2 Setting the Master Encryption Key

To use transparent data encryption, you must create a keystore and set the master encryption key.

1. In the keystore_location parameter of postgresql.conf, specify the directory to store the keystore.

   Specify a different location for each database cluster. Specify a different directory from those below as the keystore storage destination:

   - Data storage destination
   - Tablespace storage destination
   - Transaction log storage destination
   - Backup data storage destination

   ```
   keystore_location = '/key/store/location'
   ```

   Refer to "Appendix A Parameters" for information on postgresql.conf.

   After editing the postgresql.conf file, either start or restart the instance.

   - Using WebAdmin

     Refer to "2.1.1 Using WebAdmin", and restart the instance.

   - Using the pg_ctl command

     Specify the following in the pg_ctl command:

     - Specify "restart" as the mode.
     - Specify the data storage destination directory in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.
     - Specify the -w option. This means that the command returns after waiting for the instance to start. If the -w option is not specified, it may not be possible to determine if the starting of the instance completed successfully or if it failed.

     Example

     ```
     > pg_ctl restart -w -D /database/inst1
     ```

2. Execute an SQL function, such as the one below, to set the master encryption key. This must be performed by the superuser. Execute it as the database superuser.

   ```
   SELECT pgx_set_master_key('passphrase');
   ```

   The value "passphrase" is the passphrase that will be used to open the keystore. The master encryption key is protected by this passphrase, so avoid specifying a short simple string that is easy to guess.

Refer to "B.2 Transparent Data Encryption Control Functions" for information on the pgx_set_master_key function.

Note

Note that if you forget the passphrase, you will not be able to access the encrypted data. There is no method to retrieve a forgotten passphrase and decrypt data. Do not, under any circumstances, forget the passphrase.

The pgx_set_master_key function creates a file with the name keystore.ks in the keystore storage destination. It also creates a master encryption key from random bit strings, encrypts it with the specified passphrase, and stores it in keystore.ks. At this point, the keystore is open.

5.3 Opening the Keystore

To create encrypted tablespaces and access the encrypted data, you must first open the keystore. When you open the keystore, the master encryption key is loaded into the database server memory and becomes usable for encryption and decryption.

You need to open the keystore each time you start the instance. To open the keystore, the database superuser must execute the following SQL function.

```
SELECT pgx_open_keystore('passphrase');
```

The value "passphrase" is the passphrase specified during creation of the keystore.

Refer to "B.2 Transparent Data Encryption Control Functions" for information on the pgx_open_keystore function.

Note that, in the following cases, the passphrase must be entered when starting the instance, because the encrypted WAL must be decrypted for recovery. In this case, the above-mentioned pgx_open_keystore function cannot be executed.

- If performing crash recovery at the time of starting the instance
- If performing recovery using continuous archiving

For the above cases, specify the --keystore-passphrase option in the pg_ctl command, and then start the instance. This will display the prompt for the passphrase to be entered, as shown below.

```
> pg_ctl --keystore-passphrase start
Enter the passphrase:
The server is starting
>
```

Point

When using an automatically opening keystore, you do not need to enter the passphrase and you can automatically open the keystore when the database server starts. Refer to "5.6.3 Enabling Automatic Opening of the Keystore" for details.

5.4 Encrypting a Tablespace

The keystore must be open before you can create an encrypted tablespace.

When creating a tablespace that will be encrypted, configure the encryption algorithm in the runtime parameters. For example, to create a tablespace with the name secure_tablespace using AES with a key length of 256 bits as the encryption algorithm, configure as shown below.

```
-- Specify the encryption algorithm for the tablespace to be created below
SET tablespace_encryption_algorithm = 'AES256';
CREATE TABLESPACE secure_tablespace LOCATION '/My/Data/Dir';
-- Specify that the tablespace to be created below is not to be encrypted
SET tablespace_encryption_algorithm = 'none';
```

Or

```
CREATE TABLESPACE secure_tablespace LOCATION '/My/Data/Dir' WITH (tablespace_encryption_algorithm =
'AES256' );
```

When the tablespace is empty, the encryption algorithm can be modified with the command below.

```
ALTER TABLESPACE secure_tablespace SET (tablespace_encryption_algorithm=AES256);
```

Trying to set the encryption algorithm for a non-empty tablespace causes an error.

You can use AES with a key length of 128 bits or 256 bits as the encryption algorithm. It is recommended that you use 256-bit AES. Refer to "Appendix A Parameters" for information on how to specify the runtime parameters.

If user provides both GUC and command line options while creating the tablespace, the preference is given to the command line option.

The pg_default and pg_global tablespaces cannot be encrypted.

Create tables and indexes in the encrypted tablespace that you created. Relations created in the encrypted tablespace are automatically encrypted.

Example

Example 1: Specifying an encrypted tablespace when creating it

```
CREATE TABLE my_table (...)
    TABLESPACE secure_tablespace;
```

Example 2: Not explicitly specifying a tablespace when creating it and instead using the default tablespace

```
SET default_tablespace = 'secure_tablespace';
CREATE TABLE my_table (...);
```

The process is the same for encrypting temporary tables and temporary indexes. In other words, either explicitly specify the TABLESPACE clause or list encrypted tablespaces in the temp_tablespaces parameter, and then execute CREATE TEMPORARY TABLE or CREATE INDEX.

Point

If an encrypted tablespace is specified in the TABLESPACE clause of the CREATE DATABASE statement, relations created in the database without explicitly specifying a tablespace will be encrypted. Furthermore, the system catalog will also be encrypted, so the source code of user-defined functions is also protected.

Example: Specifying a tablespace in a database definition statement

```
CREATE DATABASE DB01 TABLESPACE=SP01 ... ;
```

Part of the data is also stored in the system catalog - to encrypt this data as well, specify an encrypted tablespace as above and create a database.

5.5 Checking an Encrypted Tablespace

The pgx_tablespaces system view displays information about whether each tablespace has been encrypted, and about the encryption algorithm. Refer to "D.1 pgx_tablespaces" for information on strings.

You can discover which tablespaces have been encrypted by executing the following SQL statements.

However, when considering continued compatibility of applications, do not reference system catalogs (pg_tablespace) directly in SQL statements.

```
SELECT spcname, spcencalgo
FROM pg_tablespace ts, pgx_tablespaces tsx
WHERE ts.oid = tsx.spctablespace;
```

Example

```
postgres=# SELECT spcname, spcencalgo FROM pg_tablespace ts, pgx_tablespaces tsx WHERE ts.oid =
tsx.spctablespace;
      spcname      | spcencalgo
-------------------+------------
 pg_default        | none
 pg_global         | none
 secure_tablespace | AES256
(3 rows)
```

See

Refer to "Notes on Application Compatibility" in the Application Development Guide for information on how to maintain application compatibility.

5.6 Managing the Keystore

This section describes how to manage the keystore and the master encryption key to guard against the threat of theft.

5.6.1 Changing the Master Encryption Key

Using the same encryption key for an extended period gives attackers an opportunity to decipher the encrypted data. It is recommended that you change the key at regular intervals, or whenever the key is exposed to risk.

Adhere to the industry's best practices for encryption algorithms and key management when considering how often the key should be changed. For example, the NIST in the United States has published "NIST Special Publication 800-57". The PCI DSS also refers to this publication. This publication recommends changing the master encryption key once a year.

To change the master encryption key, execute the pgx_set_master_key function, which is the same function used for configuring the key. Refer to "5.2 Setting the Master Encryption Key" for details.

After changing the master encryption key, you must immediately back up the keystore.

5.6.2 Changing the Keystore Passphrase

In security policies for organizations, it is usually a requirement that the passphrase be changed whenever a security administrator who knows the passphrase is removed from duties due to transfer or retirement. It is also recommended that the passphrase be changed if it is ever exposed to risks due to deception such as social engineering.

To change the keystore passphrase, execute the following SQL function as a superuser.

```
SELECT pgx_set_keystore_passphrase('oldPassphrase', 'newPassphrase');
```

After changing the passphrase, you must immediately back up the keystore.

Refer to "B.2 Transparent Data Encryption Control Functions" for information on the pgx_set_keystore_passphrase function.

5.6.3 Enabling Automatic Opening of the Keystore

When using an automatically opening keystore, you do not need to enter the passphrase and you can automatically open the keystore when the instance starts. Execute the pgx_keystore command to enable automatic opening of the keystore.

```
> pgx_keystore --enable-auto-open /key/store/location/keystore.ks
Enter the passphrase:
Automatic opening of the keystore is now enabled
>
```

See

Refer to "pgx_keystore" in the Reference for information on pgx_keystore command.

When automatic opening is enabled, an automatically opening keystore is created in the same directory as the original keystore. The file name of the automatically opening keystore is keystore.aks. The file keystore.aks is an obfuscated copy of the decrypted content of the keystore.ks file. As long as this file exists, there is no need to enter the passphrase to open the keystore when starting the instance.

Do not delete the original keystore file, keystore.ks. It is required for changing the master encryption key and the passphrase. When you change the master encryption key and the passphrase, keystore.aks is recreated from the original keystore file, keystore.ks.

Protect keystore.ks, keystore.aks, and the directory that stores the keystore so that only the user who starts the instance can access them.

Configure the permission of the files so that only the user who starts the instance can access the SQL functions and commands that create these files. Accordingly, manually configure the same permission mode if the files are restored.

Example

```
# chown -R fsepuser:fsepuser /key/store/location
# chmod 700 /key/store/location
# chmod 600 /key/store/location/keystore.ks
# chmod 600 /key/store/location/keystore.aks
```

To use WebAdmin for backup and recovery, you must enable automatic opening of the keystore.

An automatically opening keystore will only open on the computer where it was created.

To disable automatic opening of the keystore, delete keystore.aks.

5.6.4 Backing Up and Recovering the Keystore

Back up the keystore at the following times in case it is corrupted or lost. Note that you must store the database and the keystore on separate data storage media. Storing both on the same data storage medium risks the danger of the encrypted data being deciphered if the medium is stolen. A passphrase is not required to open an automatically opening keystore, so store this type of keystore in a safe location.

- When the master encryption key is first configured
- When the master encryption key is changed
- When the database is backed up
- When the keystore passphrase is changed

Point

Do not overwrite an old keystore when backing up a keystore. This is because during database recovery, you must restore the keystore to its state at the time of database backup. When the backup data of the database is no longer required, delete the corresponding keystore.

Example

- Back up the database and the keystore on March 1, 2022.

  ```
  > pgx_dmpall -D /database/inst1
  > cp -p /key/store/location/keystore.ks /keybackup/keystore_20220301.ks
  ```

  Specify the following in the pgx_dmpall command:

  - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

- Change the master encryption key, and back up the keystore on March 5, 2022.

  ```
  > psql -c "SELECT pgx_set_master_key('passphrase')" postgres
  > cp -p /key/store/location/keystore.ks /keybackup/keystore_20220305.ks
  ```

  Specify the following in the psql command:

  - Specify the SQL function that sets the master encryption key in the -c option.
  - Specify the name of the database to be connected to as the argument.

If the keystore is corrupted or lost, restore the keystore containing the latest master encryption key. If there is no keystore containing the latest master encryption key, restore the keystore to its state at the time of database backup, and recover the database from the database backup. This action recovers the keystore to its latest state.

Example

- Restore the keystore containing the latest master encryption key as of March 5, 2022.

  ```
  > cp -p /keybackup/keystore_20220305.ks /key/store/location/keystore.ks
  ```

- If there is no backup of the keystore containing the latest master encryption key, recover the keystore by restoring the keystore that was backed up along with the database on 1 March 2022.

  ```
  > cp -p /keybackup/keystore_20220301.ks /key/store/location/keystore.ks
  > pgx_rcvall -B /backup/inst1 -D /database/inst1 --keystore-passphrase
  ```

  Specify the following in the pgx_rcvall command:

  - Specify the data storage directory in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.
  - Specify the backup data storage directory in the -B option.
  - The --keystore-passphrase option prompts you to enter the passphrase to open the keystore.

If you have restored the keystore, repeat the process of enabling automatic opening of the keystore. This ensures that the contents of the automatically opening keystore (keystore.aks) are identical to the contents of the restored keystore.

It is recommended that you do not back up the automatically opening keystore file, keystore.aks. If the database backup medium and the backup medium storing the automatically opening keystore are both stolen, the attacker will be able to read the data even without knowing the passphrase.

If the automatically opening keystore is corrupted or lost, you must again enable automatic opening. The keystore.aks file will be recreated from keystore.ks at this time.

See

Refer to "pgx_rcvall" and "pgx_dmpall" in the Reference for information on the pgx_rcvall and pgx_dmpall commands.

Refer to "psql" under "Reference" in the PostgreSQL Documentation for information on the psql command.

Refer to "B.2 Transparent Data Encryption Control Functions" for information on the pgx_set_master_key function.

Refer to "5.6.3 Enabling Automatic Opening of the Keystore" for information on how to enable automatic opening of the keystore.

5.7 Backing Up and Restoring/Recovering the Database

Fujitsu Enterprise Postgres enables you to use the five backup and recovery methods described below. Regardless of the method you use, you must back up the keystore at the same time.

Note that you must store the database and the keystore on separate data storage media. Storing both on the same data storage medium risks the danger of the encrypted data being deciphered if the medium is stolen.

Backup and recovery using WebAdmin

- Backup

  WebAdmin backs up encrypted data.

  Back up the key store after backing up the database.

- Recovery

  Restore the keystore to its state at the time of database backup. Refer to "5.6.4 Backing Up and Recovering the Keystore" for details.

  Enable automatic opening of the keystore in accordance with the procedure described in "5.6.3 Enabling Automatic Opening of the Keystore". Then, use WebAdmin to recover the database.

Backup and recovery using the pgx_dmpall and pgx_rcvall commands

- Backup

  The pgx_dmpall command backs up the encrypted data.

  Back up the key store after backing up the database.

- Recovery

  Restore the keystore to its state at the time of the database backup.

  Configure automatic opening of the key store as necessary.

  If automatic opening of the keystore is not enabled, execute the pgx_rcvall command with the --keystore-passphrase option specified. This will display the prompt for the passphrase to be entered.

Example

- Back up the database and the keystore on March 1, 2022.

```
> pgx_dmpall -D /database/inst1
> cp -p /key/store/location/keystore.ks /keybackup/keystore_20220301.ks
```

  Specify the following in the pgx_dmpall command:

  - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

- Recover the database and the keystore from the backup taken on March 1, 2022.

```
> cp -p /keybackup/keystore_20220301.ks /key/store/location/keystore.ks
> pgx_keystore --enable-auto-open /key/store/location/keystore.ks  (Execute only when enabling
automatic opening)
> pgx_rcvall -B /backup/inst1 -D /database/inst1 --keystore-passphrase
```

  Specify the following in the pgx_rcvall command:

  - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

  - Specify the backup data storage directory in the -B option.

  - The --keystore-passphrase option prompts you to enter the passphrase to open the keystore.

Dump and restore using SQL

- Backup

  The files output by the pg_dump and pg_dumpall commands are not encrypted. You should, therefore, encrypt the files using OpenSSL commands or other means before saving them, as described in "5.8 Importing and Exporting the Database" below.

  Back up the key store after backing up the database.

- Restore

  If the backup data has been encrypted using, for example Open SSL commands, decrypt that data.

  The data generated by the pg_dumpall command includes a specification to encrypt tablespaces by default. For this reason, the psql command encrypts tablespaces during restoration.

File system level backup and restore

- Backup

  Stop the instance and backup the data directory and the tablespace directory using the file copy command of the operating system. The files of encrypted tablespaces are backed up in the encrypted state.

  Back up the key store after performing the backup.

- Restore

  Restore the keystore to its state at the time of the database backup.

  Stop the instance and restore the data directory and the tablespace directory using the file copy command of the operating system.

Continuous archiving and point-in-time recovery

- Backup

  The pg_basebackup command backs up the encrypted data as is.

  Back up the key store after performing the backup.

- Recovery

  Restore the keystore to its state at the time of the database backup.

  Configure automatic opening of the key store as necessary.

  If automatic opening of the keystore is not enabled, execute the pg_ctl command to start the instance with the --keystore-passphrase option specified. This will display the prompt for the passphrase to be entered.

See

- Refer to "pg_ctl" under "Reference" in the PostgreSQL Documentation for information on the pg_ctl command.
- Refer to "Reference" in the PostgreSQL Documentation for information on the following commands:
  - psql
  - pg_dump
  - pg_basebackup
- Refer to the Reference for information on the following commands:
  - pgx_rcvall
  - pgx_dmpall
  - pg_dumpall

If you have restored the keystore, repeat the process of enabling automatic opening of the keystore This ensures that the contents of the automatically opening keystore (keystore.aks) are identical to the contents of the restored keystore.

Refer to "5.6.3 Enabling Automatic Opening of the Keystore" for information on how to enable automatic opening of the keystore.

5.8 Importing and Exporting the Database

The files output by the COPY TO command are not encrypted. Therefore, when transferring files to other systems, you should encrypt files using OpenSSL commands or other means and use scp or sftp to encrypt the data being transferred.

Use a safe method to delete obsolete plain text files.

You can use the following methods to safely delete files:

- shred command

Example

```
# Export the contents of the table my_table to a CSV file.
> psql -c "COPY my_table TO '/tmp/my_table.csv' (FORMAT CSV)" postgres

# Encrypt the exported file.
> openssl enc -e -aes256 -in my_table.csv -out my_table.csv.enc
(The user is prompted to enter the passphrase to be used for encryption)

# Safely delete plain text files.
> shred -u -x my_table.csv
 (Transfer encrypted files to other systems)

# Decrypt the encrypted files on other systems.
> openssl enc -d -aes256 -in my_table.csv.enc -out my_table.csv
(The user is prompted to enter the passphrase to be used for decryption)
```

If you use COPY FROM to import data to tables and indexes in an encrypted tablespace, the imported data is automatically encrypted before being stored.

5.9 Encrypting Existing Data

You cannot encrypt existing unencrypted tablespaces. In addition, you cannot change encrypted tablespaces so that they do not encrypt.

As an alternative, transfer the tables and indexes to other tablespaces. You can use the following SQL commands for this.

```
ALTER TABLE table_name SET TABLESPACE new_tablespace;
ALTER INDEX index_name SET TABLESPACE new_tablespace;
ALTER DATABASE database_name SET TABLESPACE new_tablespace;
```

See

Refer to "SQL Commands" under "Reference" in the PostgreSQL Documentation for information on SQL commands.

5.10 Operations in Cluster Systems

This section describes how to use transparent data encryption on cluster systems such as high-availability systems, streaming replication, and database multiplexing.

5.10.1 HA Clusters that do not Use Database Multiplexing

Take the following points into account when using transparent data encryption in an HA cluster environment that does not use database multiplexing.

Placement and automatic opening of the keystore file

There are two alternatives for placing the keystore file:

- Sharing the keystore file
- Placing a copy of the keystore file

Sharing the keystore file

This involves using the same keystore file on the primary server and the standby server.

As the standby server is not active while the primary server is running, this file would not be accessed simultaneously, and therefore, it can be shared.

To manage the keystore file in a more secure manner, place it on the key management server or the key management storage isolated in a secure location.

Enable the automatic opening of the keystore on both the primary and standby servers.

Placing a copy of the keystore file

This involves placing a copy of the primary server keystore file on the standby server.

You can do this if you cannot prepare a shared server or disk device that can be accessed from both the primary and standby servers.

However, if you change the master encryption key and the passphrase on the primary server, you must copy the keystore file to the standby server again.

To manage the keystore file in a more secure manner, prepare the key management server or the key management storage isolated in a secure location for both the primary and standby servers, and place the keystore files there.

Enable the automatic opening of the keystore on both the primary and standby servers. Note that copying the automatically opening keystore file (keystore.aks) to the standby server does not enable the automatic opening of the keystore.

5.10.2 Database Multiplexing Mode

Note the following when using transparent data encryption in environments that use streaming replication, or database multiplexing with streaming replication.

Placing the keystore file

Place a copy of the primary server keystore file on the standby server.

This is required as the keystore file cannot be shared, and both servers may need to access it simultaneously.

Point

To manage the keystore file in a more secure manner, place it on the key management server or the key management storage isolated in a secure location. A keystore used by both the primary and standby servers can be managed on the same key management server or key management storage.

However, create different directories for the keystores to be used by the primary server and the standby server. Then copy the keystore for the primary server to the directory used on the standby server.

Automatically opening the keystore

You must enable automatic opening of the keystore.

To do this, enable automatic opening of the keystore in all servers that make up database multiplexing. The settings for automatic opening of the keystore include information unique to each server, so simply copying the file does not enable it.

Changing the passphrase

Changes to the passphrase are reflected in all servers that make up database multiplexing, so no special operation is required.

Building and starting a standby server

Before using the pg_basebackup command or pgx_rcvall command to build a standby server, copy the keystore file from the primary server to the standby server. When using an automatically opening keystore, use the copied keystore file to enable automatic opening on the standby server.

Open the keystore each time you start the standby server. This step is necessary for decrypting and restoring encrypted WAL received from the primary server. To open the keystore, specify the --keystore-passphrase option in the pg_ctl command or pgx_rcvall command and enter the passphrase, or use an automatically opening keystore.

Changing the master encryption key and the passphrase

Change the master encryption key and the passphrase on the primary server. You need not copy the keystore from the primary server to the standby server. You need not even restart the standby server or reopen the keystore. Changes to the master encryption key and the passphrase are reflected in the keystore on the standby server.

See

Refer to "pgx_rcvall " in the Reference for information on pgx_rcvall command.

Refer to "pg_ctl" under "Reference" in the PostgreSQL Documentation for information on pg_ctl command.

Refer to "pg_basebackup" under "Reference" in the PostgreSQL Documentation for information on pg_basebackup command.

Refer to "High Availability, Load Balancing, and Replication" under "Server Administration" in the PostgreSQL Documentation for information on how to set up streaming replication.

5.11 Security-Related Notes

- Decrypted data is cached in the database server memory (shared buffer). As a result, unencrypted data is stored in a core file, which is a process memory dump. You should, therefore, safely delete the memory dump.
  You can safely delete files by using the following command:
  - shred command
- Unencrypted data may be written from the database server memory to the operating system's swap area. To prevent leakage of information from the swap area, consider either disabling the use of swap area or encrypting the swap area using a full-disk encryption product.
- The content of the server log file is not encrypted. Therefore, in some cases the value of a constant specified in a SQL statement is output to the server log file. To prevent this, consider setting a parameter such as log_min_error_statement.
- When executing an SQL function that opens the keystore and modifies the master encryption key, ensure that the SQL statement containing the passphrase is not output to the server log file. To prevent this, consider setting a parameter such as log_min_error_statement. If you are executing this type of SQL function on a different computer from the database server, encrypt the communication between the client and the database server with SSL.
- Starting with FEP 10, logical replication is available, which allows non-backed up clusters to subscribe to databases where transparent data encryption is enabled. Logical replication does not need to have the same encryption strategy between publisher and subscriber.

  In this scenario, if the user wants to encrypt the subscribed copy of data as well, then it is the user's responsibility to create encryption policies to the subscribed databases. By default, published encrypted tablespace data will not be encrypted in the subscriber side.

5.12 Tips for Installing Built Applications

With transparent data encryption, you can easily encrypt all the data in an application without modifying the application. Database administrators install built applications in the following manner. However, this procedure stores data to the default tablespace, so take necessary action if processing differs from the original design.

1. (Normal procedure) Create an owner and a database for the built application.

```
CREATE USER crm_admin ...;
CREATE DATABASE crm_db ...;
```

2. (Procedure for encryption) Create an encrypted tablespace to store the data for the built application.

```
SET tablespace_encryption_algorithm = 'AES256';
CREATE TABLESPACE crm_tablespace LOCATION '/crm/data';
```

3. (Procedure for encryption) Configure an encrypted tablespace as the default tablespace for the owner of the built application.

```
ALTER USER crm_admin SET default_tablespace = 'crm_tablespace';
ALTER USER crm_admin SET temp_tablespaces = 'crm_tablespace';
```

4. (Normal procedure) Install the built application. The application installer prompts you to enter the host name and the port number of the database server, the user name, and the database name. The installer uses the entered information to connect to the database server and execute the SQL script. For applications that do not have an installer, the database administrator must manually execute the SQL script.

Normally, the application's SQL script includes logic definition SQL statements, such as CREATE TABLE, CREATE INDEX, and GRANT or REVOKE, converted from the entity-relationship diagram. It does not include SQL statements that create databases, users, and tablespaces. Configuring the default tablespace of the users who will execute the SQL script deploys the objects generated by the SQL script to the tablespace.

Chapter 6 Using Transparent Data Encryption with Key Management Systems as Keystores

This chapter describes the operation of transparent data encryption when a key management system is used as a keystore.

See

Refer to "Key Management System Requirements" in the Installation and Setup Guide for Server for the key management system requirements that can be used with Fujitsu Enterprise Postgres.

6.1 Protecting Data Using Encryption

Refer to "5.1 Protecting Data Using Encryption". The following describes the differences from the transparent data encryption operation in the file-based keystore described in "5.1 Protecting Data Using Encryption".

Encryption mechanisms

Two-layer encryption key and the keystore

Each tablespace has a tablespace encryption key that encrypts/decrypts all data in it. Tablespace encryption keys are stored encrypted with the master encryption key.
Use an encryption key stored in a key management system as a common master encryption key for your database cluster. Fujitsu Enterprise Postgres refers to the key management system as a keystore for master encryption keys.

Type of key management system

Two types of key management systems are available:

- kmip

  It is a key management system that can be used using a protocol called KMIP (Key Management Interoperability Protocol) standardized by OASIS (Organization for the Advancement of Structured Information Standards).

- custom

  It is a key management system that cooperates using an adapter that converts the request format without adopting the KMIP protocol.

See

Refer to "Key Management System Requirements" in the Installation and Setup Guide for Server for the key management system requirements that can be used with Fujitsu Enterprise Postgres.

Sharing tablespace encryption keys

When using an adapter to link with a key management system, encryption and decryption of the tablespace encryption key using the master encryption key are performed on the key management system side.

Tablespace encryption keys can be shared within a database cluster so that you do not need to access the key management system each time you want to use the tablespace encryption key.

The cost of encryption/decryption using the master encryption key becomes an issue in the following cases:

- Multiple connections to the database access encrypted tablespaces
- Connections accessing encrypted tablespaces are repeated and connection pooling is disabled

Encryption key identifier

Key IDs are used as identifiers to identify encryption keys stored on the key management system.

Key ID

Information that identifies the encryption key on the key management system, and is unique within the key management system. The correspondence between the encryption key substance (byte string) and the key ID does not change throughout the life cycle of the encryption key.

The name of the identifier differs depending on each key management system, but in this feature, such information is called the key ID.

Changes to the key management system

After starting operation of the transparent data encryption function, the key management system to be used can be changed to another key management system.

6.2 Setting the Master Encryption Key

To use transparent data encryption, you must create a keystore and set the master encryption key.

1. Load the shared_preload_libraries parameter in postgresql.conf with the library name "tde _ kms"

   ```
   shared_preload_libraries = 'tde_kms'
   ```

2. When using an adapter, register the adapter as a plug-in. Specify the directory where the plugin is stored in the tde_kms.plugin_path parameter in postgresql.conf. Store your plugins in this directory.

   ```
   tde_kms.plugin_path = '/home/fsepuser/plugin/'
   ```

3. To share the tablespace encryption key, set the tde_kms.enable_shared_dek parameter in postgresql.conf to "on".

   ```
   tde_kms.enable_shared_dek = on
   ```

4. Set the tde_kms.kms_conninfo_file parameter in postgresql.conf to a file that contains key management system connection information. Refer to "Appendix A Parameters" for information.

   #### Example for the key management system connection information file kms_conninfo.conf

   ```
   tde_kms.kms_conninfo_file = 'kms_conninfo.conf'
   ```

   #### Example of key management system connection information file

   For type kmip

   ```
   kmip   mykmipsvr  mykmipsvr.example.com   5696   cert   sslcert=postgres.crt
   sslkey=postgres.key sslrootcert=root.crt
   ```

   For type custom

   ```
   custom   mykms   mykms  arg=--profile arg=user1
   ```

5. Execute a CREATE EXTENSION statement to install the extension.

   ```
   CREATE EXTENSION tde_kms;
   ```

6. To enable transparent data encryption, call the pgx_declare_external_master_key function to declare the encryption key to use as the master encryption key. Specify a key ID as an identifier to identify the encryption key. Refer to "B.2.3 pgx_declare_external_master_key" for information on the pgx_declare_external_master_key function.

   ```
   SELECT pgx_declare_external_master_key( kms_name => 'mykmipsvr', key_id =>
   'a0eebc99-9c0b-0000-0000-000000000000', sslpassphrase => 'mykmippassphrase' );
   ```

6.3 Opening the Keystore

To create encrypted tablespaces and access the encrypted data, you must first open the keystore. When you open the keystore, the master encryption key is available for encryption and decryption.

You need to open the keystore each time you start the instance. To open the keystore, the database superuser must execute the following SQL function.

```
SELECT pgx_open_keystore( sslpassphrase => 'passphrase');
```

passphrase is the passphrase of the private key file for the client certificate.

Refer to "B.2 Transparent Data Encryption Control Functions" for information on the pgx_open_keystore function.

Note that, in the following cases, the passphrase must be entered when starting the instance, because the encrypted WAL must be decrypted for recovery. In this case, the above-mentioned pgx_open_keystore function cannot be executed.

- If performing crash recovery at the time of starting the instance
- If performing recovery using continuous archiving

For the above cases, specify the --kms-secret option in the pg_ctl command, and then start the instance. This will display the prompt for the passphrase to be entered, as shown below.

```
> pg_ctl --kms-secret start
Enter secret:
```

Point

When using an automatically opening keystore, you do not need to enter the passphrase and you can automatically open the keystore when the database server starts. Refer to "6.6.2 Enabling Automatic Opening of the Keystore" for details.

6.4 Encrypting a Tablespace

Refer to "5.4 Encrypting a Tablespace".

6.5 Checking an Encrypted Tablespace

Refer to "5.5 Checking an Encrypted Tablespace".

6.6 Managing the Keystore

Describes how to manage master encryption keys when a key management system is used.

6.6.1 Changing the Master Encryption Key

To change the master encryption key, run the pgx_declare_external_master_key function as you originally set it. Refer to "6.2 Setting the Master Encryption Key" for more information.

Also, by specifying a different key management system name than the key management system name specified during installation, you can change the key management system to be used. Refer to"6.6.5 Changes to the Key Management System" for details.

6.6.2 Enabling Automatic Opening of the Keystore

You can automatically open a keystore at instance startup without entering a passphrase by specifying all credentials, including those that should be kept secret, in the key management system connection information file. To enable automatic keystore opening, run the pgx_keystore command.

Example of storing obfuscated credentials in the file sslkeypasprase.ksc

```
> pgx_keystore -s -o sslkeypassphrase.ksc
Enter secret:
```

Specify obfuscated credentials in the key management system connection information file.

```
kmip    mykmipsvr    kmip.example.com    5696    cert    sslcert=postgres.crt    sslkey=postgres.key
sslrootcert=root.crt    sslkeypassphrase-obf=sslkeypassphrase.ksc
```

The key management system connection information file is valid only on the computer on which it was created.

To disable automatic keystore opening, delete the file containing obfuscated credentials for the private key specified in sslkeypasserase-obf and delete the sslkeypasserase-obf option in the key management system connection information file.

See

Refer to "pgx_keystore" in the Reference for information on pgx_keystore command.

Refer to "Appendix A Parameters" for information on the key management system connection information file.

6.6.3 Changing Credentials for Key Management Systems

If the credentials for the key management service change, you must also change the credentials that Fujitsu Enterprise Postgres uses to connect to the key management system.

You can change the credentials used by Fujitsu Enterprise Postgres using the pgx_open_keystore function. The new credentials are used to connect to the new key management system.

If you have a streaming replication configuration, you must change credentials on all replicas.

Refer to "B.2.1 pgx_open_keystore" for information on the pgx_open_keystore function.

6.6.4 Verifying the Master Encryption Key

The view pgx_tde_master_key shows information about your master encryption key. For more information about columns, Refer to "E.1 pgx_tde_master_key" for more information about columns..

6.6.5 Changes to the Key Management System

If you need to change the linked key management system after installing the transparent data encryption function that uses the key management system, you can do so by following the procedure below.

After this step, data in Fujitsu Enterprise Postgres will be encrypted using encryption keys on the new key management system.

Defining a new key management system

If the new key management system is not listed in the key management system's connection information file, add a definition to that configuration file. Give the old key management system and the new key management system different key management system names.

Reload the configuration file for the changes to take effect.

Declaring the master encryption key to use

Use the pgx_dexlare_external_master_key function to declare a new encryption key to use. Specify the name you gave the new key management system as the key management system name. Other arguments required are the key ID to use on the new key management system and credentials. Upon successful completion, it will be encrypted using the encryption key on the new key management system.

Note

The master encryption key that encrypts the backup data before changing the key management system exists only on the old key management system. As such, you need the old key management system and the encryption keys residing there for any period of time when you might restore backup data from before the change. If you delete the encryption key on the old key management system, destroy the old key management system, cancel the key management service, etc., you will not be able to decrypt the backup data and you will not be able to use the data.

6.7 Backing Up and Restoring/Recovering the Database

Fujitsu Enterprise Postgres enables you to use the five backup and recovery methods described below.

Backup and recovery using WebAdmin

- Backup

  WebAdmin backs up encrypted data.

- Recovery

  If you recover to a point in time when the old master encryption key was in use, change to the most recent master encryption key immediately after recovery.

  Enable automatic opening of the keystore in accordance with the procedure described in "6.6.2 Enabling Automatic Opening of the Keystore". Then, use WebAdmin to recover the database.

Backup and recovery using the pgx_dmpall and pgx_rcvall commands

- Backup

  The pgx_dmpall command backs up the encrypted data.

- Recovery

  If you recover to a point in time when the old master encryption key was in use, change to the most recent master encryption key immediately after recovery.

  Configure automatic opening of the key store as necessary.

  If automatic opening of the keystore is not enabled, execute the pgx_rcvall command with the --kms-secret option specified. This will display the prompt for the passphrase to be entered.

Dump and restore using SQL

- Backup

  The files output by the pg_dump and pg_dumpall commands are not encrypted. You should, therefore, encrypt the files using OpenSSL commands or other means before saving them, as described in "5.8 Importing and Exporting the Database" below.

- Restore

  If the backup data has been encrypted using, for example Open SSL commands, decrypt that data.

  The data generated by the pg_dumpall command includes a specification to encrypt tablespaces by default. For this reason, the psql command encrypts tablespaces during restoration.

File system level backup and restore

- Backup

  Stop the instance and backup the data directory and the tablespace directory using the file copy command of the operating system. The files of encrypted tablespaces are backed up in the encrypted state.

- Restore

  Stop the instance and use the OS file copy command to restore the data storage directory or tablespace directory.

  If you recover to a point in time when the old master encryption key was in use, change to the most recent master encryption key immediately after recovery.

Continuous archiving and point-in-time recovery

- Backup

  The pg_basebackup command backs up the encrypted data as is.

- Recovery

  If you recover to a point in time when the old master encryption key was in use, change to the most recent master encryption key immediately after recovery.

  Configure automatic opening of the key store as necessary.

  If automatic opening of the keystore is not enabled, execute the pg_ctl command to start the instance with the --kms-secret option specified. This will display the prompt for the passphrase to be entered.

See

- Refer to "pg_ctl" under "Reference" in the PostgreSQL Documentation for information on the pg_ctl command.
- Refer to "Reference" in the PostgreSQL Documentation for information on the following commands:
  - psql
  - pg_dump
  - pg_basebackup
- Refer to the Reference for information on the following commands:
  - pgx_rcvall
  - pgx_dmpall
  - pg_dumpall

6.8 Importing and Exporting the Database

Refer to "5.8 Importing and Exporting the Database".

6.9 Encrypting Existing Data

Refer to "5.9 Encrypting Existing Data".

6.10 Operations in Cluster Systems

This section describes how to use transparent data encryption on cluster systems such as high-availability systems, streaming replication, and database multiplexing.

6.10.1 HA Clusters that do not Use Database Multiplexing

Take the following points when using transparent data encryption with a key management system as a key store in an HA cluster environment that does not use database multiplexing.

Placement and automatic opening of the connection information file of the key management system

The file that describes the connection information of the key management system specified in the tde_kms.kms_conninfo_file parameter of the postgresql.conf file, and the files such as certificates that are referenced from that file can be shared by the primary server and the standby server. However, the obfuscated credential file used to enable automatic opening of the keystore must be created and placed on each server according to the instructions for enabling automatic opening.

If not shared, it must be possible to connect to the key management system used from the standby server with the same key management system name as the key management system name set on the primary server. Place the connection information file and files such as certificates referenced from the connection information file on the standby server. The obfuscated credential file used to enable automatic opening of the keystore must be created and placed on each server according to the instructions in enabling automatic opening.

Changing credentials for key management systems

If the credentials for the key management system have changed, use the pgx_open_keystore function on the primary server to change the credentials. Re-enable automatic opening of the keystore on the standby server.

6.10.2 Database Multiplexing Mode

Note the following when using transparent data encryption with a key management system as a key store in environments that use streaming replication, or database multiplexing with streaming replication.

Placement and automatic opening of the connection information file of the key management system

The file that describes the connection information of the key management system specified in the tde_kms.kms_conninfo_file parameter of the postgresql.conf file, and the files such as certificates that are referenced from that file can be shared by the primary server and the standby server. However, the obfuscated credential file used to enable automatic opening of the keystore must be created and placed on each server according to the instructions for enabling automatic opening.

If not shared, it must be possible to connect to the key management system to be used with the same key management system name as the key management system name set on the primary server. Place the connection information file and files such as certificates referenced from the connection information file on all servers that configure database multiplexing mode. The obfuscated credential file used to enable automatic opening of the keystore must be created and placed on each server according to the instructions in enabling automatic opening.

Changing credentials for key management systems

If the credentials for the key management system are changed, use the pgx_open_keystore function on all servers that configure database multiplexing to change the credentials.

Starting a standby server

Open the keystore when starting the standby server. This is required to decrypt and replay the encrypted WAL received from the primary server. To open a keystore, use the pg_ctl or pgx_rcvall command with --kms-secret and provide your credentials, or enable automatic opening of the keystore.

Changing the master encryption key

Change the master encryption key on the primary server. No need to restart the standby server or reopen the keystore. Changes to the master encryption key are also reflected on the standby server.

See

Refer to "pgx_rcvall " in the Reference for information on pgx_rcvall command.

Refer to "pg_ctl" under "Reference" in the PostgreSQL Documentation for information on pg_ctl command.

Refer to "High Availability, Load Balancing, and Replication" under "Server Administration" in the PostgreSQL Documentation for information on how to set up streaming replication.

6.11 Security-Related Notes

Refer to "5.11 Security-Related Notes".

6.12 Tips for Installing Built Applications

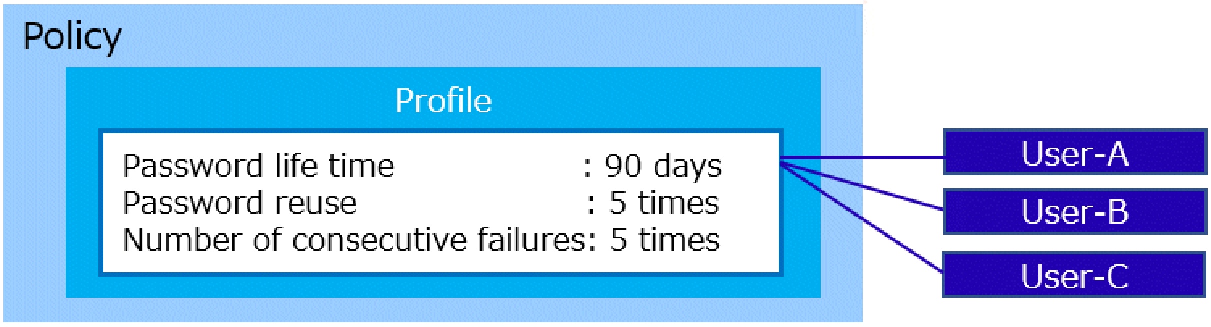
Refer to "5.12 Tips for Installing Built Applications".

Chapter 7 Policy-based Password Management

To apply a password operational policy to a user, define the policy as a profile and assign the profile to the user. This allows you to operate according to your security policy when users log in or change their passwords.

You can set the following password authentication restrictions for a profile. The contents of the profile are saved as database objects.

- Set a password life time
- Restrict password reuse
- Lock accounts that have failed to log in continuously


Policy
Profile
Password life time : 90 days
Password reuse : 5 times
Number of consecutive failures: 5 times
User-A
User-B
User-C

The policy covers the following authentication methods:

- password
- md5
- scram-sha-256

7.1 Changing the Contents of the default Profile

When you create a database user with CREATE ROLE, the database user is assigned the default profile. For the default profile, all parameters are disabled, so change the parameter values to suit your policy. Change the value with the pgx_alter_profile function.

Users with CREATEROLE privilege can modify the contents of a profile.

Refer to "B.3.1 Profile Management Functions" for functions.

[Example]

```
SELECT pgx_alter_profile('default',
‘{
    "PASSWORD_LIFE_TIME": 50,
    "PASSWORD_GRACE_TIME": "10"
    "PASSWORD_REUSE_MAX": 5,
    "PASSWORD_LOCK_TIME": 0.5,
    "PASSWORD_ALLOW_HASHED": true
}’);
```

7.2 Creating and Assigning Profiles

If you want to apply a different policy than the default profile, create a new profile and assign it to the user.

The pgx_create_profile function creates a profile.

To assign a profile to a user, use the pgx_assign_profile_to_user function.

Users with CREATEROLE privilege can create and assign profiles.

Refer to "B.3.1 Profile Management Functions" and "B.3.2 User Management Functions" for functions.

7.3 Actions to be Taken in the Event of Deviation from Policy

When a user is locked out due to consecutive password authentication failures

If password authentication fails consecutively, the user is locked and cannot log in.

If you want to allow a locked user to log in again, unlock the user. The pgx_unlock_user function releases the lock. Operations can be performed by users with CREATEROLE privilege.

Refer to "B.3.2 User Management Functions" for functions.

[Example]

```
SELECT pgx_unlock_user('user1');
```

To release a lock when a database administrator is locked, users other than the locked user with CREATEROLE privilege can do so. If the user does not exist, Fujitsu Enterprise Postgres must be started in single-user mode and the lock released.

When the password expires

When a password life time is over, the password expires and cannot be used to connect to and operate on the database until the password is changed.

The password can be changed by the expired user itself, or by a user who has the CREATEROLE privilege and who is an administrator of that user (who has ADMIN privilege for that user).

7.4 Settings in Streaming Replication Configuration

For streaming replication, the policy is enabled on both the primary and standby servers. The user lock status and user password status (Life Time, Grace Time, Expiration) are reflected and shared by all servers in the cluster, even if they are detected on either the primary or standby server.

Perform changing the contents of the profile, creating and assigning profiles, releases the lock, and change the password can only be done on the primary server.

This section describes how to setting streaming replication configuration.

1. Sets parameters in the postgresql.conf file

   Specifies the database name that can be connected to the userprofile_database parameter. If omitted, the default database "postgres" is assumed.

   Refer to "Parameters for the Policy-based Password Management" for parameters.

2. Grant privilege to users for streaming replication

   The standby server connects to the primary server and propagates any state changes it detects to the primary server. The used users for streaming replication (specified in primary_conninfo) is used to connect for state change. Grant privilege to this user. Allow the user to SET ROLE directly to the group role pgx_update_profile_status.

   [Example]

   ```
   # GRANT pgx_update_profile_status TO repluser WITH SET TRUE;
   ```

   This action enables users for streaming replication to lock all other database users or unlock the lock state due to policy deviations. Also, membership in the pgx_update_profile_status group role should only be granted to users for streaming replication.

3. Add a record to pg_hba.conf

   Add a record to accept connections from the standby server. Duplicate the record for streaming replication and specify the database name in the userprofile_database parameter as the destination database name. Authentication methods other than password authentication are recommended.

   ```
   host  replication  repluser  standbyServerAddress authenticationMethod    # For Streaming
   Replication
   ```

```
host  postgres     repluser  standbyServerAddress authenticationMethod    # For state change
propagation
```

Information

- If the primary and standby servers are disconnected, profile-based restrictions still apply on each server. However, the standby server itself cannot change its password or unlock it explicitly. After removing the cause of the disconnect, restore the connection to the primary server and business operations, and then change the password or explicitly unlock.
- Connection information for reflecting status changes to the primary server or standby server is output as "User profile status sender" to application_name in the pg_stat_replication view. When monitoring the streaming replication status with the pg_stat_replication view, exclude the "User profile status sender" line from the monitoring target.

7.5 Backup and Recovery

Backup

Profile contents, user and profile assignment status, and password history can be backed up with physical backups and the pg_dumpall command.

When dumping with SQL statements using the pg_dumpall command, database role information is backed up at the same time it is dumped.

Recovery

If you recover old backup data, the password may have expired at the time of recovery. In this case, change the password after recovery.

7.6 Profile parameters

Details of the parameters set in the profile are explained.

PASSWORD_LIFE_TIME

Number of days the same password can be used for authentication

[Supported values]

numeric: A NUMERIC value greater than or equal to 0

The unit is days. Hours and seconds can be specified with decimal places (e.g. 4.5 is equivalent to "4 days and 12 hours"). Precision is 1 second. The maximum value is 24855 days.

DEFAULT: The value of the same parameter in the default profile

UNLIMITED: No life time (same password can be used indefinitely)

The password life time will be over after PASSWORD_LIFE_TIME days from the last time the password was updated for the target role. The timing of updating the profile is not the starting point. Therefore, if you specify an extremely short number of days (such as 1 day), it may already be past the life time at the time of renewal.

It is possible to specify when a password becomes invalid using the VALID UNTIL clause of CREATE ROLE or ALTER ROLE. If you specify both the VALID UNTIL clause and PASSWORD_LIFE_TIME, both values are valid. Note that in this case, you can log in only if both constraints are met.

If there is no grace period, the password expires when the password life time is over. If you log in using password authentication in this state, you will receive a "password expired" warning and you will not be able to execute commands other than changing your password. You can change the password to resume normal operations. Other than password authentication, you can connect to and work with the database.

For streaming replication, users will not be able to connect to the standby server after the password life time is over. There is no grace period for password expiration on standby servers. Password changes must be made on the primary server.

PASSWORD_GRACE_TIME

The number of days after a password life time is over before the password expires.

[Supported values]

numeric: A NUMERIC value greater than or equal to 0

The unit is days. Hours and seconds can be specified with decimal places (e.g. 4.5 is equivalent to "4 days and 12 hours"). Precision is 1 second. The maximum value is 24855 days.

DEFAULT: The value of the same parameter in the default profile

UNLIMITED: Indefinite period

The password expiration grace period is PASSWORD_GRACE_TIME days from the first login time after the password life time is over. You can log in with your current password during the grace period, but a warning prompts you to change your password. You can operate normally except that a warning is displayed. If specified as UNLIMITED, the grace period is infinite and you will be warned to change your password at every login. Once transitioned to the grace period, even if you change the profile value after that, you cannot go back to before the grace period, and you cannot change the life time. If 0 is specified, there is no grace period and the password expires at the first login after the password life time is over. In this case, you can perform normal operations again by changing the password.

Because the standby server does not have a grace period before the password expires, users cannot connect to the standby server after the password life time is over. Password changes must be made on the primary server.

PASSWORD_REUSE_TIME

Number of days the same password cannot be reused

The password cannot be reused by the same user for this period from the time the password is updated.

This parameter must be set in combination with PASSWORD_REUSE_MAX.

[Supported values]

numeric: A NUMERIC value greater than or equal to 0

The unit is days. Hours and seconds can be specified with decimal places (e.g. 4.5 is equivalent to "4 days and 12 hours"). Precision is 1 second. The maximum value is 24855 days.

DEFAULT: The value of the same parameter in the default profile

UNLIMITED: Not reusable (but ignore these if PASSWORD_REUSE_MAX is also UNLIMITED)

PASSWORD_REUSE_MAX

Number of password changes required before password can be reused

This parameter must be set in combination with PASSWORD_REUSE_TIME.

[Supported values]

integer: An INTEGER value greater than or equal to 0

DEFAULT: The value of the same parameter in the default profile

UNLIMITED: Not reusable (but ignore these if PASSWORD_REUSE_TIME is also UNLIMITED)

Both PASSWORD_REUSE_TIME and PASSWORD_REUSE_MAX constraints must be met to reuse passwords.

For example, if you specify PASSWORD_REUSE_TIME = 30, PASSWORD_REUSE_MAX = 10, a certain password can be reused if 30 days have passed since the update time and the password has been updated 10 times or more. When password update fails due to these parameters, the SQLSTATE will be "22023: invalid_parameter_value".

Also, if one parameter has a value and the other parameter specifies UNLIMITED, the password cannot be reused. If you want to use only one condition for reuse judgment, you need to set the value of the unused parameter to 0. However, if UNLIMITED is specified for both, these parameters are ignored and passwords can be reused without restriction.

PASSWORD_ALLOW_HASHED

Whether to allow password specification by hash value when changing password

Allow if true.

[Supported values]

boolean: true or false

DEFAULT: The value of the same parameter in the default profile

When setting or changing a password in a CREATE ROLE or ALTER ROLE statement, you can specify a new password in the form of a hash value, but this is rejected if this parameter is false.

If the specified password is in the form of a hash value that is not plain text, the server cannot verify that the password has not been reused, and user can change it to the same password to extend its lifetime. Specify false to avoid this.

The password is specified in the form of a hash value when changing the password using the psql command's \password meta-command or when dumping the database user's password using the pg_dumpall command.

If this parameter is false, the password cannot be changed by the psql command's \password meta command. Specify the new password in clear text in the PASSWORD clause of the CREATE ROLE or ALTER ROLE statement. At this time, if command input history is enabled on the client side or if the setting to log SQL statements is enabled on the server side, the plaintext password might remain in the log file.

When the pg_dumpall command is run without the --no-role-passwords option, the database user's password is included in the backup file in the form of a hash value. Therefore, when recovering the database user's password, temporarily set this parameter to true in the default profile of the restore destination. If the restore is successful, the contents of the default profile are restored to the contents at the time of the backup.

FAILED_LOGIN_ATTEMPTS

Number of consecutive failed login attempts for a user before the user is locked

[Supported values]

integer: An INTEGER value greater than 0

DEFAULT: The value of the same parameter in the default profile

UNLIMITED: Indefinite period (Can fail any number of times)

If password authentication fails consecutively for the number of times specified by this parameter, the user is locked and cannot log in.

The number of failed login attempts is counted separately on each server.

PASSWORD_LOCK_TIME

Number of days after a user is locked due to consecutive login failures before the user is unlocked

[Supported values]

numeric: A NUMERIC value greater than or equal to 0

The unit is days. Hours and seconds can be specified with decimal places (e.g. 4.5 is equivalent to "4 days and 12 hours"). Precision is 1 second. The maximum value is 24855 days.

DEFAULT: The value of the same parameter in the default profile

UNLIMITED: Locked indefinitely

If set to UNLIMITED, the user will be locked indefinitely and will not be automatically unlocked. Unlocking requires an explicit pgx_unlock_user function call by a user with CREATEROLE privilege.

7.7 Worker Processes

This feature allows up to two background worker processes to reside. This information is displayed in the statistics view pg_stat_activity when it needs to be processed by a worker process. The worker processes added by this feature are those with column backend_type values of "user profile status writer" and "user profile status sender".

These processes may wait on the wait event "UserProFileWorkerMain" of type "Activity".

Chapter 8 Data Masking

Data masking is a feature that can change the returned data for queries generated by applications, so that it can be referenced by users. For example, for a query of employee data, digits except the last four digits of an eight-digit employee number can be changed to "*" so that it can be used for reference.

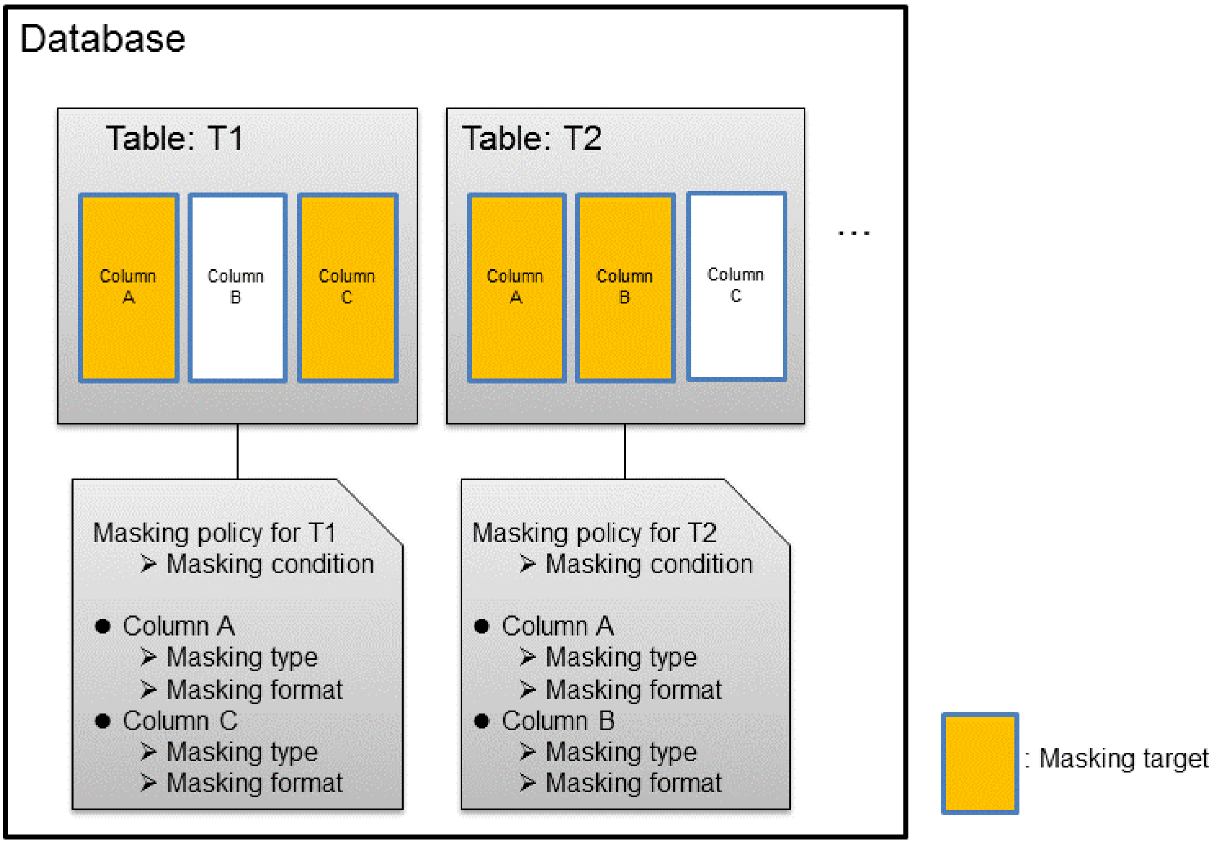
Note

When using this feature, it is recommended that the changed data be transferred to another medium for users to reference. This is because, if users directly access the database to extract the masked data, there is a possibility that they can deduce the original data by analyzing the masking policy or query result to the masking target column.

8.1 Masking Policy

Masking policy is a method of changing data under specific conditions when it is returned for a query from an application. One masking policy can be created per table. You can configure masking target, masking type, masking condition and masking format in a masking policy.

Figure 8.1 Masking policy



Note

When a masking policy is defined, the search performance for the corresponding table may deteriorate.

8.1.1 Masking Target

Masking target refers to a column to which a masking policy will be applied. When referring to a masking target or a function that includes a masking target, the execution result will be changed and obtained.

The following commands can change the execution result:

- SELECT
- COPY
- pg_dump
- pg_dumpall

Note

- If a masking target is specified to INSERT...SELECT target columns, processing will be performed using data before change.
- If a masking target other than SELECT target columns is specified, processing will be performed using data before change.
- If a masking target is specified in a function where the data type will be converted, an error will occur.

8.1.2 Masking Type

Masking type is a method to change column data that is returned from queries. Specify the masking type in the function_type parameter. The following masking types can be specified and selected depending on the masking target data type.

Full masking

All the data in the specified column is changed. The changed value returned to the application that made the query varies depending on the column data type.
For example, 0 is used for a numeric type column and a space is used for a character type column.

Partial masking

The data in the specified column is partially changed.
For example, digits except the last four digits of an employee number can be changed to "*".

Regular expression masking

The data in the specified column is changed via a search that uses a regular expression.
For example, for strings such as email address that can have variable length, "*" can be used to change characters preceding "@" by using a regular expression. Regular expression masking can only be used for character type data.

Note

- If multiple valid masking targets are specified for a function, the masking type for the left-most masking target will be applied.
  For example, if "SELECT GREATEST(c1, c2) FROM t1" is executed for numeric type masking target c1 and c2, the masking type for c1 will be applied.
- When masking the data that includes multibyte characters, do not specify partial masking for masking type. The result may not be as expected.

8.1.3 Masking Condition

Masking condition refers to the conditions configured to perform masking. Specify the masking condition in the expression parameter. Changed or actual data can be displayed for different users by defining masking condition. An expression that returns a boolean type result needs to be specified in masking condition and masking is performed only when TRUE is returned. Refer to "Value Expressions" in the PostgreSQL Documentation for information on the expressions that can be specified. Note that expressions that include a column cannot

be specified.
For example, when masking data only for "postgres" users, specify 'current_user = "postgres"' in the masking condition.

Information

Specify '1=1' so the masking condition is always evaluated to be TRUE and masking is performed all the time.

8.1.4 Masking Format

Masking format is a combination of change method and displayed characters when the masking condition is met. Masking format varies depending on the masking type. The following describes the masking format.

Full masking

With full masking, all characters are changed to values as determined by the database. Changed characters can be referenced in the pgx_confidential_values table. Also, replacement characters can be changed using the pgx_update_confidential_values system management function.

See

Refer to "8.3 Data Types for Masking" for information on the data types for which data masking can be performed.

Partial masking

With partial masking, data is changed according to the content in the function_parameters parameter. The method of specifying function_parameters varies depending on the data type.

Category	Method of specifying function_parameters
Numeric type	'*replacementCharacter, startPosition, endPosition'* - *replacementCharacter*: Specify the number to display. Specify a value from 0 to 9. - *startPosition*: Specify the start position of masking. Specify a positive integer. - *endPosition*: Specify the end position of masking. Specify a positive integer that is greater than *startPosition.* Example Specify as below to change the values from the 1st to 5th digits to 9. function_parameters := '9, 1, 5' In this example, if the original data is "123456789", it will be changed to "999996789".
Character type	*'inputFormat, outputFormat, replacementCharacter, startPosition, endPosition'* - *inputFormat*: Specify the current format of the data. Specify "V" for characters that will potentially be masked, and specify "F" for values such as spaces or hyphens that will not be masked. - *outputFormat*: Define the method to format the displayed data. Specify "V" for characters that will potentially be masked. Any character to be output can be specified for each character "F" in *inputFormat.* If you want to output a single quotation mark, specify two of them consecutively. - *replacementCharacter*: Specify any single character. If you want to output a single quotation mark, specify two of them consecutively. - *startPosition*: Specify the position of "V" as the start position of masking. For example, to specify the position of the 4th "V" from the left, specify 4. Specify a positive integer.

Category	Method of specifying function_parameters
	- *endPosition*: Specify the position of "V" as an end position of masking. When working out the end position, do not include positions of "F". For example, to specify the position of the 11th "V" from the left, specify 11. Specify a positive integer that is greater than *startPosition*. Example Specify as below to mask a telephone number other than the first three digits using *. function_parameters := 'VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11' In this example, if the original data is "012-3156-7890", it will be changed to "012-****-****".
Date/timestamp type	'MDYHMS' - M: Masks month. To mask month, enter the month from 1 to 12 after a lowercase letter m. Specify an uppercase letter M to not mask month. - D: Masks date. To mask date, enter the date from 1 to 31 after a lowercase letter d. If a value bigger than the last day of the month is entered, the last day of the month will be displayed. Specify an uppercase letter D to not mask date. - Y: Masks year. To mask year, enter the year from 1 to 9999 after a lowercase letter y. Specify an uppercase letter Y to not mask year. - H: Masks hour. To mask hour, enter the hour from 0 to 23 after a lowercase letter h. Specify an uppercase letter H to not mask hour. - M: Masks minute. To mask minute, enter the minute from 0 to 59 after a lowercase letter m. Specify an uppercase letter M to not mask minute. - S: Masks second. To mask second, enter the second from 0 to 59 after a lowercase letter s. Specify an uppercase letter S to not mask second. Example Specify as below to mask hour, minute, and second and display 00:00:00. function_parameters := 'MDYh0m0s0' In this example, if the original data is "2022-02-10 10:10:10", it will be changed to "2022-02-10 00:00:00".

See

- Refer to "B.4.2 pgx_create_confidential_policy" for information on function_parameters.
- Refer to "8.3 Data Types for Masking" for information on the data types for which masking can be performed.

Regular expression masking

With regular expression masking, data is changed according to the content of the regexp_pattern, regexp_replacement and regexp_flags parameters. For regexp_pattern, specify the search pattern using a regular expression. For regexp_replacement, specify the replacement character to use when data matches the search pattern. For regexp_flags, specify the regular expression flags.

Example

Specify as below to change all three characters starting from b to X.

regexp_pattern := 'b..'

regexp_replacement:= 'X'

regexp_flags := 'g'

In this example, if the original data is "foobarbaz", it will be changed to "fooXX".

See

- Refer to "POSIX Regular Expressions" in the PostgreSQL Documentation and check pattern, replacement, and flags for information on the values that can be specified for regexp_pattern, regexp_replacement, and regexp_flags.
- Refer to "8.3 Data Types for Masking" for information on the data types for which masking can be performed.

Note

- When column data type is character(*n*) or char(*n*) and if the string length after change exceeds n, the extra characters will be truncated and only characters up to the nth character will be displayed.
- When column data type is character varying(*n*) or varchar(*n*) and if the string length after change exceeds the length before the change, the extra characters will be truncated and only characters up to the length before change will be displayed.

8.2 Usage Method

Preparation

The following preparation is required to use this feature.

1. Set the postgresql.conf file parameters.

   Prepend "pgx_datamasking" to the shared_preload_libraries parameter.

2. Restart the instance.

3. Execute CREATE EXTENSION for the database that will use this feature.

   The target database is described as "postgres" here.

   Use the psql command to connect to the "postgres" database.

   Example

   ```
   postgres=# CREATE EXTENSION pgx_datamasking;
   CREATE EXTENSION
   ```

   Note

   You must always prepend "pgx_datamasking" to the "shared_preload_libraries" parameter.

   Information

   - Specify "false" for pgx_datamasking.enable to not use this feature. Data will not be masked even if a masking policy is configured. This feature becomes available again once "true" is specified for pgx_datamasking.enable. This setting can be made

by specifying a SET statement or specifying a parameter in the postgresql.conf file.
Example

```
postgres=# SET pgx_datamasking.enable=false;
```

- Hereafter, also perform this preparatory task for the "template1" database, so that this feature can be used by default when creating a new database.

Usage

To perform masking, a masking policy needs to be configured. The masking policy can be created, changed, confirmed, enabled, disabled or deleted during operation.
The procedures to perform these tasks are explained below with examples.

1. Creating a masking policy
2. Changing a masking policy
3. Confirming a masking policy
4. Enabling and disabling a masking policy
5. Deleting a masking policy

Note

Only database superusers can configure masking policies.

8.2.1 Creating a Masking Policy

An example of the operation on the server is shown below.

1. Create a masking policy
   Execute the pgx_create_confidential_policy system management function to create a masking policy.
   The following values are configured in this example.
   - Masking target: Numeric type c1
   - Masking type: FULL
   - Masking condition: '1=1'

```
postgres=# select pgx_create_confidential_policy(table_name := 't1', policy_name := 'p1',
expression := '1=1', column_name := 'c1', function_type := 'FULL');
 pgx_create_confidential_policy
--------------------------------
 t
(1 row)
```

2. Confirm the displayed data
   Confirm that the masking target data (column c1) has been correctly changed.

```
postgres=# select * from t1;
 c1 |      c2
----+---------------
  0 | 012-3456-7890
  0 | 012-3456-7891
  0 | 012-3456-7892
(3 row)
```

See

- Refer to "B.4.2 pgx_create_confidential_policy" for information on the pgx_create_confidential_policy system management function.

Note

- Only one masking policy can be created per table.
- All users can view the masking policy created, so do not grant the login privilege of the database where this feature is set to the users who refer to the changed data. Masking policies are defined in the "pgx_confidential_columns", "pgx_confidential_policies" and "pgx_confidential_values" tables.

8.2.2 Changing a Masking Policy

1. An example of the operation on the server is shown below.
2. Change a masking policy
   Execute the pgx_alter_confidential_policy system management function to change a masking policy.
   The following values are changed in this example.
   - Content of change: Add a masking target
   - Masking target: Character type c2
   - Masking type: PARTIAL
   - Masking condition: 'VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11'

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1',
action := 'ADD_COLUMN', column_name := 'c2', function_type := 'PARTIAL', function_parameters :=
'VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11');
 pgx_alter_confidential_policy
-------------------------------
 t
(1 row)
```

3. Confirm the displayed data
   Confirm that the masking target data has been correctly changed.

```
postgres=# select * from t1;
 c1 |      c2
----+---------------
  0 | 012-****-****
  0 | 012-****-****
  0 | 012-****-****
(3 row)
```

See

- Refer to "B.4.1 pgx_alter_confidential_policy" for information on the pgx_alter_confidential_policy system management function.

8.2.3 Confirming a Masking Policy

An example of the operation on the server is shown below.

1. Confirm information about a masking target where a masking policy is set
   Refer to the pgx_confidential_columns table to confirm the masking target where the masking policy is set.

```
postgres=# select * from pgx_confidential_columns;
 schema_name | table_name | policy_name | column_name | function_type |
function_parameters           | regexp_pattern | regexp_replacement | regexp_flags |
column_description
-------------+------------+-------------+-------------+---------------
+----------------------------------------+----------------+--------------------+-------------
+--------------------
 public      | t1         | p1          | c1          | FULL
|                                        |                |                    |             |
 public      | t1          | p1           | c2           | PARTIAL       | VVVFVVVVFVVVV, VVV-VVVV-
VVVV, *, 4, 11 |                |                    |             |
(2 row)
```

2. Confirm information about the masking policy content
   Refer to pgx_confidential_policies to confirm the masking policy content.

```
postgres=# select * from pgx_confidential_policies;
 schema_name | table_name | policy_name | expression | enable | policy_description
-------------+------------+-------------+------------+--------+--------------------
 public      | t1         | p1          | 1=1        | t      |
(1 row)
```

See

- Refer to "F.1 pgx_confidential_columns" for information on the pgx_confidential_columns table.
- Refer to "F.2 pgx_confidential_policies" for information on the pgx_confidential_policies table.

8.2.4 Enabling and Disabling a Masking Policy

An example of the operation on the server is shown below.

1. Disable a masking policy
   Execute the pgx_enable_confidential_policy system management function to disable a masking policy.

```
postgres=# select pgx_enable_confidential_policy(table_name := 't1', policy_name := 'p1',
enable := 'f');
 pgx_enable_confidential_policy
---------------------------------
 t
(1 row)
```

2. Confirm the displayed data
   Confirm that the original data is displayed by disabling the masking policy.

```
postgres=# select * from t1;
 c1 |      c2
----+---------------
  1 | 012-3456-7890
  2 | 012-3456-7891
  3 | 012-3456-7892
(3 row)
```

3. Enable a masking policy
   Execute the pgx_enable_confidential_policy system management function to enable a masking policy.

```
postgres=# select pgx_enable_confidential_policy(table_name := 't1', policy_name := 'p1',
enable := 't');
 pgx_enable_confidential_policy
---------------------------------
 t
(1 row)
```

4. Confirm the displayed data
   Confirm that the masking target data has been correctly changed.

```
postgres=# select * from t1;
 c1 |      c2
----+---------------
  0 | 012-****-****
  0 | 012-****-****
  0 | 012-****-****
(3 row)
```

See

- Refer to "B.4.4 pgx_enable_confidential_policy" for information on the pgx_enable_confidential_policy system management function.

8.2.5 Deleting a Masking Policy

An example of the operation on the server is shown below.

1. Delete a masking policy
   Execute the pgx_drop_confidential_policy system management function to delete a masking policy.

```
postgres=# select pgx_drop_confidential_policy(table_name := 't1', policy_name := 'p1');
 pgx_drop_confidential_policy
------------------------------
 t
(1 row)
```

2. Confirm the displayed data
   Confirm that the original data is displayed by deleting the masking policy.

```
postgres=# select * from t1;
 c1 |      c2
----+---------------
  1 | 012-3456-7890
  2 | 012-3456-7891
  3 | 012-3456-7892
(3 row)
```

See

- Refer to "B.4.3 pgx_drop_confidential_policy" for information on the gx_drop_confidential_policy function.

8.3 Data Types for Masking

The data types for which data masking can be performed are shown below.

Category	Data type	Masking type		
		Full masking	Partial masking	Regular expression masking
Numeric type	smallint	Y	Y	N
	integer	Y	Y	N
	bigint	Y	Y	N
	decimal	Y	Y	N
	numeric	Y	Y	N
	float	Y	Y	N
	real	Y	Y	N
	double precision	Y	Y	N

Category	Data type	Masking type		
		Full masking	Partial masking	Regular expression masking
Character type	character varying(*n*)	Y	Y	Y
	varchar(*n*)	Y	Y	Y
	character(*n*)	Y	Y	Y
	char(*n*)	Y	Y	Y
Date/timestamp type	date	Y	Y	N
	timestamp	Y	Y	N

Note

Even if the data type can be masking, if the data is a special value (NaN, Infinity, -Infinity), it is not.

8.4 Security Notes

- The logical replication is available, which allows non-backed up clusters to subscribe to databases where data masking policies are enabled. Logical replication allows publisher and subscriber databases to have their own or the same data masking policies.

  In this scenario, the user must disable data masking on the publisher database whenever a subscription is created. This ensures that subscribers are able to obtain the original data (initial copy) instead of the masked version. Then, it is the user's responsibility to set masking policies to each subscribed database.

- Take strong caution in publishing data masking's confidential tables (pgx_confidential_policies, pgx_confidential_columns, etc.) unless the user is publishing all tables of the database and wants to apply the same data masking's policies on the subscribed database for all of them.

  Otherwise, as these confidential tables contain the masking policies for all tables of the database, confidential policies of unpublished tables may be unintentionally published. Additionally, it is not possible to apply different data masking policies on the subscriber database.

Chapter 9 Periodic Operations

This chapter describes the operations that must be performed periodically when running daily database jobs.

9.1 Configuring and Monitoring the Log

Fujitsu Enterprise Postgres enables you to output database errors and warnings to a log file.

This information is useful for identifying if errors have occurred and the causes of those errors.

By default, this information is output to the system log. It is recommended that you configure Fujitsu Enterprise Postgres to collect logs from its log files (for example, log_destination) before operating Fujitsu Enterprise Postgres.

Periodically monitor the log files to check if any errors have occurred.

See

- Refer to "Error Reporting and Logging" under "Server Administration" in the PostgreSQL Documentation for information on logs.
- Refer to "Configuring Parameters" in the Installation and Setup Guide for Server for information on log settings when operating with WebAdmin.

9.2 Monitoring Disk Usage and Securing Free Space

When a database is used for an extended period, free space on the disk is continuously consumed and in some cases the disk space runs out. When this happens, database jobs may stop and no longer run.

You should, therefore, periodically monitor the usage of disk space, and delete obsolete files located in the disk.

Monitor the disk usage of the disk where the following directories are located:

- Data storage destination directory
- Transaction log storage destination (if the transaction log is stored in a different directory from the data storage destination directory)
- Backup data storage destination directory
- Tablespace storage destination directory

9.2.1 Monitoring Disk Usage

To check the disk usage, use the following operating system commands:

- df command

You can even use SQL statements to check tables and indexes individually.

Refer to "Determining Disk Usage" under "Server Administration" in the PostgreSQL Documentation for information on this method.

Information

If you are using WebAdmin for operations, a warning is displayed when disk usage reaches 80%

9.2.2 Securing Free Disk Space

Secure free disk space by using the following operating system commands to delete unnecessary files, other than the database, from the same disk unit.

- rm command

You can also secure disk space by performing the following tasks periodically:

- To secure space on the data storage destination disk:

  Execute the REINDEX statement. Refer to "9.5 Reorganizing Indexes" for details.

- To secure space on the backup data storage destination disk:

  Execute backup using WebAdmin or the pgx_dmpall command.

9.3 Automatically Closing Connections

If an application stops responding and abnormally terminates for any reason, the connection from the application may remain active on the database server. If this situation continues for an extended period, other applications attempting to connect to the database server may encounter an error, or an error indicating that the tables are unavailable may occur.

It is, therefore, recommended that idle connections be closed automatically at regular intervals.

Set the following parameters in the postgresql.conf file to indicate the time permitted to elapse before a connection is closed.

Parameter name	Setting	Description
tcp_keepalives_idle	Time until keepalive is sent (seconds) If 0, the default value of the system is used.	Sends keepalive to an idle connection at the specified interval in seconds It is recommended to specify 30 seconds.
tcp_keepalives_interval	keepalive send interval (seconds) If 0, the default value of the system is used.	Sends keepalive at the specified interval It is recommended to specify 10 seconds.
tcp_user_timeout	Time to wait for a response from the server (milliseconds) If 0, the default value of the system is used. If not set, the behavior is the same as if 0 were specified.	After establishing the connection, when sending from the client to the server, if the TCP resend process operates, specify the time until it is considered to be disconnected. If a value other than 0 is specified in this parameter, the time until automatic disconnection is determined by the waiting time specified in this parameter. The actual wait time is until the timing of the first keepalive retransmission after the time specified by this parameter has elapsed.

Note

If a value other than 0 is specified for the tcp_user_timeout parameter, the waiting time set by the tcp_keepalives_idle parameter and tcp_keepalives_interval parameter will be invalid and the waiting time specified by the tcp_user_timeout parameter will be used.

See

Refer to "Connection Settings" under "Server Administration" in the PostgreSQL Documentation for information on the parameters.

9.4 Monitoring the Connection State of an Application

Fujitsu Enterprise Postgres does not immediately delete the updated or deleted data. If the VACUUM determines there are no transactions that reference the database, Fujitsu Enterprise Postgres collects obsolete data.

However, obsolete data is not collected if there are connections that have remained active for an extended period or connections occupying resources. In this case the database may expand, causing performance degradation.

See

Refer to "Routine Vacuuming" under "Server Administration" in the PostgreSQL Documentation for information on the VACUUM command.

In such cases, you can minimize performance degradation of the database by monitoring problematic connections.

The following method is supported for monitoring connections that have been in the waiting status for an extended period:

- 9.4.1 Using the View (pg_stat_activity)

9.4.1 Using the View (pg_stat_activity)

Use the view (pg_stat_activity) to identify and monitor connections where the client has been in the waiting status for an extended period.

Example

The example below shows connections where the client has been in the waiting status for at least 60 minutes.

However, when considering continued compatibility of applications, do not reference system catalogs directly in the following SQL statements.

```
postgres=# select * from pg_stat_activity where backend_type = 'client backend' and state='idle in
transaction' and current_timestamp > cast(query_start + interval '60 minutes' as timestamp);
-[ RECORD 1 ]----+------------------------------
datid            | 13003
datname          | db01
pid              | 4638
leader_pid       |
usesysid         | 10
usename          | fsep
application_name | apl01
client_addr      | 192.33.44.15
client_hostname  |
client_port      | 27500
backend_start    | 2022-02-24 09:09:21.730641+09
xact_start       | 2022-02-24 09:09:23.858727+09
query_start      | 2022-02-24 09:09:23.858727+09
state_change     | 2022-02-24 09:09:23.858834+09
wait_event_type  | Client
wait_event       | ClientRead
state            | idle in transaction
backend_xid      |
backend_xmin     |
query_id         |
query            | begin;
backend_type     | client backend
```

See

- Refer to "Notes on Application Compatibility" in the Application Development Guide for information on maintaining application compatibility.

- Refer to "The Statistics Collector" under "Server Administration" in the PostgreSQL Documentation for information on pg_stat_activity.

9.5 Reorganizing Indexes

Normally, a database defines indexes in tables, but if data is frequently updated, indexes can no longer use free space in the disk efficiently. This situation can also cause a gradual decline in database access performance.

To rearrange used space on the disk and prevent the database access performance from declining, it is recommended that you periodically execute the REINDEX command to reorganize indexes.

Check the disk usage of the data storage destination using the method described in "9.2 Monitoring Disk Usage and Securing Free Space".

Note

Because the REINDEX command retrieves the exclusive lock for an index being processed and locks writing of tables that are the source of the index, other processes that access these may stop while waiting to be locked.

Therefore, it is necessary to consider measures such as executing the command after the task is completed.

See

Refer to "Routine Reindexing" under "Server Administration" in the PostgreSQL Documentation for information on reorganizing indexes by periodically executing the REINDEX command.

Point

Typically, reorganize indexes once a month at a suitable time such as when conducting database maintenance. Use SQL statements to check index usage. If this usage is increasing on a daily basis, adjust the frequency of recreating the index as compared to the free disk space.

The following example shows the SQL statements and the output.

However, when considering continued compatibility of applications, do not reference system catalogs and functions directly in the following SQL statements. Refer to "Notes on Application Compatibility" in the Application Development Guide for details.

[SQL statements]

```
SELECT
  nspname AS schema_name,
  relname AS index_name,
  round(100 * pg_relation_size(indexrelid) / pg_relation_size(indrelid)) / 100 AS index_ratio,
  pg_size_pretty(pg_relation_size(indexrelid)) AS index_size,
  pg_size_pretty(pg_relation_size(indrelid)) AS table_size
FROM pg_index I
  LEFT JOIN pg_class C ON (C.oid = I.indexrelid)
  LEFT JOIN pg_namespace N ON (N.oid = C.relnamespace)
WHERE
  C.relkind = 'i' AND
  pg_relation_size(indrelid) > 0
ORDER BY pg_relation_size(indexrelid) DESC, index_ratio DESC;
```

[Output]

```
 schema_name |            index_name             | index_ratio | index_size | table_size
-------------+-----------------------------------+-------------+------------+------------
 public      | pgbench_accounts_pkey             |        0.16 | 2208 KB    | 13 MB
 pg_catalog  | pg_depend_depender_index          |         0.6 | 224 KB     | 368 KB
 pg_catalog  | pg_depend_reference_index         |        0.58 | 216 KB     | 368 KB
...
```

See

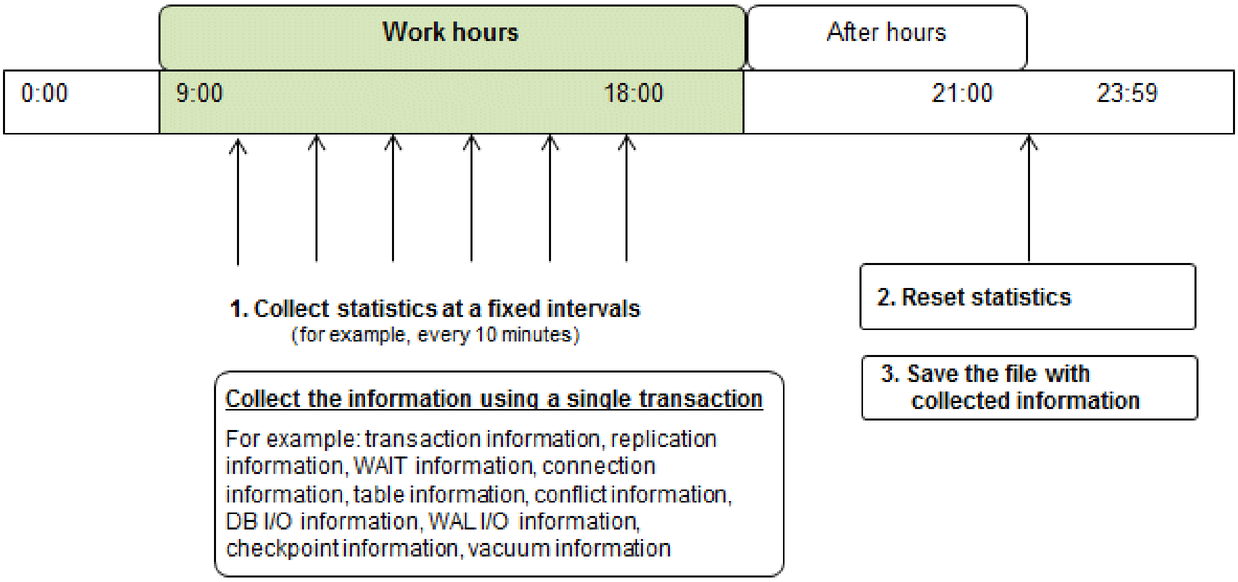
Refer to "Notes on Application Compatibility" in the Application Development Guide for information on maintaining application compatibility.

9.6 Monitoring Database Activity

Fujitsu Enterprise Postgres enables you to collect information related to database activity. By monitoring this information, you can check changes in the database status.

This information includes wait information for resources such as internal locks, and is useful for detecting performance bottlenecks. Furthermore, you should collect this information in case you need to request Fujitsu technical support for an investigation.

Figure 9.1 Overview of information collection



Work hours
After hours
0:00
9:00
18:00
21:00
23:59
1. Collect statistics at a fixed intervals
(for example, every 10 minutes)
2. Reset statistics
3. Save the file with collected information
Collect the information using a single transaction
For example: transaction information, replication information, WAIT information, connection information, table information, conflict information, DB I/O information, WAL I/O information, checkpoint information, vacuum information

1. Collect statistics at fixed intervals during work hours.

   Accumulate the collected information into a file.

   Wherever possible, collect data from the various statistics views using a single transaction, because it enables you to take a snapshot of system performance at a given moment.

   Refer to "9.6.1 Information that can be Collected" for information on the system views that can be collected.

2. Reset statistics after work hours, that is, after jobs have finished.

   Refer to "9.6.3 Information Reset" for information on how to reset statistics.

3. Save the file with collected information.

   Keep the file with collected information for at least two days, in order to check daily changes in performance and to ensure that the information is not deleted until you have sent a query to Fujitsu technical support.

Where jobs run 24 hours a day, reset statistics and save the file with collected information when the workload is low, for example, at night.

Note

Statistics cumulatively add the daily database value, so if you do not reset them, the values will exceed the upper limit, and therefore will not provide accurate information.

The subsections below explain the following:

- Information that can be collected
- Collection configuration
- Information reset

9.6.1 Information that can be Collected

Information that can be collected is categorized into the following types:

- Information common to PostgreSQL
- Information added by Fujitsu Enterprise Postgres

Information common to PostgreSQL

See

Refer to "Monitoring Database Activity" under "Server Administration" in the PostgreSQL Documentation for information on information common to PostgreSQL.

Information added by Fujitsu Enterprise Postgres

You can collect the following information added by Fujitsu Enterprise Postgres.

Table 9.1 Information added by Fujitsu Enterprise Postgres

View name	Description
pgx_stat_lwlock	Displays statistic related to lightweight lock, with each type of content displayed on a separate line. This information helps to detect bottlenecks. Refer to "D.2 pgx_stat_lwlock" for details.
pgx_stat_latch	Displays statistics related latches, with each type of wait information within Fujitsu Enterprise Postgres displayed on a separate line. This information helps to detect bottlenecks. Refer to "D.3 pgx_stat_latch" for details.
pgx_stat_walwriter	Displays statistics related to WAL writing, in a single line. Refer to "D.4 pgx_stat_walwriter" for details.
pgx_stat_sql	Displays statistics related to SQL statement executions, with each type of SQL statement displayed on a separate line. Refer to "D.5 pgx_stat_sql" for details.
pgx_stat_gmc	Displays statistics related to Global Meta Cache hit ration and used memory size. Refer to “D.6 pgx_stat_gmc” for detail. Also refer to Chapter 14 Global Meta Cache" for information on the Global Meta Cache.

9.6.2 Collection Configuration

The procedure for configuring collection depends on the information content.

- Information common to PostgreSQL
- Information added by Fujitsu Enterprise Postgres

Information common to PostgreSQL

See

Refer to "The Statistics Collector" in "Monitoring Database Activity" under "Server Administration" in the PostgreSQL Documentation for information on information common to PostgreSQL.

Information added by Fujitsu Enterprise Postgres

Information added by Fujitsu Enterprise Postgres is collected by default.

To enable or disable information collection, change the configuration parameters in postgresql.conf. The following table lists the views for which you can enable or disable information collection, and the configuration parameters.

View name	Parameter
pgx_stat_lwlock pgx_stat_latch	track_waits (*1)
pgx_stat_sql	track_sql
pgx_stat_gmc	track_gmc

Remarks: You cannot change the collection status for pgx_stat_walwriter.

*1: When executing the SQL statement with EXPLAIN ANALYZE, processing time may increase because of this information collection. It is recommended to set this parameter to "off" when executing EXPLAIN ANALYZE to check the processing time.

Refer to "Appendix A Parameters" for information on the parameters.

9.6.3 Information Reset

This section describes how to reset information.

Information added by Fujitsu Enterprise Postgres

You can reset information added by Fujitsu Enterprise Postgres by using the pg_stat_reset_shared function in the same way as for information common to PostgreSQL.

Configure the following parameters in the pg_stat_reset_shared function:

Function	Type of return value	Description
pg_stat_reset_shared(text)	void	Reset some cluster-wide statistics counters to zero, depending on the argument (requires superuser privileges). Calling pg_stat_reset_shared('lwlock') will zero all counters shown in pgx_stat_lwlock. Similarly, in the following cases, all values of the pertinent statistics counter are reset: - If pg_stat_reset_shared('latch') is called: All values displayed in pgx_stat_latch - If pg_stat_reset_shared('walwriter') is called: All values displayed in pgx_stat_walwriter - If pg_stat_reset_shared('sql') is called: All values displayed in pgx_stat_sql - If pg_stat_reset_shared('gmc') is called:

Function	Type of return value	Description
		All values except size column in pgx_stat_gmc

See

Refer to "Statistics Functions" in "Monitoring Database Activity" under "Server Administration" in the PostgreSQL Documentation for information on other parameters of the pg_stat_reset_shared function.

9.7 Monitoring Deferred SQL and Periodically Backing up statistics

After production starts, SQL execution can be delayed due to a variety of factors, including unexpected system load and data bias. You can detect delays early by monitoring for delays that affect operations. Monitoring can also reduce the time required for recovery tasks, such as determining the cause and resolving delays.

One cause of deferred SQL is a change in statistics. Some delays caused by statistics can be resolved by restoring the statistics before the delay occurred. For this reason, back up your statistics periodically to guard against delays.

Operational workflow

The workflow required to monitor delayed SQL and back up statistics is shown below.

Corresponding time	Work item	Work details
During environment construction	(1) Enabling the pg_dbms_stats extension	Enable the pg_dbms_stats extension with the Create Extension statement.
	(2) Logging settings for delayed SQL monitoring	Set the log_min_duration_statement parameter etc. in postgresql.conf.
During system development	(3) Creating and reviewing a script for backing up statistics (*1)	Provide a script to back up statistics according to the statistics and Vacuum operation.
During operation	(4) Periodic backup of statistics	Execute the script created in (3) to back up the statistics.
	(5) Monitoring delayed SQL and adjusting the allowable delay time	Monitor the occurrence of delayed SQL and adjust the allowable delay time to find the optimal settings for operation.
When delayed SQL occurs	(6) Investigating the cause of delayed SQL	If a delayed SQL occurs, we will investigate the cause of the SQL delay and consider countermeasures.
	(7) Restoring statistics (*2)	Delayed SQL is eliminated by restoring and fixing the backed up statistics.

*1: Although it is possible to perform the work during operation, please consider the impact on operation and perform the work before starting operation.

*2: If it is determined that the cause is not statistics, take appropriate measures according to the cause.

The specific work content is shown below.

(1) Enabling the pg_dbms_stats extension

To back up and restore statistics, you need the export and import functions of the pg_dbms_stats extension, so enable the pg_dbms_stats extension.

For instructions on how to enable it, refer to " Setting Up pg_dbms_stats" in the Installation and Setup Guide for Server.

(2) Logging settings for delayed SQL monitoring

Database logs are used to monitor delayed SQL.

Make the following settings in postgresql.conf. Note that the date and time are important when investigating the cause of delayed SQL, set the format and file name to include the date and time.

Parameter	Specified value
logging_collector	on
log_line_prefix	Specify the following format and add information required for investigation, such as the time and executed application. [%t]%u %d %p[%l]
log_filename	Since you will need to check past execution records, set it so that you can see the date and time. postgresql.%Y-%m-%d
log_min_duration_statement	300000

Point

If the value specified for the log_min_duration_statement parameter is small, even SQL that does not need to be considered delayed will be output to the log, and if it is too large, delayed SQL cannot be detected. Therefore, if the setting value cannot be estimated, set it small at the start of operation, and adjust it to a larger value as necessary while monitoring during operation.

(3) Creating and reviewing a script for backing up statistics

To back up statistics, use the export function of the pg_dbms_stats extension.

The backup is used not only for restoration but also for identifying the cause of delayed SQL, so the time the backup was performed is important. Therefore, be sure to create a script like the one below so that the backup time is known.

We also provide a sample script that uses the export function to export on a database basis. This allows users to respond more flexibly by making modifications themselves, such as making changes on a schema basis, so copy the script to the execution current before using it.

The sample script is stored below. Also, "<*x*>" indicates the product version.

```
/opt/fsepv<x>server64/share/doc/extension/export_plain_stats-16.sql.sample
```

The following is an example of how to create a database using Fujitsu Enterprise Postgres 16 by copying the database to the current execution directory as "export_plain_stats-16.sql".

```
expprt PATH=/opt/fsepv16server64/bin:$PATH
export CURRENTDIR=/pg_dbms_stats/backup

pushd "${CURRENTDIR}"

# make file name
EXECDATE=`date '+%Y%m%d%H%M%S'`
FILENAME=pg_dbms_stats_back.${EXECDATE}.dmp

# export
psql -d database1 -f export_plain_stats-16.sql

# rename dump file
mv export_stats.dmp ${FILENAME}
popd
```

(4) Periodic backup of statistics

Execute the script created in "(3) Creating and reviewing a script for backing up statistics" to back up the statistics information on a regular basis.

Backup execution interval

Perform backup at the following timings according to the statistics update method.

When statistics is automatically updated (autovacuum is enabled)

Statistics are updated when autovacuum is run.

Therefore, use the OS's cron command or schedule function of the task manager to periodically execute the export function and back up the statistics to a file.

Therefore, use the OS's cron command or a scheduler such as task manager to periodically execute the script created in "(3) Creating and reviewing a script for backing up statistics" and back up the statistics information to a file.

Obtain the shortest autovacuum execution interval from the log, and back up the statistics at an interval shorter than the obtained autovacuum execution interval. autovacuum logs can be obtained by setting the log_autovacuum_min_duration parameter to 0.

When users update statistics

If the user controls the updating of statistics using the ANALYZE command, etc.,back up the statistics at the same time as updating the statistics.

Backup retention period

The backed up statistics will be used as investigation material to determine the cause of delayed SQL if a performance problem occurs. Therefore, keep it for the period of time expected to be required from the occurrence of the operational problem to its resolution.

Point

- The capacity required for backup mainly depends on the number of objects (schemas, tables, etc.) contained in the database. Estimate the required capacity from the backup file when checking the operation of the script and secure this space.
- To restore backed up statistics, the names of objects such as schemas and tables in the backup source must match. If you change the name of an object during operation, change the backup file name or storage location so that you can check whether it matches the object definition when restoring statistics.

(5) Monitoring delayed SQL and adjusting the allowable delay time

Monitors SQL that is delayed beyond the time set in the log_min_duration_statement parameter.

The settings in this section will be output in the following format.

```
[2024-04-30 10:20:11 JST]user1 postgres 3414[1]LOG: duration: 301001.541 ms  statement: SELECT
pg_sleep(301)
[2024-04-30 10:26:12 JST]user1 postgres 3414[1]LOG: duration: 302002.321 ms  statement: SELECT
pg_sleep(302)
```

When detected SQL is deemed to be delayed more than expected

Refer to "(6) Investigating the cause of delayed SQL" to determine the cause and take appropriate measures.

When SQL with execution time that does not affect operation is detected

The current setting just outputs redundant logs, so make major changes to the log_min_duration_statement parameter and then reload to enable it.

(6) Investigating the cause of delayed SQL

There are many factors that can cause SQL delays, so you must first identify the cause.

Here we will show you how to investigate SQL delays caused by changes in statistics.

Note that in the following cases, delays caused by statistics are unlikely, so recommend investigating from a different perspective (I/O, system load, etc.).

- When the SQL statement does not contain a search condition
- When the SQL statement uses the pg_hint_plan extension

1. Check the delayed SQL from the server log and identify the schema used by that SQL.
2. In a database cluster (such as a development environment) separate from the production environment, create an environment (hereafter referred to as the reproduction environment) with the same database name and table definition as the environment where the problem occurred. The reproduction environment also requires the pg_dbms_stats extension, so enable the pg_dbms_stats extension.
3. Use the pg_dump command from the production environment to obtain the target table definition.
   Since the data is not needed at this time, use the -s option of the pg_dump command from the production environment and use the extracted table definition.

   The following is an example of output to the file ddl_schema.dmp when the schema to be investigated is schema_1 in database database1.

   ```
   pg_dump -d database1 -s schema_1 > ddl_schema.dmp
   ```

   Using ddl_schema.dmp, create a research schema named schema_1 in the database named database1 in the reproduction environment.

   ```
   psql -d database1 -f ddl_schema.dmp
   ```

4. In order to identify the cause of the SQL delay, use the backed up statistics to check whether there is a cause of the delay in the statistics. Import the statistics into the schema in the reproduction environment using the pg_dbms_stats import function (dbms_stats.import_schema_stats).
   The following is an example of restoring statistics for schema schema_1 from the backup file "pg_dbms_stats_back. 20240422011223.dmp".

   ```
   psql -d database1 -c
   "SELECT dbms_stats.import_schema_stats('schema_1','pg_dbms_stats_back.20240422011223.dmp')"
   ```

   Then, execute the EXPLAIN command for the deferred SQL to obtain the execution result (query plan).

5. Restore the statistics by separately importing the statistics that were backed up during the previous time period when deferred SQL was running stably.
   For example, if the target SQL is executed daily as part of regular batch processing, use the data from the same time the previous day.
   For online processing, use the data from a previous time period when a similar business application is expected to be executed.
   The following is an example of restoring the statistics from the backup file "pg_dbms_stats_back.20240421010001.dmp" from one day ago, assuming the SQL is used in batch processing.

   ```
   psql -d database1 -c
   "SELECT dbms_stats.import_schema_stats('schema_1','pg_dbms_stats_back.20240421010001.dmp')"
   ```

   It likewise obtains the result of executing the EXPLAIN command on deferred SQL (query plan).

6. Compare the execution results (query plans) of 4. and 5., and take the following action based on the results.
   - If the query plans are the same or the expected execution time of the query plan in 5. is longer

     This may be a problem other than statistics (system load, index imbalance, I/O, etc.), so investigate from a different perspective.

   - Other than the above

     The problem is likely that the optimal query plan was not selected due to updated statistics.

     Refer to "(7) Restoring statistics" and import the statistics from the period when the system was operating stably into the production environment to restore the system.

(7) Restoring statistics

Restore the production environment statistics to the state they were in before the delay occurred. The procedure is the same as for verification, but it is as follows:

```
psql -d database1 -c
"SELECT dbms_stats.import_schema_stats('schema_1','pg_dbms_stats_back.20240422000000')"
```

9.8 Performance Tuning

9.8.1 Enhanced Query Plan Stability

Fujitsu Enterprise Postgres estimates the cost of query plans based on SQL statements and database statistics, and selects the least expensive query plan. However, like other databases, Fujitsu Enterprise Postgres does not necessarily select the most suitable query plan. For example, it may suddenly select unsuitable query plan due to changes in the data conditions.

If you know that the statistics will not change significantly, you can control the selection of the same query plan by directly specifying which query plan to select in pg_hint_plan, or by fixing the statistics referenced by the planner in pg_dbms_stats.

Also, pg_dbms_stats from Fujitsu Enterprise Postgres can fix statistics (height of Btree indexes) that OSS pg_dbms_stats cannot account for. Refer to "9.8.1.1 Fixing the Height of a Btree Index".

See

- For information about setting up pg_hint_plan and pg_dbms_stats, refer to "pg_hint_plan" and "pg_dbms_stats" in the Installation and Setup Guide for Server.
- For basic usage of pg_hint_plan and pg_dbms_stats, refer to below.
    - Control execution plans with pg_hint_plan

      https://www.postgresql.fastware.com/postgresql-insider-tun-hint-plan
    - Control execution plans by fixing statistics with pg_dbms_stats

      https://www.postgresql.fastware.com/postgresql-insider-tun-dbms-stt
- For more information about pg_hint_plan and pg_dbms_stats, refer to the OSS web page.

9.8.1.1 Fixing the Height of a Btree Index

When PostgreSQL creates a query plan, it also references the height of the Btree index to be used as part of the statistics. However, OSS pg_dbms_stats does not include the height of the Btree index in its fixed statistics. The height of a Btree index is also fixed in pg_dbms_stats for Fujitsu Enterprise Postgres.

Fixing the height of a Btree index covers Fujitsu Enterprise Postgres 16 and later, so to maintain compatibility with earlier versions, specify something like:

- PostgreSQL 9.2 and earlier compatibility

```
pg_dbms_stats.use_tree_height = off
```

- Fujitsu Enterprise Postgres 15 SP2 and earlier compatibility

```
pg_dbms_stats.lock_tree_height = off
```

See the table below for details on each parameter. If you want the parameter value to be valid for all sessions, specify it in postgresql.conf and then reload.

Parameter	Specified value	Default
pg_dbms_stats.use_tree_height	Specify whether to include the Btree index height in cost calculation and plan generation when statistics are fixed. on: Includes Btree index height in cost calculations and plan generation. off: Do not include Btree index height in cost calculation or plan generation.	on
pg_dbms_stats.lock_tree_height	Specify whether to also fix the height of the Btree index when you fix the statistics. Note that this is valid only when the specified value of pg_dbms_stats.use_tree_height is "on". on: Fixes the height of the Btree index at the height when locked. off: The height of the Btree index is not fixed and varies depending on the amount of data in the index.	on

When you want to import statistics from your legacy environment

The format of files handled by the import/export functions of this version of pg_dbms_stats is different from the format of pg_dbms_stats in OSS or Fujitsu Enterprise Postgres 15SP2 or earlier. Therefore, statistics exported from pg_dbms_stats for OSS or Fujitsu Enterprise Postgres 15SP2 or earlier should be imported using the dbms_stats.import_<obj>_stats_no_tree_height function.

How to specify the dbms_stats.import_<obj>_stats_no_tree_height function

```
dbms_stats.import_<obj>_stats_no_tree_height('Absolute path of export file')
```

<obj> indicates a database (currently connected), schema, table, or column.

Import using the dbms_stats.import_<obj>_stats function provided by OSS is not possible.

If you use the wrong import function, an error message containing the following HINT will be output. Check the combination of input file and import function type, and use the correct import function.

```
HINT:  The import function may be incorrectly combined with the format of the exported data. Please
check the documentation for the relationship between the import function and the available data.
```

Chapter 10 Streaming Replication Using WebAdmin

This chapter describes streaming replication using WebAdmin.

A standby instance for streaming replication can be created from a standalone instance, a master instance, or another standby instance. The standby instance connects to the master instance and uses WAL records to replicate data. The standby instance can be used for read-only operations.

WebAdmin cannot back up standby instances.

Additionally, streaming replication can be set to asynchronous or synchronous mode.

Point

- If a streaming replication cluster is created using WebAdmin, the network with the host name (or IP address) specified in [Host name] will be used across sessions of WebAdmin, and also used as the log transfer network.
- To use a network other than the job network as the log transfer network, specify the host name other than the job network one in [Host name].

For characters that cannot be specified in WebAdmin, refer to “Appendix J WebAdmin Disallow User Inputs Containing Hazardous Characters”.

10.1 Creating a Standby Instance

Follow the procedure below to create a standby instance.

1. In the [Instances] tab, select the instance from which a standby instance is to be created.
2. Click ⋖.
3. Enter the information for the standby instance to be created.
    - [Location]: Whether to create the instance in the server that the current user is logged in to, or in a remote server. The default is "Local", which will create the instance in the server machine where WebAdmin is currently running.
    - [Replication credential]: The user name and password required for the standby instance to connect to the master instance. The user name and password can be entered or selected from the Wallet. Refer to "Appendix I WebAdmin Wallet" for information on creating wallet entries.
    - [Instance name]: Name of the standby database instance to create.

      The name must meet the conditions below:

        - Maximum of 16 characters
        - The first character must be an ASCII alphabetic character
        - The other characters must be ASCII alphanumeric characters
    - [Instance port]: Port number of the standby database instance.
    - [Host IP address]: The IP address of the server machine where the standby instance is to be created. This information is needed to configure the standby instance to be connected to the master.
    - [Data storage path]: Directory where the database data will be stored
    - [Backup storage path]: Directory where the database backup will be stored
    - [Transaction log path]: Directory where the transaction log will be stored
    - [Encoding]: Database encoding system
    - [Replication mode]: Replication mode of the standby instance to be created ("Asynchronous" or "Synchronous")

- [Application name]: The reference name of the standby instance used to identify it to the master instance.

  The name must meet the conditions below:

  - Maximum of 16 characters
  - The first character must be an ASCII alphabetic character
  - The other characters must be ASCII alphanumeric characters

4. Click ✓ to create the standby instance.

If using WebAdmin to manage Mirroring Controller, the message below may be output to the server log or system log in the standby instance. No action is required, as the instance is running normally.

```
ERROR:  pgx_rcvall failed
ERROR:  pgx_rcvall: backup of the database has not yet been performed, or an incorrect backup storage
directory was specified
```

10.2 Promoting a Standby Instance

Streaming replication between a master and standby instance can be discontinued using WebAdmin.

Follow the procedure below to promote a standby instance to a standalone instance, thereby discontinuing the streaming replication.

1. In the [Instances] tab, select the standby instance that needs to be promoted.
2. Click ↑.
3. Click [Yes] from the confirmation dialog box.

The standby instance will be promoted and will become a standalone instance, which is not part of a streaming replication cluster.

Once the standby instance is promoted to become a standalone instance, the backup storage status will be "Error". This is because no backups are available when the instance is newly promoted to a standalone instance. The status will be reset if a new backup is performed by clicking [Solution] or 🗄.

10.3 Converting an Asynchronous Replication to Synchronous

You can convert an asynchronous standby instance to a synchronous standby instance.

Converting an Asynchronous standby instance to Synchronous can cause the master instance to queue the incoming transactions until the standby instance is ready. For this reason, it is recommended that this operation be performed during a scheduled maintenance period.

When adding a synchronous standby instance, Fujitsu Enterprise Postgres will only keep the first entry in [Synchronous standby names] in synchronous state.

Follow the procedure below to convert an Asynchronous standby instance to Synchronous.

1. In the [Instances] tab, select the master instance of the relevant cluster.
2. Click ☰.
3. In the [Streaming replication] section, edit the value for [Synchronous standby names].
   - Add the "Application name" of the standby instance you want to be in Synchronous mode.
4. Click ✓.
5. Select the master instance and click ⟲.
6. Select the standby instance. [Instance type] will now show the updated status.

See

To learn more about the differences between synchronous and asynchronous standby modes and their behavior, refer to "Streaming Replication" in "High Availability, Load Balancing, and Replication" in the PostgreSQL Documentation.

10.4 Converting a Synchronous Replication to Asynchronous

Follow the procedure below to convert a Synchronous standby instance to Asynchronous.

1. In the [Instances] tab, select the master instance of the relevant cluster.
2. Click ☰.
3. In the [Streaming replication] section, edit the value for [Synchronous standby names].
   - Remove the "Application name" of the standby instance you want to be in Asynchronous mode.
4. Click ✓.
5. Select the master instance and click ⟲.
6. Select the standby instance. [Instance type] will now show the updated status.

See

To learn more about the differences between synchronous and asynchronous standby modes and their behavior, refer to "Streaming Replication" in "High Availability, Load Balancing, and Replication" in the PostgreSQL Documentation.

10.5 Joining a Replication Cluster

WebAdmin facilitates the joining of an old master of the cluster as a standby node.

1. In the [Instances] tab, select the remote instance (from where the new cluster node will stream WAL entries), and then click 👤.
2. Configure the node to accept streaming requests from the new node.
3. In the [Instances] tab, select the new standby instance (which needs to be connected to the cluster), and then click ⚙.
4. Set [Replication host name] to the remote instance.
5. Enter [Replication credential].

   Specify the user name and password required for the standby instance to connect to the remote instance. The user name and password can be entered or selected from the Wallet. Refer to "Appendix I WebAdmin Wallet" for information on creating wallet entries. Replication credential (user name and password) should not contain hazardous characters. Refer to “Appendix J WebAdmin Disallow User Inputs Containing Hazardous Characters”.
6. Enter [Host IP address].

   Specify the IP address of the node where the standby instance was created.
7. Click ✓ to open the [Join replication cluster] dialog box.

   Click [Yes] to set up the standby instance.
8. Upon successful completion, the confirmation dialog box will be displayed.
9. Click [Close] to return to the instance details window.

The instance will become a standby instance, and will be part of the streaming replication cluster. The replication diagram will display the relationship between the standby instance and the remote instance. The user can change the replication relationship of the remote instance from asynchronous to synchronous (and vice versa) using the [Configuration] window.

Chapter 11 Installing and Operating the In-memory Feature

The in-memory feature enables fast aggregation using Vertical Clustered Index (VCI) and memory-resident feature.

VCI has a data structure suitable for aggregation, and features parallel scan and disk compression, which enable faster aggregation through reduced disk I/O.

The memory-resident feature reduces disk I/O that occurs during aggregation. It consists of the preload feature that reads VCI data to memory in advance, and the stable buffer feature that suppresses VCI data eviction from memory. The stable buffer feature secures the proportion specified by parameter in the shared memory for VCI.

This chapter describes how to install and operate the in-memory feature.

11.1 Installing Vertical Clustered Index (VCI)

This section describes the installation of VCI.

1. Evaluating whether to Install VCI
2. Estimating Resources
3. Setting up

11.1.1 Evaluating whether to Install VCI

VCI uses available resources within the server to increase scan performance.

It speeds up processing in many situations, and can be more effective in the following situations:

- Single table processing
- Processing that handles many rows in the table
- Processing that handles some columns in the table
- Processing that performs very heavy aggregation such as simultaneous sum and average aggregation

VCI will not be used in the following cases, so it is necessary to determine its effectiveness in advance:

- The data type of the target table or column contains VCI restrictions.
- The SQL statement does not meet the VCI operating conditions
- VCI is determined to be slower based on cost estimation

Note

If performing operations that use VCI, the full_page_writes parameter setting in postgresql.conf must be enabled (on). For this reason, if this parameter is disabled (off), operations that use VCI return an error. In addition, to perform operations for tables that do not create a VCI when the full_page_writes parameter setting is temporarily disabled (off), do not create a VCI or perform operations to tables that created a VCI during that time.

See

- Refer to "11.1.4 Data that can Use VCI" for information on VCI restrictions.
- Refer to "Scan Using a Vertical Clustered Index (VCI)" - "Operating Conditions" in the Application Development Guide for information on VCI operating conditions.

11.1.2 Estimating Resources

Estimate resources before setting up VCI.

Select the aggregation that you want to speed up and identify the required column data. The additional resources below are required according to the number of columns.

- Memory

  Secure additional capacity required for the disk space for the column for which VCI is to be created.

- Disk

  Secure additional disks based on the disk space required for the column for which VCI is to be created, as VCI stores column data as well as existing table data on the disk. It is recommended to provide a separate disk in addition to the existing one, and specify it as the tablespace to avoid impact on any other jobs caused by I/O.

Information

The operations on VCI can continue even if the memory configured for VCI is insufficient by using VCI data on the disk.

See

Refer to "Estimating Memory Requirements" and "Estimating Database Disk Space Requirements" in the Installation and Setup Guide for Server for information on how to estimate required memory and disk space.

11.1.3 Setting up

This section describes how to set up VCI.

Setup flow

1. Setting Parameters
2. Installing the Extensions
3. Creating VCI
4. Confirming that VCI has been Created

11.1.3.1 Setting Parameters

Edit postgresql.conf to set the required parameters for VCI. After that, start or restart the instance.

The following table lists the parameters that need or are recommended to be configured in advance:

Parameter name	Setting value	Description	Required
shared_preload_libraries	Literal 'vci, pg_prewarm'	VCI and shared library to be preloaded at server start.	Y
session_preload_libraries	Literal 'vci, pg_prewarm'	VCI and shared library to be preloaded at connection start.	Y
reserve_buffer_ratio	Percentage of shared memory to be used for stable buffer table	Proportion of shared memory to be used for a stable buffer table.	N
vci.control_max_workers	Number of background workers that manage VCI	Number of background workers that manage VCI. Add this value to max_worker_processes.	N
vci.max_parallel_degree	Maximum number of background workers used for parallel scan	Maximum number of background workers used for parallel scan. Add this value to max_worker_processes.	N

Example

```
shared_preload_libraries = 'vci, pg_prewarm'
session_preload_libraries = 'vci, pg_prewarm'
reserve_buffer_ratio = 20
vci.control_max_workers = 8
vci.max_parallel_degree = 4
max_worker_processes = 18 # Example: If the initial value was 6, 6 + 8 + 4 = 18
```

Note

An error occurs if you use VCI to start instances when procfs is not mounted. Ensure that procfs is mounted before starting instances.

See

- Refer to "Appendix A Parameters" for information on all parameters for VCI. Refer also to default value for each parameter and details such as specification range in the same chapter. Refer to "Server Configuration" under "Server Administration" in the PostgreSQL documentation for information on shared_preload_libraries, session_preload_libraries, and max_worker_processes.

11.1.3.2 Installing the Extensions

Execute CREATE EXTENSION to install the VCI and pg_prewarm extensions. Both extensions need to be installed for each database.

- Installing VCI

```
db01=# CREATE EXTENSION vci;
```

- Installing pg_prewarm

```
db01=# CREATE EXTENSION pg_prewarm;
```

Note

- Only superusers can install VCI extensions.
- VCI extensions can only be installed in public schema.
- Some operations cannot be performed for VCI extensions. Refer to "11.2.1 Commands that cannot be Used for VCI" for details.

11.1.3.3 Creating a VCI

Execute the CREATE INDEX statement with the "USING vci" clause to create a VCI for the desired columns and the "WITH (stable_buffer=true)" clause to enable the stable buffer feature.

To use a separate disk for the VCI, specify the TABLESPACE clause.

```
db01=# CREATE INDEX idx_vci ON table01 USING vci (col01, col02) WITH (stable_buffer=true);
```

Note

- Some table types cannot be specified on the ON clause of CREATE INDEX. Refer to "11.1.4.1 Relation Types" for details.
- Some data types cannot be specified on the column specification of CREATE INDEX. Refer to "11.1.4.2 Data Types" for details.
- Some operations cannot be performed for VCI. Refer to "11.2.1 Commands that cannot be Used for VCI" for details.

- The same column cannot be specified more than once on the column specification of CREATE INDEX.

- VCI cannot be created for table columns that belong to the template database.

- CREATE INDEX creates multiple views named vci_*10digitRelOid_5digitRelAttr_1charRelType* alongside VCI itself. These are called VCI internal relations. Do not update or delete them as they are used for VCI aggregation.

- All data for the specified column will be replaced in columnar format when VCI is created, so executing CREATE INDEX on an existing table with data inserted takes more time compared with a general index (B-tree). Jobs can continue while CREATE INDEX is running.

- When CREATE INDEX USING VCI is invoked on a partitioned table, the default behavior is to recurse to all partitions to ensure they all have matching indexes. Each partition is first checked to determine whether an equivalent index already exists, and if so, that index will become attached as a partition index to the index being created, which will become its parent index. If no matching index exists, a new index will be created and automatically attached; the name of the new index in each partition will be determined as if no index name had been specified in the command. If the ONLY option is specified, no recursion is done, and the index is marked invalid. (ALTER INDEX ... ATTACH PARTITION marks the index valid, once all partitions acquire matching indexes.) Note, however, that any partition that is created in the future using CREATE TABLE ... PARTITION OF will automatically have a matching index, regardless of whether ONLY is specified.

- Parallel index build is not supported on VCI indexes.

11.1.3.4 Confirming that the VCI has been Created

Execute the SELECT statement to reference the pg_indexes catalog, and confirm that the VCI was created for the target columns.

Example

```
db01=# SELECT indexdef FROM pg_indexes WHERE indexdef LIKE '%vci%';
                         indexdef
----------------------------------------------------------
CREATE INDEX idx_vci ON table01 USING vci (col01, col02)
(1 row)
```

11.1.4 Data that can Use VCI

This section describes on which relation types and for which data types VCIs can be created.

11.1.4.1 Relation Types

VCIs cannot be created on some relation types.

The ON clause of CREATE INDEX described in "11.1.3.3 Creating a VCI" cannot specify relations on which VCIs cannot be created.

- Relations on which VCIs can be created
  - Normal tables
  - UNLOGGED TABLEs
- Relations on which VCIs cannot be created
  - Materialized views
  - Temporary tables
  - Views
  - Temporary views
  - Foreign tables

11.1.4.2 Data Types

VCIs cannot be created for some data types.

The column specification of CREATE INDEX described in "11.1.3.3 Creating a VCI" cannot specify a column with data type on which VCIs cannot be created.

Category	Data type	Supported by VCI?
Numeric	smallint	Y
	integer	Y
	bigint	Y
	decimal	Y
	numeric	Y
	real	Y
	double precision	Y
	serial	Y
	bigserial	Y
Monetary	money	Y
Character	varchar(*n*)	Y
	char(*n*)	Y
	nchar	Y
	nvarchar(*n*)	Y
	text	Y
Binary	bytea	Y
Date/time	timestamp	Y
	timestamp with time zone	Y
	date	Y
	time	Y
	time with time zone	Y
	interval	Y
Boolean	boolean	Y
Geometric	point	N
	line	N
	lseg	N
	box	N
	path	N
	polygon	N
	circle	N
Network address	cidr	N
	inet	N
	macaddr	N
	macaddr8	N

Category	Data type	Supported by VCI?
Bit string	bit(*n*)	Y
	bit varying(*n*)	Y
Text search	tsvector	N
	tsquery	N
UUID	uuid	Y
XML	xml	N
JSON	json	N
	jsonb	N
Range	int4range	N
	int8range	N
	numrange	N
	tsrange	N
	tstzrange	N
	daterange	N
Object identifier	oid	N
	regproc	N
	regprocedure	N
	regoper	N
	regoperator	N
	regclass	N
	regtype	N
	regconfig	N
	regdictionary	N
pg_lsn type	pg_lsn	N
Array type	-	N
User-defined type (Basic type, enumerated type, composite type, and range type)	-	N

11.2 Operating VCI

This section describes how to operate VCI.

11.2.1 Commands that cannot be Used for VCI

Some operations cannot be performed for VCI extensions and VCI itself.

This section describes SQL commands that cannot be executed for the VCI extensions and VCI itself, and client application commands.

SQL commands

- Operations that cannot be performed for the VCI extension

Command	Subcommand	Description
ALTER EXTENSION	UPDATE	The VCI extension cannot be specified. This operation is not required for VCI.
	SET SCHEMA	
	ADD	
	DROP	
CREATE EXTENSION	SCHEMA	The subcommands on the left cannot be performed if the VCI extension is specified, except when the public schema is specified. This operation is not required for VCI.

- Operations that cannot be performed on relations containing a VCI

Command	Subcommand	Description
ALTER INDEX	SET	The subcommands on the left cannot be performed if a VCI is specified. If the operation is required, delete the VCI using DROP INDEX, and re-create it using CREATE INDEX after completing the operation.
	SET TABLESPACE	
	ALL IN TABLESPACE	
ALTER OPERATOR CLASS	RENAME TO	The subcommands on the left cannot be performed if a VCI is specified. This operation is not supported in VCI.
	OWNER TO	
	SET SCHEMA	
ALTER OPERATOR FAMILY	ADD	
	DROP	
	RENAME TO	
	OWNER TO	
	SET SCHEMA	
ALTER TABLE	ALL IN TABLESPACE *name* [OWNED BY *roleName*] SET TABLESPACE *newTablespace*	A tablespace that contains a VCI cannot be specified. If the operation is required, delete the VCI using DROP INDEX, and re-create it using CREATE INDEX after completing the operation.
	DROP [COLUMN] [IF EXISTS] *colName* [RESTRICT | CASCADE]	A column that contains a VCI cannot be specified. If the operation is required, delete the VCI using DROP INDEX, and re-create it using CREATE INDEX after completing the operation.
	ALTER [COLUMN] *colName* [SET DATA] TYPE *dataType* [COLLATE *collation*] [USING *expr*]	
	CLUSTER ON *indexName*	A VCI cannot be specified. This operation is not supported in VCI.
	REPLICA IDENTITY {DEFAULT | USING INDEX *indexName* | FULL | NOTHING}	
CLUSTER	-	A table that contains a VCI and VCI cannot be specified. If the operation is required, delete the VCI using DROP INDEX, and re-create it using CREATE INDEX after completing the operation.
CREATE INDEX	UNIQUE	The subcommands on the left cannot be performed if a VCI is specified.

Command	Subcommand	Description
	CONCURRENTLY	This operation is not supported in VCI.
	[ASC | DESC]	
	[NULLS { FIRST | LAST }]	
	WITH	
	WHERE	
	INCLUDE	
CREATE OPERATOR CLASS	-	A VCI cannot be specified. This operation is not supported in VCI.
CREATE OPERATOR FAMILY	-	
CREATE TABLE	EXCLUDE	
DROP INDEX	CONCURRENTLY	The subcommands on the left cannot be performed if a VCI is specified. This operation is not supported in VCI.
REINDEX	-	A VCI cannot be specified. This command is not required as VCI uses daemon's automatic maintenance to prevent disk space from increasing.

Client application command

- Operations that cannot be performed on relations containing a VCI

Command	Description
clusterdb	Clustering cannot be performed for tables that contain a VCI.
reindexdb	VCIs cannot be specified on the --index option.

11.2.2 Data Preload Feature

The first aggregation using VCI immediately after an instance is started may take time, because the VCI data has not been loaded to buffer.

Therefore, use the preload feature to load the VCI data to buffer in advance when performing VCI aggregation after an instance is started.

When using the preload feature, execute the function pgx_prewarm_vci to each VCI created with CREATE INDEX.

```
db01=# SELECT pgx_prewarm_vci('idx_vci');
```

See

Refer to "B.5 VCI Data Load Control Function" for information on pgx_prewarm_vci.

Chapter 12 Parallel Query

Fujitsu Enterprise Postgres enhances parallel queries, by taking into consideration the aspects below:

- CPU load calculation
- Increase of workers during runtime
- Statistics view displays the action state

12.1 CPU Load Calculation

There may be a case when the user tries to execute a parallel query but there is not enough CPU available.

Adding dynamic workers at this stage will provide no benefits - instead, it may add overhead due to context switching.

Fujitsu Enterprise Postgres takes into consideration the current CPU load when deciding on the number of workers for parallel query.

12.2 Increase of Workers during Runtime

This Fujitsu Enterprise Postgres enhancement allows systems to allocate additional workers during query execution (if there are workers available at the time). This improves query performance, which could otherwise starve of CPU if there were fewer or no workers when the query started.

Note

The ability to increase workers during runtime is available only with parallel query.

12.3 Statistics View Displays the Action State

You can check the action state by using the existing statistics collector. Monitoring the statistics enables you to check the action state of parallel processing through, for example, changes in the system load.

The following table describes the column added to the statistics views:

pg_stat_all_tables

Column	Data type	Description
parallel_query	bigint	Number of parallel query initialized in the table.

The parallel query column has also been added to the following views:

- pg_stat_sys_tables
- pg_stat_user_tables
- pg_stat_xact_all_tables
- pg_stat_xact_sys_tables
- pg_stat_xact_user_tables

See

Refer to "The Statistics Collector" in "Server Administration" in the PostgreSQL Documentation for information on the statistics collector and views.

Values will be set for the pgx_stat_sql view in Fujitsu Enterprise Postgres if parallel query is run.

See

Refer to "D.5 pgx_stat_sql" for information on pgx_stat_sql view.

Chapter 13 High-Speed Data Load

High-speed data load uses the pgx_loader command to load data from files at high speed into Fujitsu Enterprise Postgres.

Note

- This feature is not available in single-user mode. This is because in single-user mode instances run in a single process, and it cannot start parallel workers.

13.1 Installing High-Speed Data Load

This section describes how to install high-speed data load.

Installation flow

1. Deciding whether to Install
2. Estimating Resources
3. Setup

13.1.1 Deciding whether to Install

The feature achieves high speed data load by executing the COPY FROM command in parallel. If the database system is unable to use sufficient resources due to the feature using more resources than the COPY FROM command of PostgreSQL, load performance may be inferior to that of the COPY FROM command of PostgreSQL. Therefore, determine if the feature will be effective by considering the factors below before deciding to install.

Database server memory

If the value of shared_buffers in postgresql.conf is small, fewer data pages are cached to the shared memory of the database server. This will result in multiple parallel workers more often having to wait for write exclusive locks to the same data page. Moreover, the smaller the number of data pages, the more often the table expands. During table expansion, access to the table is exclusive (standby event name: extend), so write time increases. To cater for that, increase the value of shared_buffers.

See

The standby event name is stored in the wait_event column of the pg_stat_activity view. Refer to "wait_event Description" in "The Statistics Collector" in the PostgreSQL Documentation for details.

Frequency of checkpoints

If checkpoints are issued at short intervals, write performance is reduced. If the messages below are output to the server log during data writes, increase the values of max_wal_size and checkpoint_timeout in postgresql.conf to reduce the frequency of checkpoints.

Example

```
LOG:  checkpoints are occurring too frequently (19 seconds apart)
HINT:  Consider increasing the configuration parameter "max_wal_size".
```

13.1.2 Estimating Resources

Estimate the memory requirements for high-speed data load.

Up to 128 parallel workers to perform data load can be specified for this feature. The additional resources below are required depending on the number of parallel workers.

- Dynamic shared memory created during data load

  The feature creates shared memory and shared memory message queues during data load. These are used to send external data from the back end to the parallel workers, and for error notifications.

Note

If the value of shared_buffers in postgresql.conf is small, the system will often have to wait for write exclusive locks to the same data page (as described in "Database server memory" in "13.1.1 Deciding whether to Install"). Since input data cannot be loaded from the shared memory message queues during such waits, they will often be full. In these cases, it will not be possible to write to the shared memory message queues, resulting in degraded data load performance.

See

Refer to "High-Speed Data Load Memory Requirements" in the Installation and Setup Guide for Server for information on the formula for estimating memory requirements.

13.1.3 Setup

This section describes how to set up high-speed data load.

Setup flow

1. Setting Parameters
2. Installing the Extension

13.1.3.1 Setting Parameters

Set the parameters required for high-speed data load in postgresql.conf. After that, start or restart the instance.

The table below lists the postgresql.conf parameters that must be changed, and the values that must be added to their current values. After editing postgresql.conf, start or restart the instance.

Parameter	Setting	Required
max_prepared_transactions	Add the number of transactions that can be prepared by parallel workers during data load to the parameter's current value. The resulting value must be equal to or greater than the maximum number of parallel workers used with this feature.	Mandatory
max_worker_processes	Number of parallel workers to perform data load.	Mandatory
max_parallel_workers	Add the maximum number of parallel workers to be used in a parallel query by this feature to the parameter's current value. The resulting value must be equal to or greater than the number of parallel workers used with this feature.	Mandatory

Example

The example below shows how to configure 2 instances of high-speed data load being executed simultaneously using a degree of parallelism of 4.

```
max_prepared_transactions = 13  #Example if the initial value was 5: 5 + 2 x 4 = 13
max_worker_processes = 16        #Example if the initial value was 8: 8 + 2 x 4 = 16
max_parallel_workers = 12        #Example if the initial value was 4: 4 + 2 x 4 = 12
```

Note

As shown in the example above, set the value of max_prepared_transactions, max_worker_processes and max_parallel_workers multiplied by the number of instances of this feature executed simultaneously.

The table below lists the postgresql.conf parameters that must also be checked.

Parameter	Setting	Required
dynamic_shared_memory_type	Implementation of dynamic shared memory to be used by the instance. The default value is recommended.	Mandatory

See

Refer to "Resource Consumption" in the PostgreSQL Documentation for information on the parameters.

13.1.3.2 Installing the Extension

Execute CREATE EXTENSION to install the high-speed data load extension. The extension needs to be installed on each database.

Example

The example below installs the extension on the "postgres" database.

```
postgres=# CREATE EXTENSION pgx_loader;
CREATE EXTENSION
```

Note

- Only superusers can install the high-speed data load extension.
- The high-speed data load extension can only be installed on the public schema.

13.2 Using High-Speed Data Load

This section describes how to use high-speed data load.

13.2.1 Loading Data

To load data from a file into a Fujitsu Enterprise Postgres table, execute the pgx_loader command in load mode.

Example

The example below loads the file /path/to/data.csv (2000 records) into table tbl using a degree of parallelism of 3.

```
$ pgx_loader load -j 3 -c "COPY tbl FROM '/path/to/data.csv' WITH CSV"
LOAD 2000
```

Point

If an external file contains data that violates the format or constraints, the data load may fail partway through, resulting in delays for routine tasks such as nightly batch processing. Therefore, it is recommended to remove the invalid data before executing the data load.

Note

The data inserted using this feature is dumped as a COPY command by the pg_dump command and the pg_dumpall command.

See

- Refer to "pgx_loader" in the Reference for information on the command.
- Refer to "COPY" in the PostgreSQL Documentation for information on the deployment destination and access privileges for external files.

13.2.2 Checking Progress

If you are performing a data load with a large external file as input, you can verify that the process is continuing by getting progress information during the load. Progress information can be obtained from the pgx_stat_progress_loader view. This view displays the sum of the progress information of the back-end process and the number of parallel worker processes. Search the pgx_stat_progress_loader view, for example, with a SELECT statement, to locate the appropriate row. After running the pgx_loader command, look in the pg_stat_activity view and locate a row in the pgx_stat_progress_loader view with the PID obtained.

Example

1. See the pg_stat_activity view. (9311 for back-end processes, 9312, 9313, 9314 for worker processes)

```
postgres=# select pid, application_name, backend_type from pg_stat_activity
pid  | application_name |         backend_type
-----+------------------+-----------------------------
6216 |                  | autovacuum launcher
6218 |                  | logical replication launcher
6271 | psql             | client backend
9311 | pgx_loader       | client backend
9312 |                  | parallel loader for PID 9311
9313 |                  | parallel loader for PID 9311
9314 |                  | parallel loader for PID 9311
6214 |                  | background writer
6213 |                  | checkpointer
6215 |                  | walwriter
```

2. Check the information in the pgx_stat_progress_loader view.

```
postgres=# SELECT * FROM pgx_stat_progress_loader
pid  | datid | datname | relid | command   | type | bytes_processed | bytes_total |
tuples_processed | tuples_excluded
-----+-------+---------+-------+----------+------+-----------------+-------------
+------------------+----------------
9311 |  222  | testdb  | 333   |COPY FROM | FILE |        192000  |    450000   |        189000
|           3000
```

Refer to "D.7 pgx_stat_progress_loader" for information on the pgx_stat_progress_loader view.

Note

When you run the pgx_loader command, the PostgreSQL pg_stat_progress_copy view prints the progress of the back-end process and the number of parallel worker processes on each line. The backend process progress information tuples_processed, tuples_excluded is 0. Also, bytes_processed and bytes_total for worker processes are 0.

```
postgres=# SELECT * FROM pg_stat_progress_copy
pid  | datid | datname | relid | command   | type | bytes_processed | bytes_total | tuples_processed |
tuples_excluded
-----+-------+---------+-------+-----------+------+-----------------+-------------+------------------
+----------------
9311 |  222  | testdb  | 333   |COPY FROM | FILE |         192000  |    450000   |              0
|              0
9312 |  222  | testdb  | 333   |COPY FROM | FILE |              0  |         0   |          63000
|           1000
9313 |  222  | testdb  | 333   |COPY FROM | FILE |              0  |         0   |          63000
|           1000
9314 |  222  | testdb  | 333   |COPY FROM | FILE |              0  |         0   |          63000
|           1000
```

Refer to "pg_stat_progress_copy View" in the PostgreSQL Documentation for information on the pg_stat_progress_copy view.

13.2.3 Recovering from a Data Load that Ended Abnormally

If a system interruption such as a server failure occurs while high-speed data load is being performed, transactions prepared using this feature may be changed to the in-doubt state. At that point, resources occupied by the transaction will be locked, and access to the relevant resources from other transactions will be blocked, rendering them unusable.

In such cases, check transactions that are in an in-doubt state, and resolve them.

Checking for in-doubt transactions

This section describes how to check for in-doubt transactions.

1. Refer to the pgx_loader_state table in the pgx_loader schema.

   Retrieve the global transaction identifier (gid column) of in-doubt transactions. In-doubt transactions will contain "rollback" in the column "state".

   Example

   The example below retrieves the global transaction identifier (gid) of in-doubt transactions performed by the database role myrole and that used table tbl. The retrieved global transaction identifiers pgx_loader:9589 and pgx_loader:9590 identify in-doubt transactions.

   ```
   postgres=# SELECT gid, state FROM pgx_loader.pgx_loader_state WHERE
   postgres-# role_oid IN (SELECT oid FROM pg_roles WHERE rolname = 'myrole') AND
   postgres-# relation_oid IN (SELECT relid FROM pg_stat_all_tables WHERE
   postgres-# relname = 'tbl');
          gid       | state
   -----------------+------------
    pgx_loader:9590 | rollback
    pgx_loader:9591 | commit
    pgx_loader:9589 | rollback
   (3 rows)
   ```

2. Refer to the pg_prepared_xacts system view.

   Check if the in-doubt transactions retrieved above exist.

Example

The example below checks if in-doubt transactions with the global transaction identifiers pgx_loader:9589 and pgx_loader:9590 exist.

```
postgres=# SELECT gid FROM pg_prepared_xacts WHERE gid IN ('pgx_loader:9589','pgx_loader:9590');
       gid
------------------
 pgx_loader:9590
 pgx_loader:9589
(2 rows)
```

See

Refer to "G.1 pgx_loader_state" for information on the pgx_loader_state table.

Resolving in-doubt transactions

Execute the pgx_loader command in recovery mode to resolve in-doubt transactions.

After executing the pgx_loader command in recovery mode, perform the procedure described in "Checking for in-doubt transactions" to check if the in-doubt transactions have been resolved.

Example

The example below completes the in-doubt transactions prepared for table tbl.

```
$ pgx_loader recovery -t tbl
```

Point

The recovery mode of the pgx_loader command only resolves transactions prepared by high-speed data load. For transactions prepared by an application using distributed transactions other than this feature, follow the procedure described in "17.13 Actions in Response to Error in a Distributed Transaction".

13.3 Removing High-Speed Data Load

This section describes how to remove high-speed data load.

13.3.1 Removing the Extension

Execute DROP EXTENSION to remove the high-speed data load extension. The extension needs to be removed on each database.

Example

The example below removes the extension on the "postgres" database.

```
postgres=# DROP EXTENSION pgx_loader;
DROP EXTENSION
```

Note

- The information required for operation of high-speed data load is stored in the pgx_loader_state table of the pgx_loader schema. Do not remove the high-speed data load extension if the pgx_loader_state table is not empty.
- Only superusers can remove the high-speed data load extension.
- The high-speed data load extension can only be removed on the public schema.

Chapter 14 Global Meta Cache

The Global Meta Cache (GMC) feature loads a meta cache into shared memory using the pgx_global_metacache parameter. This reduces the amount of memory required throughout the system.

14.1 Usage

Describes how to use the Global Meta Cache feature.

14.1.1 Deciding Whether to Enable the Global Meta Cache Feature

Global Meta Cache is a mechanism for sharing meta caches between processes, so it works well on systems with a high number of resources accessed and SQL connections. The number of resources is primarily the number of tables accessed by a process, the number of indexes, or the total number of all columns in all tables accessed.

In particular, consider using Global Meta Cache if the total size of the meta cache for each process exceeds the amount of installed memory, or takes up a large portion of that memory, thereby squeezing memory allocations to the database cache or the Operating system file cache. Using Global Meta Cache may increase the time it takes to execute a single SQL to reference a meta cache on shared memory, but you can expect a greater benefit from being able to allocate more memory, such as for the database cache.

If performance degradation using Global Meta Cache is not acceptable, you may want to limit the number of tables accessed by a process.

14.1.2 Estimating Memory for Global Meta Cache

To enable the Global Meta Cache feature, the pgx_global_metacache parameter must specify an upper limit on the size of the shared memory (Hereinafter, the GMC area) dedicated to Global Meta Cahche. Ideally, this upper limit should be the size estimated in “Appendix A Parameters”. Values lower than this can still work, but refer to "14.1.3 How the GMC Memory Area Is Used" on using the GMC area to understand the disadvantages.

14.1.3 How the GMC Memory Area Is Used

At startup, the memory for the GMC area is not used much, but the GMC area grows as new meta caches are placed in the GMC area. If it does, it discards any meta caches that the system determines are not heavily used and places a new one in the GMC area.

Therefore, the GMC area will work even if it is smaller than the estimate, but the meta cache will be regenerated if the discarded meta cache needs to be reused. Note that if this happens frequently, it will degrade overall performance.

With this in mind, it may not be a problem if, for example, the tables to be accessed are different depending on the time zone, and the degradation of the time zone immediately after the change is acceptable.

In any case, be sure to test and tune the system thoroughly before running it.

14.1.4 Enabling the Global Meta Cache Feature

To Enable the Global Meta Cache feature edit the postgresql.conf file and set the pgx_global_metacahe parameter. Restarting the instance after editing the postgresql.conf file is required. Refer to "Appendix A Parameters" for information on the parameters.

Parameter Name	Description
pgx_global_metacache	Specify the maximum amount of memory for the GMC area on shared memory. When it’s set to 0 (default value), the Global Meta Cache feature is disabled. When enabled, the minimum value allowed is 10MB.

When the cache is created, if the total amount of meta caches on shared memory exceeds the value specified by pgx_global_metacache, the inactive, unreferenced meta caches are removed from the GMC area. Note that if all GMC are in use and the cache cannot be created in the GMC area, the cache is temporarily created in the local memory of the backend process.

Example

Here is an example postgresql.conf configuration:

```
pgx_global_metacache = 800 MB
```

Wait Events

The Global Meta Cache feature may cause wait events. Wait events are identified in the wait_event column of the pg_stat_activity view. GMC specific wait events are described below.

[GMC Feature Wait Events]

Wait Event Type	Wait Event Name	Description
LWLock	GlobalCatcache	Waiting to find, add, and remove meta caches in the GMC area.
IPC	GMCSweep	Waiting to select a meta cache that can be deleted when GMC space is low. If the GMC is fully referencing and there is no deletable meta cache, it is waiting for the reference to be removed and a deletable meta cache to be selected.

Note

If GMCSweep is happened frequently, increase the pgx_global_metacache setting.

See

Refer to "Viewing Statistics" in the PostgreSQL Documentation for information on the pg_stat_activity view.

14.1.5 Estimating Resources

Refer to "Global Meta Cache Memory Requirements" in the Installation and Setup Guide for Server for formulas to estimate the amount of memory used by the Global Meta Cache feature.

14.2 Statistics

Describes the statistics for the Global Meta Cache feature.

14.2.1 System View

You can check the cache hit ratio and size of the GMC area in the system view pgx_stat_gmc. Refer to "D.6 pgx_stat_gmc" for information on the columns.

If the cache hit ratio is low and the current memory usage is close to pgx_global_metacache, increase the pgx_global_metacache setting because performance may be degraded.

Refer to "9.6 Monitoring Database Activity" in the Operations Guide for information on the statistics.

Chapter 15 Local Meta Cache Limit

Local Meta Cache Limit feature limits the size of a Local Meta Cache by removing it if it has not been accessed for a long time.

15.1 Usage

Describes how to use the Local Meta Cache Limit feature.

15.1.1 Deciding Whether to Enable the Local Meta Cache Limit Feature

Refer to "Appendix A Parameters", after estimating the total amount of memory to be used as the catalog cache and relation cache, when the total amount of memory exceeds the amount of installed memory or occupies a large amount of installed memory, consider using this feature.

This feature adds the action of discarding the meta cache that has been held permanently. If you attempt to refer to a destroyed meta cache again, the meta cache is recreated, so using this feature will result in poor performance compared to not using it.

Therefore, read the following to understand how to discard a meta cache.

- 15.1.3 Cache Removal when Local Meta Cache Limit is Enabled
- 15.1.4 Performance Impact and Parameter Tuning of the Local Meta Cache Limit Feature
- Parameters for the Local Meta Cache Limit feature

How to set the upper limit with these considerations is described in detail in the estimation formula in "Appendix A Parameters".

15.1.2 How to Set Parameters for the Local Meta Cache Limit Feature

To enable the Local Meta Cache Limit feature, set the pgx_catalog_cache_max_size and pgx_relation_cache_max_size parameters.

Parameter Name	Description
pgx_catalog_cache_max_size	Specify the maximum amount of memory that the backend process should use as the catalog cache. You can enable catalog cache removal by setting it to 8 KB or more. When it is set to 0 (default value), the catalog cache removal is disabled.
pgx_relation_cache_max_size	Specify the maximum amount of memory that the backend process should use as the relation cache. You can enable relation cache removal by setting it to 8 KB or more. When it is set to 0 (default value), the relation cache removal is disabled.

Example

Here is an example postgresql.conf configuration:

```
pgx_catalog_cache_max_size = 1MB
pgx_relation_cache_max_size = 1MB
```

15.1.3 Cache Removal when Local Meta Cache Limit is Enabled

When this feature is enabled, the caching strategy is to keep the cache as long as possible within the specified upper limit. If holding a new cache exceeds the limit, consider locality of reference and remove the cache from the one with the longest unreferenced time.

However, because the cache used by active transactions cannot be removed, if a transaction uses a large number of caches, the cache may be held above the limit. In this case, remove the all caches at the end of the transaction. This is necessary to free up memory.

In PostgreSQL, in order to acquire memory at high speed, a memory block of a certain size is acquired from the OS, and a small memory is cut out from the block and used. The memory for the metacache is cut out in the same way. Therefore, it is possible to return the memory block to the OS by destroying all the meta caches scattered throughout the memory block. When this happens, the next SQL execution will be slowed down due to the re-creation of the meta cache. Therefore, upper limit of feature should be set to a value larger than the size of the meta cache used by at least one transaction.

When the size of the meta cache exceeds the upper limit, the following message is output:

```
WARNING: could not reduce Cat/RelCacheMemoryContext size to AA kilobytes, reduced to BB kilobytes
HINT: consider increasing the configuration parameter pgx_catalog/relation_cache_max_size
```

(***AA***: Upper limit, ***BB***: Amount of memory actually used)

CatCacheMemoryContext and RelCacheMemoryContext are memory areas for storing the catalog cache and relation cache, respectively. If this message is output, consider increasing the upper limit.

If the memory consumption by the backend process exceeds the allowable value by increasing the upper limit, reconsider the SQL to be executed, such as reducing the number of tables accessed in one transaction, or add memory adjust to the amount of memory used.

15.1.4 Performance Impact and Parameter Tuning of the Local Meta Cache Limit Feature

By observing how much meta cache regeneration is taking place, you can determine if the low upper limit is the cause of the failure to achieve the desired performance.

From the message below, calculate the cache hit ratio as follows:

```
Cache hit ratio = Number of cache hits ÷ Number of times the cache was searched
```

If the cache hit ratio is 80% or higher, this feature will not be the main factor that impedes performance. If not, raise the upper limit and see if performance can reach the goal. In doing so, first try to shift the focus of allocation to the relations cache. This is because when executing SQL, the relation cache generated based on the catalog cache is mainly referenced, so it is advantageous to leave a large amount of relation cache.

```
Catalog cache:catalog cache hit stats: search XX, hits YY
Relation cache:relation cache hit stats: search XX, hits YY
```

(***XX***: Number of times the cache was searched, ***YY***: Number of cache hits)

This message is printed when the transaction ends. However, if you output the message frequently, the performance will be degraded by itself, so you can adjust the output interval with the following parameters.

Parameter Name	Description
pgx_cache_hit_log_interval	When the transaction ends, if the time set in this parameter has elapsed since the previous message was output, the message is output. If set to 0, a message will be output each time the transaction ends. Setting -1 disables the output. The default value is 10min. Even if pgx_catalog_cache_max_size and pgx_relation_cache_max_size are disabled, the message output of the corresponding cache will be invalid. Immediately after connecting to the server, a small transaction occurs before the request from the user application, such as for user authentication. Since it is meaningless to know the hit ratio for these, a message is output at the end of the transaction that started after the time set in this parameter has elapsed after connecting to the server. For the same reason, setting a small value such as 0 may result in a message being printed at the end of such a small transaction. You can check which transaction the message corresponds to from the information output at the beginning. This information depends on the setting of the parameter log_line_prefix.

Example

Here is an example postgresql.conf configuration:

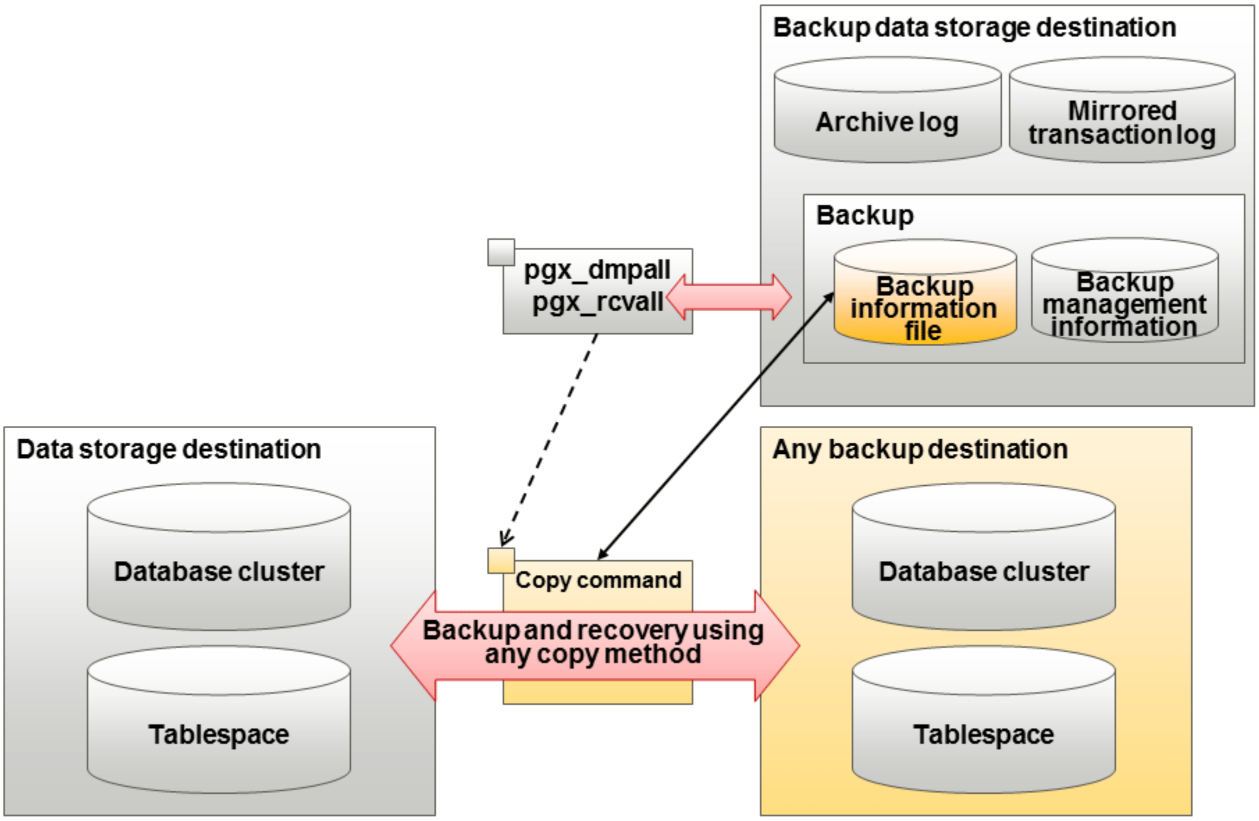
```
pgx_cache_hit_log_interval= 30min
```

Chapter 16 Backup/Recovery Using the Copy Command

By using a copy command created by the user, the pgx_dmpall command and the pgx_rcvall command can perform backup to any destination and can perform recovery from any destination using any copy method.

Copy commands must be created in advance as executable scripts for the user to implement the copy process on database clusters and tablespaces, and are called when executing the pgx_dmpall and pgx_rcvall commands.

This appendix describes backup/recovery using the copy command.



Point

It is also possible to back up only some tablespaces using the copy command. However, database resources not backed up using the copy command are still backed up to the backup data storage destination.

Note

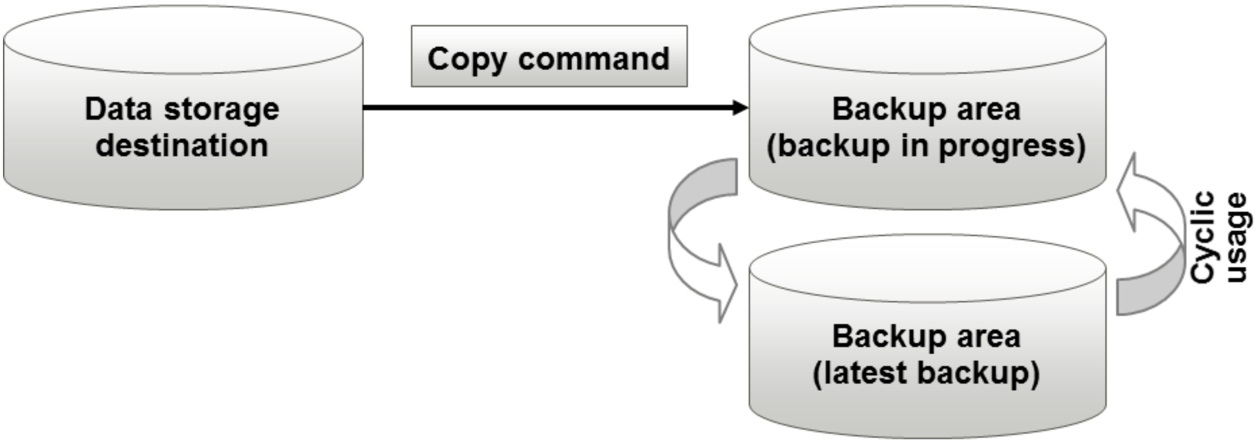
Both the backup data storage destination and the optional backup destination are necessary for recovery - if they are located in secondary media, combined management of these is necessary.

16.1 Configuration of the Copy Command

This section describes the configuration of the copy command for backup and recovery.

Cyclic usage of the backup area

Prepare two backup areas for the copy command in case an issue affects the data storage destination during backup. The copy command performs backup while cyclically using these backup areas.

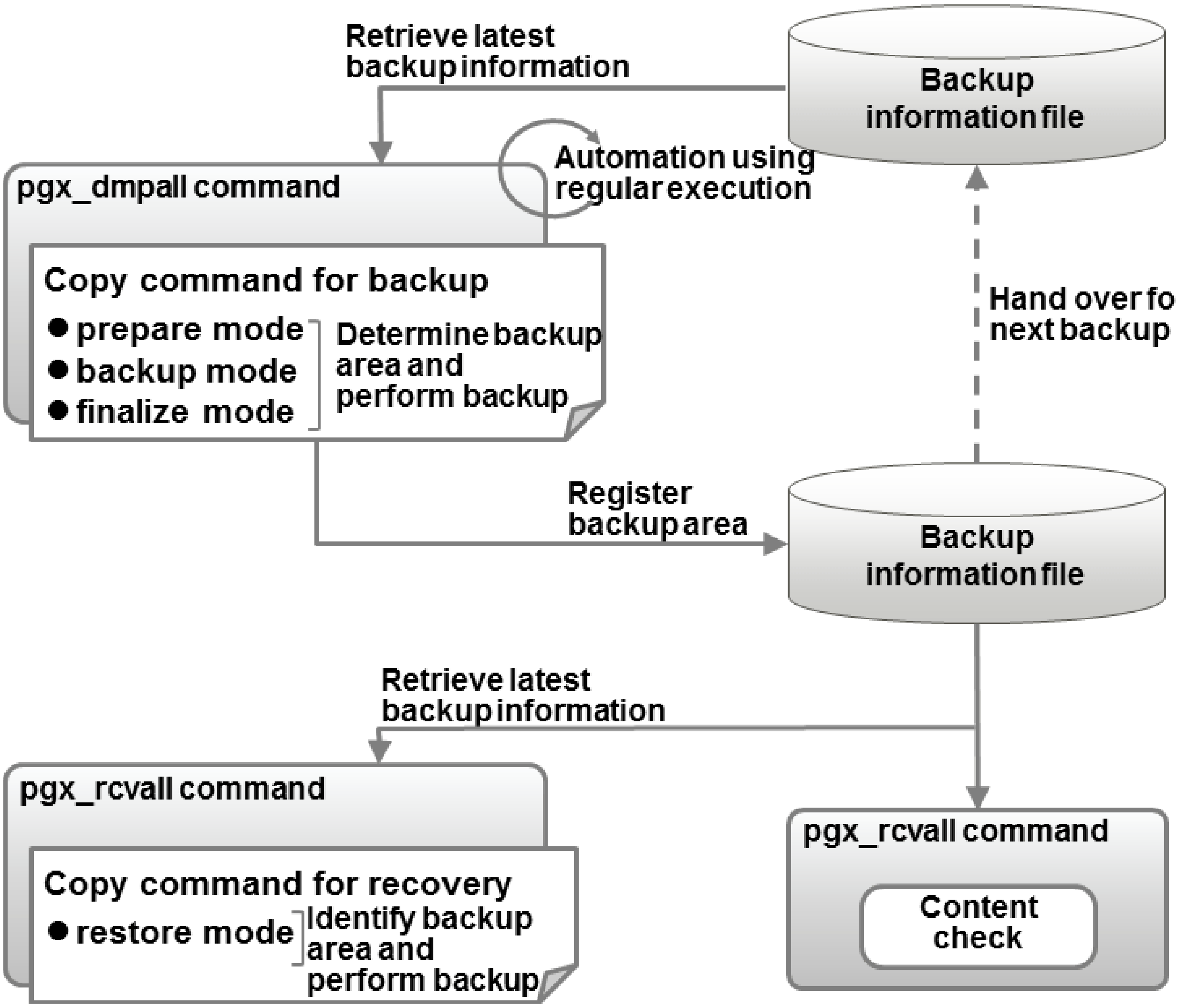


Note

The backup data storage destination cannot be used as these backup areas used by the copy command.

Backup using the backup information file

The copy command must determine the backup destination on each backup, as it is necessary to cycle through the backup areas. Backup can be automated by using the backup information file, which contains information about the backup destination.


Retrieve latest backup information
Backup information file
Automation using regular execution
pgx_dmpall command
Copy command for backup
● prepare mode
● backup mode
● finalize mode
Determine backup area and perform backup
Hand over for next backup
Register backup area
Backup information file
Retrieve latest backup information
pgx_rcvall command
Copy command for recovery
● restore mode
Identify backup area and perform backup
pgx_rcvall command
Content check

Information

The backup information file is prepared in the backup data storage destination by the pgx_dmpall command, and contains information that can be read or updated by the copy command. This file is managed by associating it with the latest backup successfully completed by the pgx_dmpall command, so the latest backup information relating to the copy command registered by the user can be retrieved. Additionally, the content of the backup information file can be displayed using the pgx_rcvall command.

Configuration of the copy command for backup

The pgx_dmpall command calls the copy command for backup after execution for the three modes below. It is therefore necessary for the copy command for backup to implement the required processing for each of the modes.

- prepare mode

  Determines which of the two backup areas will be used for the current backup.

  The backup area to be used for the current backup is determined by reading the information relating to the latest backup destination where the backup information file was written to during the previous backup.

- backup mode

  Performs backup on the backup area determined by prepare mode, using any copy method.

- finalize mode

  Writes information relating to the destination of the current backup to the backup information file.

  This enables the prepare mode to check the destination of the previous backup during the next backup.

Note

The user can use any method to hand over backup information between modes within the copy command, such as creating temporary files.

Configuration of the copy command for recovery

The pgx_rcvall command calls the copy command for recovery for the mode below. It is therefore necessary for the copy command for recovery to implement the required processing for the mode.

- restore mode

  Any copy method can be used to implement restore from the backup destination retrieved using the copy command for backup.

Point

By referring to the mode assigned to the copy command as an argument, backup and recovery can be implemented using a single copy command.

Example

Using a bash script

```
case $1 in
    prepare)
        processingRequiredForPrepareMode
        ;;
    backup)
        processingRequiredForBackupMode
        ;;
    finalize)
         processingRequiredForFinalizeMode
        ;;
    restore)
         processingRequiredForRestoreMode
        ;;
esac
```

Point

A sample batch file that backs up the database cluster and tablespace directory to a specific directory is supplied to demonstrate how to write a copy command.

The sample is stored in the directory below:

```
/installDir/share/copy_command.archive.sh.sample
```

16.2 Backup Using the Copy Command

To perform backup using the copy command, in addition to performing the standard backup procedure, it is also necessary to create a copy command, and then execute the pgx_dmpall command specifying it. This section describes the procedure specific to using the copy command.

Preparing for backup

You must prepare for backup before actually starting the backup process.

Perform the following procedure:

1. Determine the database resources to be backed up

   Determine the database resources to be backed up using the copy command. The copy command can back up the following resources:

   - Database cluster

   - Tablespace

   To back up only some tablespaces, create a file listing them. This file is not necessary to back up all tablespaces.

   Example

   To back up only tablespaces tblspc1 and tblspc2

   ```
   tblspc1
   tblspc2
   ```

2. Prepare a backup area

   Prepare a backup area to save the database resources to be backed up, as determined in step 1.

3. Create the copy command

   Create the copy commands for backup and recovery. Refer to "16.4 Copy Command Interface" for details.

Performing backup

Execute the pgx_dmpall command with the -Y option specifying the full path of the copy command for backup created in step 3 of preparation for backup.

The example below backs up only some tablespaces, but not the database cluster, using the copy command.

Example

```
$ pgx_dmpall -D /database/inst1 -Y '/database/command/backup.sh'
  --exclude-copy-cluster -P '/database/command/tablespace_list.txt'
```

Point

- To exclude up the database cluster from backup using the copy command, specify the --exclude-copy-cluster option.
- To back up only some tablespaces using the copy command, use the -P option specifying the full path of the file created in step 1 of preparation for backup.

See

- Refer to "pgx_dmpall" in the Reference for information on the command.

Checking backup status

Use the pgx_rcvall command to check the backup status.

Execute the pgx_rcvall command with the -l option specified to output backup data information. If backup was performed using the copy command, the resources backed up using the copy command will also be output.

Example

```
$ pgx_rcvall -l -D /database/inst1
Date                Status   Dir                              Resources backed up by the copy command
2022-03-01 13:30:40 COMPLETE /backup/inst1/2022-03-01_13-30-40 pg_data,dbspace,indexspace
```

16.3 Recovery Using the Copy Command

To perform recovery using the copy command, in addition to performing the standard recovery procedure, it is also necessary to create a copy command, and then execute the pgx_rcvall command specifying it. This section describes the procedure specific to using the copy command.

Determining the backup area of the latest backup

Check the backup information file to determine the backup area used for the latest backup, and confirm that it is in a recoverable state.

Execute the pgx_rcvall command with the --view-results-of-copying option to output the content of the backup information file.

Example

```
$ pgx_rcvall -D /database/inst1 --view-results-of-copying
```

Perform recovery

Execute the pgx_rcvall command with the -Y option specifying the full path of the copy command for recovery created in step 3 of the preparation for backup described in "16.2 Backup Using the Copy Command".

The example below recover only some tablespaces, but not the database cluster, using the copy command.

Example

```
$ pgx_rcvall -D /database/inst1 -B /backup/inst1 -Y '/database/command/recovery.sh'
```

Point

If the latest backup was performed using the copy command, the pgx_rcvall command automatically recognizes which database resources were backed up using the copy command, or whether resources were backed up to the backup data storage destination. Therefore, recovery can be performed by simply executing the pgx_rcvall command specifying the copy command for recovery.

See

Refer to "pgx_rcvall" in the Reference for information on the command.

16.4 Copy Command Interface

The following types of copy command are available:

- Copy command for backup
- Copy command for recovery

This appendix describes the interface of each copy command.

16.4.1 Copy Command for Backup

Feature

User command called from the pgx_dmpall command.

Format

The syntax for calling the copy command from the pgx_dmpall command is described below.

If the operation mode is "prepare"

```
copyCommandName prepare 'pathOfBackupInfoFile' 'pathOfBackupTargetListFile'
```

If the operation mode is "backup"

```
copyCommandName backup
```

If the operation mode is "finalize"

```
copyCommandName finalize 'pathOfBackupInfoFile'
```

Argument

- Operation mode

Mode	Description
prepare	Implements the preparation process for backing up using the copy command. Called before the PostgreSQL online backup mode is started.
backup	Implements the backup process. Called during the PostgreSQL online backup mode.

Mode	Description
finalize	Implements the backup completion process. Called after the PostgreSQL online backup mode is completed.

- Full path of the backup information file

  Full path of the backup information file of the latest backup, enclosed in single quotation marks. If a backup has not been performed, specify '-'.

- Full path of the backup target list file

  Full path of the file containing the resources to be backed up using the copy command, enclosed in single quotation marks. One of the following is described in each resource name.

Resource	Description
Database cluster	pg_data
Tablespace	Tablespace name

Example

To back up the database cluster and the tablespaces dbspace and indexspace using the copy command, the file should contain the following:

```
pg_data
dbspace
indexspace
```

Information

The encoding of resource names output to the backup target list file by the pgx_dmpall command is the encoding used when this command connects to the database with auto specified for the client_encoding parameter, and is dependent on the locale at the time of command execution.

The number of arguments vary depending on operation mode. The argument of each operation mode is as follows.

Operation mode	First argument	Second argument	Third argument
prepare	Operation mode	Backup information file path name	Backup target list file path name
backup		None	None
finalize		Backup information file path name	

Additionally, the access permissions for the backup information file and backup target list file are different depending on the operation mode. The access permissions of each operation mode are as follows.

Operation mode	Backup information file	Backup target list file
prepare	Can be viewed by the instance administrator only	Can be viewed by the instance administrator only
backup	-	-
finalize	Can be viewed and updated by the instance administrator only	-

Return value

Return value	Description
0	Normal end The pgx_dmpall command continues processing.
Other than 0	Abnormal end The pgx_dmpall command terminates in error.

Description

- The copy command operates with the privileges of the operating system user who executed the pgx_dmpall command. Therefore, grant copy command execution privileges to users who will execute the pgx_dmpall command. Additionally, have the copy command change users as necessary.
- To write to the backup information file, use a method such as redirection from the copy command.
- Because the copy command is called for each mode, implement all processing for each one.
- To copy multiple resources simultaneously, have the copy command copy them in parallel.

Note

- The backup information file and backup target list file cannot be deleted. Additionally, the privileges cannot be changed.
- Standard output and standard error output of the copy command are output to the terminal where the pgx_dmpall command was executed.
- If the copy command becomes unresponsive, the pgx_dmpall command will also become unresponsive. If the copy command is deemed to be unresponsive by the operating system, use an operating system command to forcibly stop it.
- Output the copy command execution trace and the result to a temporary file, so that if it terminates in error, the cause can be investigated at a later time.
- For prepare mode only, it is possible to use the PostgreSQL client application to access the database using the copy command. For all other modes, do not execute Fujitsu Enterprise Postgres commands or PostgreSQL applications.
- Enable the fsync parameter in postgresql.conf, because data on the shared memory buffer needs to have been already written to disk when backup starts.

16.4.2 Copy Command for Recovery

Feature

User command called from the pgx_rcvall command.

Format

The syntax for calling the copy command from the pgx_rcvall command is described below.

```
copyCommandName restore 'pathOfBackupInfoFile' 'pathOfBackupTargetListFile'
```

Argument

- Operation mode

Mode	Description
restore	Performs restore.

- Full path of the backup information file

  Full path of the backup information file, enclosed in single quotation marks.

- Full path of the backup target list file

  Full path of the file containing the resources to be restored using the copy command, enclosed in single quotation marks.

The access permissions for the backup information file and backup target list file are as below.

Backup information file	Backup target list file
Can be viewed by the instance administrator only	Can be viewed by the instance administrator only

Return value

Return value	Description
0	Normal end The pgx_rcvall command continues processing.
Other than 0	Abnormal end The pgx_rcvall command terminates in error.

Description

- The copy command operates with the privileges of the operating system user who executed the pgx_rcvall command. Therefore, grant copy command execution privileges to users who will execute the pgx_rcvall command. Additionally, have the copy command change users as necessary.
- The copy command is called once only in restore mode.
- To copy multiple resources simultaneously, have the copy command copy them in parallel.

Note

- The backup information file and backup target list file cannot be deleted. Additionally, the privileges cannot be changed.
- Standard output and standard error output of the copy command are output to the terminal where the pgx_rcvall command was executed.
- If the copy command becomes unresponsive, the pgx_rcvall command will also become unresponsive. If the status of the copy command is deemed to be unresponsive by the operating system, use an operating system command to forcibly stop it.
- Output the copy command execution trace and the result to a temporary file, so that if it terminates in error, the cause can be investigated at a later time.
- Do not execute Fujitsu Enterprise Postgres commands or PostgreSQL applications in the copy command.
- There may be files and directories not required for recovery using the archive log included in the backup, such as postmaster.pid, pg_wal/*subdirectory* and pg_replslot in the database cluster. If such unnecessary files and directories exist, have the copy command delete them after the restore.

Chapter 17 Actions when an Error Occurs

This chapter describes the actions to take when an error occurs in the database or an application, while Fujitsu Enterprise Postgres is operating.

Depending on the type of error, it may be necessary to recover the database cluster. The recovery process recovers the following resources:

- Data storage destination
- Transaction log storage destination (if the transaction log is stored in a separate disk from the data storage destination)
- Backup data storage destination

Note

Even if a disk is not defective, the same input-output error messages, as those generated when the disk is defective, may be output. The recovery actions differ for these error messages.

Check the status of the disk, and select one of the following actions:

- If the disk is defective

  Refer to "17.1 Recovering from Disk Failure (Hardware)", and take actions accordingly.

- If the disk is not defective

  Refer to "17.14 I/O Errors Other than Disk Failure", and take actions accordingly.

  A few examples of errors generated even if the disk is not defective include:

  - Network error with an external disk
  - Errors caused by power failure or mounting issues

Determining the cause of an error

If an error occurs, refer to the WebAdmin message and the server log, and determine the cause of the error.

See

Refer to "Configuring Parameters" in the Installation and Setup Guide for Server for information on server logs.

Approximate recovery time

The formulas for deriving the approximate recovery time of resources in each directory are given below.

If using the copy command with the pgx_rcvall command, the recovery time will depend on the implementation of the copy command.

- Data storage destination or transaction log storage destination

```
Recovery time = (usageByTheDataStorageDestination + usageByTheTransactionLogStorageDestination) /
diskWritePerformance x 1.5
```

  - *usageByTheDataStorageDestination*: Disk space used by the database cluster
  - *usageByTheTransactionLogStorageDestination*: Disk space used by the transaction log stored outside the database cluster
  - *diskWritePerformance*: Measured maximum data volume (bytes/second) that can be written per second in the system environment where the operation is performed
  - 1.5: Coefficient assuming the time excluding disk write, which is the most time-consuming step

- Backup data storage destination

```
Recovery time = usageByTheBackupDataStorageDestination / diskWritePerformance x 1.5
```

- *usageByTheBackupDataStorageDestination*: Disk space used by the backup data
- *diskWritePerformance*: Measured maximum data volume (bytes/second) that can be written per second in the system environment where the operation is performed
- 1.5: Coefficient assuming the time excluding disk write, which is the most time-consuming step

17.1 Recovering from Disk Failure (Hardware)

This section describes how to recover database clusters to a point immediately before failure, if a hardware failure occurs in the data storage disk or the backup data storage disk.

There are two methods of recovery:

- 17.1.1 Using WebAdmin
- 17.1.2 Using Server Command

Point

Back up the database cluster after recovering it. Backup deletes obsolete archive logs (transaction logs copied to the backup data storage destination), freeing up disk space and reducing the recovery time.

17.1.1 Using WebAdmin

Recover a failed disk using WebAdmin.

In a streaming replication configuration, the master instance can be recovered, but the standby instance cannot. If disk failure occurs on a standby instance, it may be necessary to delete and re-create the instance. Also, recovering the master instance stops streaming replication to and from all standby instances. In such an event, the standby instances can be promoted to standalone instances or can be deleted and re-created

Recover the database cluster by following the appropriate recovery procedure below for the disk where the failure occurred.

If failure occurred in the data storage disk or the transaction log storage disk

Follow the procedure below to recover the data storage disk or the transaction log storage disk.

1. Stop applications

   Stop applications that are using the database.

2. Stop the instance

   Stop the instance. Refer to "2.1.1 Using WebAdmin" for information on how to stop an instance. WebAdmin automatically stops instances if recovery of the database cluster is performed without stopping the instance.

3. Recover the failed disk

   Replace the disk, and then recover the volume configuration information.

4. Create a tablespace directory

   If a tablespace was defined after backup, create a directory for it.

5. Recover the keystore, and enable automatic opening of the keystore

   Do the following if the data in the database has been encrypted:

   - Restore the keystore to its state at the time of the database backup.
   - Enable automatic opening of the keystore.

6. Recover the database cluster

   Log in to WebAdmin, and in the [Instances] tab, click [Solution] for the error message in the lower-right corner.

7. Run recovery

   In the [Restore Instance] dialog box, click [Yes].

   Instance restore is performed. An instance is automatically started when recovery is successful.

8. Resume applications

   Resume applications that are using the database.

Point

WebAdmin may be unable to detect disk errors, depending on how the error occurred.

If this happens, refer to "17.10.3 Other Errors" to perform recovery.

If failure occurred on the backup data storage disk

Follow the procedure below to recover the backup data storage disk.

1. Recover the failed disk

   Replace the disk, and then recover the volume configuration information.

2. Recover the backup data

   Log in to WebAdmin, and in the [Instances] tab, click [Solution] for the error message.

3. Run backup

   Perform backup to enable recovery of the backup data. In the [Backup] dialog box, click [Yes]. The backup is performed. An instance is automatically started when backup is performed.

Point

If you click [Recheck the status], the resources in the data storage destination and the backup data storage destination are reconfirmed. As a result, the following occurs:

- If an error is not detected

  The status of the data storage destination and the backup data storage destination returns to normal, and it is possible to perform operations as usual.

- If an error is detected

  An error message is displayed in the message list again. Click [Solution], and resolve the problem by following the resolution for the cause of the error displayed in the dialog box.

17.1.2 Using Server Command

Recover the database cluster by following the appropriate recovery procedure below for the disk where the failure occurred.

If failure occurred on the data storage disk or the transaction log storage directory

Follow the procedure below to recover the data storage disk or the transaction log storage directory.

1. Stop applications

   Stop applications that are using the database.

2. Stop the instance

   Stop the instance, refer to "2.1.2 Using Server Commands" for details.

   If the instance fails to stop, refer to "17.11 Actions in Response to Failure to Stop an Instance".

3. Recover the failed disk

   Replace the disk, and then recover the volume configuration information.

4. Create a storage destination directory

   - If failure occurred on the data storage disk
     Create a data storage destination directory. If a tablespace was defined, also create a directory for it.

   - If failure occurred on the translation log storage disk
     Create a transaction log storage destination directory.

   Example

   To create a data storage destination directory:

   ```
   $ mkdir /database/inst1
   $ chown fsepuser:fsepuser /database/inst1
   $ chmod 700 /database/inst1
   ```

   See

   Refer to "Preparing Directories to Deploy Resources" under "Setup" in the Installation and Setup Guide for Server for information on how to create a storage directory.

5. Recover the keystore, and enable automatic opening of the keystore

   When the data in the database has been encrypted, restore the keystore to its state at the time of the database backup. Configure automatic opening of the keystore as necessary.

6. Recover the database cluster

   Recover the database cluster using the backup data.

   Specify the following in the pgx_rcvall command:

   - Specify the data storage location in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

   - Specify the backup data storage location in the -B option.

   Example

   ```
   > pgx_rcvall -D /database/inst1 -B /backup/inst1
   ```

   Note

   If recovery fails, remove the cause of the error in accordance with the displayed error message and then re-execute the pgx_rcvall command.

   If the message "pgx_rcvall: an error occurred during recovery" is displayed, then the log recorded when recovery was executed is output after this message. The cause of the error is output in around the last fifteen lines of the log, so remove the cause of the error in accordance with the message and then re-execute the pgx_rcvall command.

   The following message displayed during recovery is output as part of normal operation of pgx_rcvall command (therefore the user does not need not be concerned).

   ```
   FATAL: the database system is starting up
   ```

7. Start the instance

   Start the instance.

   Refer to "2.1.2 Using Server Commands" for information on how to start an instance.

8. Resume applications

   Resume applications that are using the database.

If failure occurred on the backup data storage disk

The procedure for recovering the backup data storage disk is described below.

There are two methods of taking action:

- Performing recovery while the instance is active
- Stopping the instance before performing recovery

The following table shows the different steps to be performed depending on whether you stop the instance.

No	Step	Instance stopped	
		No	Yes
1	Confirm that transaction log mirroring has stopped	Y	N
2	Stop output of archive logs	Y	N
3	Stop applications	N	Y
4	Stop the instance	N	Y
5	Recover the failed disk	Y	Y
6	Create a backup data storage destination directory	Y	Y
7	Resume output of archive logs	Y	N
8	Resume transaction log mirroring	Y	N
9	Start the instance	N	Y
10	Run backup	Y	Y
11	Resume applications	N	Y

Y: Required
N: Not required

The procedure is as follows:

If an instance has not been stopped

1. Confirm that transaction log mirroring has stopped

   Use the following SQL function to confirm that transaction log mirroring has stopped.

```
 postgres=# SELECT pgx_is_wal_multiplexing_paused();
 pgx_is_wal_multiplexing_paused
-------------------
 t
(1 row)
```

   If transaction log mirroring has not stopped, then stop it using the following SQL function.

```
postgres=# SELECT pgx_pause_wal_multiplexing();
LOG:  multiplexing of transaction log files has been stopped
pgx_pause_wal_multiplexing
----------------------------

(1 row)
```

2. Stop output of archive logs

   Transaction logs may accumulate during replacement of backup storage disk, and if the data storage disk or the transaction log storage disk becomes full, there is a risk that operations may not be able to continue.

   To prevent this, use the following methods to stop output of archive logs.

   - Changing archive_command

     Specify a command that will surely complete normally, such as "echo skipped archiving WAL file %f" or "/bin/true", so that archive logs will be regarded as having been output.

     If you specify echo, a message is output to the server log, so it may be used as a reference when you conduct investigations.

   - Reload the configuration file

     Execute the pg_ctl reload command or the pg_reload_conf SQL function to reload the configuration file.

   If you simply want to stop output of errors without the risk that operations will not be able to continue, specify an empty string ('') in archive_command and reload the configuration file.

3. Recover the failed disk

   Replace the disk, and then recover the volume configuration information.

4. Create a backup data storage destination

   Create a backup data storage destination.

   Example

   ```
   $ mkdir /database/inst1
   $ chown fsepuser:fsepuser /database/inst1
   $ chmod 700 /database/inst1
   ```

   Refer to "3.2.2 Using Server Commands" for information on how to create a backup data storage destination.

5. Resume output of archive logs

   Return the archive_command setting to its original value, and reload the configuration file.

6. Resume transaction log mirroring

   Execute the pgx_resume_wal_multiplexing SQL function.

   Example

   ```
   SELECT pgx_resume_wal_multiplexing()
   ```

7. Run backup

   Use the pgx_dmpall command to back up the database cluster.

   Specify the following value in the pgx_dmpall command:

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

   Example

```
> pgx_dmpall -D /database/inst1
```

If an instance has been stopped

1. Stop applications

   Stop applications that are using the database.

2. Stop the instance

   Stop the instance. Refer to "2.1.2 Using Server Commands" for details.

If the instance fails to stop, refer to "17.11 Actions in Response to Failure to Stop an Instance".

3. Recover the failed disk

   Replace the disk, and then recover the volume configuration information.

4. Create a backup data storage destination

   Create a backup data storage destination.

   Example

   ```
   # mkdir /backup/inst1
   # chown fsepuser:fsepuser /backup/inst1
   # chmod 700 /backup/inst1
   ```

   Refer to "3.2.2 Using Server Commands" for details.

5. Start the instance

   Start the instance. Refer to "2.1.2 Using Server Commands" for information on how to start an instance.

6. Run backup

   Use the pgx_dmpall command to back up the database cluster.

   Specify the following value in the pgx_dmpall command:

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

   Example

   ```
   > pgx_dmpall -D /database/inst1
   ```

7. Resume applications

   Resume applications that are using the database.

See

- Refer to "pgx_rcvall" and "pgx_dmpall" in the Reference for information on the pgx_rcvall command and pgx_dmpall command.
- Refer to "Write Ahead Log" under "Server Administration" in the PostgreSQL Documentation for information on archive_command.
- Refer to "B.1 WAL Mirroring Control Functions" for information on pgx_resume_wal_multiplexing.

17.2 Recovering from Data Corruption

If data in a disk is logically corrupted and the database does not operate properly, you can recover the database cluster to its state at the time of backup.

There are two methods of recovery:

- 17.2.1 Using WebAdmin
- 17.2.2 Using the pgx_rcvall Command

Point

- Back up the database cluster after recovering it. Backup deletes obsolete archive logs (transaction logs copied to the backup data storage destination), freeing up disk space and reducing the recovery time.
- If you recover data to a point in the past, a new time series (database update history) will start from that recovery point. When recovery is complete, the recovery point is the latest point in the new time series. When you subsequently recover data to the latest state, the database update is re-executed on the new time series.

17.2.1 Using WebAdmin

If using WebAdmin, recover the data to the point immediately prior to data corruption by using the backup data.

Refer to "17.1.1 Using WebAdmin" for details.

17.2.2 Using the pgx_rcvall Command

Recover the database cluster by specifying in the pgx_rcvall command the date and time of the backup you want to read from. Then re-execute the transaction as required to recover the data.

Follow the procedure below to recover the data storage disk.

1. Stop applications

   Stop applications that are using the database.

2. Stop the instance

   Stop the instance. Refer to "2.1.2 Using Server Commands" for information on how to stop an instance.

   If the instance fails to stop, refer to "17.11 Actions in Response to Failure to Stop an Instance".

3. Confirm the backup date and time

   Execute the pgx_rcvall command to confirm the backup data saved in the backup data storage destination, and determine a date and time prior to data corruption.

   Specify the following values in the pgx_rcvall command:

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.
   - Specify the backup storage directory in the -B option.
   - The -l option displays the backup data information.

   Example

   ```
   > pgx_rcvall -D /database/inst1 -B /backup/inst1 -l
   Date                    Status          Dir
   2022-03-20 10:00:00     COMPLETE        /backup/inst1/2022-03-20_10-00-00
   ```

4. Recover the keystore, and enable automatic opening of the keystore

   When the data in the database has been encrypted, restore the keystore to its state at the time of the database backup. Configure automatic opening of the keystore as necessary.

5. Recover the database cluster

   Use the pgx_rcvall command to recover the database cluster.

   Specify the following values in the pgx_rcvall command:

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.
   - Specify the backup storage directory in the -B option.
   - Specify the recovery date and time in the -e option.

   Example

   In the following examples, "March 20, 2022 10:00:00" is specified as the recovery time.

   ```
   > pgx_rcvall -D /database/inst1 -B /backup/inst1 -e '2022-03-20 10:00:00'
   ```

Note

If recovery fails, remove the cause of the error in accordance with the displayed error message and then re-execute the pgx_rcvall command.

If the message "pgx_rcvall: an error occurred during recovery" is displayed, then the log recorded when recovery was executed is output after this message. The cause of the error is output in around the last fifteen lines of the log, so remove the cause of the error in accordance with the message and then re-execute the pgx_rcvall command.

The following message displayed during recovery is output as part of normal operation of pgx_rcvall command (therefore the user does not need not be concerned).

```
FATAL: the database system is starting up
```

6. Start the instance

   Start the instance. Refer to "2.1.2 Using Server Commands" for information on how to start an instance.

   If necessary, re-execute transaction processing from the specified recovery time, and then resume database operations.

7. Resume applications

   Resume applications that are using the database.

See

Refer to "pgx_rcvall" in the Reference for information on the pgx_rcvall command.

17.3 Recovering from an Incorrect User Operation

This section describes how to recover database clusters when data has been corrupted due to erroneous user operations.

There are two methods of recovery:

- 17.3.1 Using WebAdmin
- 17.3.2 Using the pgx_rcvall Command

Point

- Back up the database cluster after recovering it. Backup deletes obsolete archive logs (transaction logs copied to the backup data storage destination), freeing up disk space and reducing the recovery time.
- If you recover data to a point in the past, a new time series (database update history) will start from that recovery point. When recovery is complete, the recovery point is the latest point in the new time series. When you subsequently recover data to the latest state, the database update is re-executed on the new time series.
- An effective restore point is one created on a time series for which you have made a backup. That is, if you recover data to a point in the past, you cannot use any restore points set after that recovery point. Therefore, once you manage to recover your target past data, make a backup.

17.3.1 Using WebAdmin

Recover data to a backup point using WebAdmin.

In a streaming replication configuration, the master instance can be recovered, but the standby instance cannot. If disk failure occurs on a standby instance, it may be necessary to delete and re-create the instance. Also, recovering the master instance stops streaming replication to and from all standby instances. In such an event, the standby instances can be promoted to standalone instances or can be deleted and re-created

Follow the procedure below to recover the data in the data storage disk.

1. Stop applications

   Stop applications that are using the database.

2. Stop the instance

   Stop the instance. Refer to "2.1.1 Using WebAdmin" for information on how to stop an instance.

3. Recover the keystore, and enable automatic opening of the keystore

   Do the following if the data in the database has been encrypted:

   - Restore the keystore to its state at the time of the database backup.

   - Enable automatic opening of the keystore.

4. Recover the database cluster

   Log in to WebAdmin, and in the [Instances] tab, select the instance to be recovered and click .

5. Recover to the backup point

   In the [Restore Instance] dialog box, click [Yes].

   Recovery is performed. An instance is automatically started when recovery is successful.

6. Resume database operations

   If necessary, re-execute transaction processing from the backup point to when an erroneous operation was performed, and then resume database operations.

17.3.2 Using the pgx_rcvall Command

The pgx_rcvall command recovers database clusters to the restore point created with the server command. Refer to "Setting a restore point" in "3.2.2 Using Server Commands" for information on how to create a restore point.

Follow the procedure below to recover the data in the data storage disk.

1. Stop applications

   Stop applications that are using the database.

2. Stop the instance

   Stop the instance. Refer to "2.1.2 Using Server Commands" for information on how to stop an instance.

   If the instance fails to stop, refer to "17.11 Actions in Response to Failure to Stop an Instance".

3. Confirm the restore point

   Execute the pgx_rcvall command to confirm the backup data saved in the backup data storage destination, and use a restore point recorded in an arbitrary file, as explained in "3.2.2 Using Server Commands", to determine a restore point prior to the erroneous operation.

   Specify the following values in the pgx_rcvall command:

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

   - Specify the backup data storage destination in the -B option.

   - The -l option displays the backup data information.

   Example

   ```
   > pgx_rcvall -D /database/inst1 -B /backup/inst1 -l
   Date                    Status          Dir
   2022-03-01 10:00:00     COMPLETE        /backup/inst1/2022-03-01_10-00-00
   ```

4. Recover the keystore, and enable automatic opening of the keystore

   When the data in the database has been encrypted, restore the keystore to its state at the time of the database backup. Configure automatic opening of the keystore as necessary.

5. Recover the database cluster

   Use the pgx_rcvall command to recover the database cluster.

   Specify the following values in the pgx_rcvall command:

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

   - Specify the backup data storage destination in the -B option.

   - The -n option recovers the data to the specified restore point.

   Example

   The following example executes the pgx_rcvall command with the restore point "batch_20220303_1".

   ```
   > pgx_rcvall -D /database/inst1 -B /backup/inst1 -n batch_20220303_1
   ```

   Note

   If recovery fails, remove the cause of the error in accordance with the displayed error message and then re-execute the pgx_rcvall command.

   If the message "pgx_rcvall: an error occurred during recovery" is displayed, then the log recorded when recovery was executed is output after this message. The cause of the error is output in around the last fifteen lines of the log, so remove the cause of the error in accordance with the message and then re-execute the pgx_rcvall command.

   The following message displayed during recovery is output as part of normal operation of pgx_rcvall (therefore the user does not need not be concerned).

   ```
   FATAL: the database system is starting up
   ```

6. Start the instance

   Start the instance.

   Refer to "2.1.2 Using Server Commands" for information on how to start an instance.

7. Restart operation of the database

   If necessary, re-execute transaction processing from the specified recovery time to the point when an erroneous operation was performed, and then resume database operations.

See

Refer to "pgx_rcvall" in the Reference for information on the pgx_rcvall command.

17.4 Actions in Response to an Application Error

If there is a connection from a client that has been in the waiting state for an extended period, you can minimize performance degradation of the database by closing the problematic connection.

The following methods are available for identifying a connection to be closed:

- view(pg_stat_activity) (refer to "17.4.1 When using the view (pg_stat_activity)")

- ps command (refer to "17.4.2 Using the ps Command")

Use the system management function (pg_terminate_backend) to disconnect connections.

17.4.1 When using the view (pg_stat_activity)

When using the view (pg_stat_activity), follow the procedure below to close a connection.

1. Use psql command to connect to the postgres database.

```
> psql postgres
psql (<x>) (*1)
Type "help" for help.
```

*1: <*x*> indicates the PostgreSQL version on which this product is based.

2. Close connections from clients that have been in the waiting state for an extended period.

   Use pg_terminate_backend() to close connections that have been trying to connect for an extended period.

   However, when considering continued compatibility of applications, do not reference or use system catalogs and functions directly in SQL statements. Refer to " Notes on Application Compatibility" in the Application Development Guide for details.

   Example

   The following example closes connections where the client has been in the waiting state for at least 60 minutes.

```
select pid,usename,application_name,client_addr,pg_terminate_backend(pid) from pg_stat_activity
where backend_type = 'client backend' and state='idle in transaction' and current_timestamp >
cast(query_start + interval '60 minutes' as timestamp);
-[ RECORD 1 ]--------+---------------
pid                  | 4684
username             | fsepuser
application_name     | apl1
client_addr          | 192.11.11.1
pg_terminate_backend | t
```

See

- Refer to "System Administration Functions" under "The SQL Language" in the PostgreSQL Documentation for information on pg_terminate_backend.

- Refer to "Notes on Application Compatibility" in the Application Development Guide for information on how to maintain application compatibility.

17.4.2 Using the ps Command

Follow the procedure below to close a connection using a standard Unix tool (ps command).

1. Execute the ps command.
   Note that "<*x*>" indicates the product version.

```
> ps axwfo user,pid,ppid,tty,command | grep postgres
fsepuser 19174 18027 pts/1                 \_ grep postgres
fsepuser 20517     1 ?      /opt/fsepv<x>server64/bin/postgres -D /disk01/data
fsepuser 20518 20517 ?       \_ postgres: logger
fsepuser 20520 20517 ?       \_ postgres: checkpointer
fsepuser 20521 20517 ?       \_ postgres: background writer
fsepuser 20522 20517 ?       \_ postgres: walwriter
fsepuser 20523 20517 ?       \_ postgres: autovacuum launcher
fsepuser 20524 20517 ?       \_ postgres: archiver
fsepuser 20525 20517 ?       \_ postgres: logical replication launcher
fsepuser 18673 20517 ?       \_ postgres: fsepuser postgres 192.168.100.1(49448) idle
fsepuser 16643 20517 ?       \_ postgres: fsepuser db01 192.168.100.11(49449) UPDATE waiting
fsepuser 16644 20517 ?       \_ postgres: fsepuser db01 192.168.100.12(49450) idle in transaction
```

   Process ID 16643 may be a connection that was established a considerable time ago by the UPDATE statement, or a connection that has occupied resources (waiting).

2. Close connections from clients that have been in the waiting state for an extended period.

   Use pg_terminate_backend() to close the connection with the process ID identified in step 1 above.

   The example below disconnects the process with ID 16643.

   However, when considering continued compatibility of applications, do not reference or use system catalogs and functions directly in SQL statements.

```
postgres=# SELECT pg_terminate_backend (16643);
 pg_terminate_backend
-------------------
t
(1 row)
```

See

- Refer to "System Administration Functions" under "The SQL Language" in the PostgreSQL Documentation for information on pg_terminate_backend.
- Refer to "Notes on Application Compatibility" in the Application Development Guide for information on how to maintain application compatibility.

17.5 Actions in Response to an Access Error

If access is denied, grant privileges allowing the instance administrator to operate the following directories, and then re-execute the operation. Also, refer to the event log and the server log, and confirm that the file system has not been mounted as read-only due to a disk error. If the file system has been mounted as read-only, mount it properly and then re-execute the operation.

- Data storage destination
- Tablespace storage destination
- Transaction log storage destination
- Backup data storage destination

See

Refer to "Preparing Directories to Deploy Resources" under "Setup" in the Installation and Setup Guide for Server for information on the privileges required for the directory.

17.6 Actions in Response to Insufficient Space on the Data Storage Destination

If the data storage destination runs out of space, check if the disk contains any unnecessary files and delete them so that operations can continue.

If deleting unnecessary files does not solve the problem, you must migrate data to a disk with larger capacity.

There are two methods of migrating data:

- 17.6.1 Using a Tablespace
- 17.6.2 Replacing the Disk with a Larger Capacity Disk

17.6.1 Using a Tablespace

Fujitsu Enterprise Postgres enables you to use a tablespace to change the storage destination of database objects, such as tables and indexes, to a different disk.

The procedure is as follows:

1. Create a tablespace

   Use the CREATE TABLESPACE command to create a new tablespace in the prepared disk.

2. Modify the tablespace

   Use the ALTER TABLE command to modify tables for the newly defined tablespace.

See

Refer to "SQL Commands" under "Reference" in the PostgreSQL Documentation for information on the CREATE TABLESPACE command and ALTER TABLE command.

17.6.2 Replacing the Disk with a Larger Capacity Disk

Before replacing the disk with a larger capacity disk, migrate resources at the data storage destination using the backup and recovery features.

There are two methods of performing backup and recovery:

- 17.6.2.1 Using WebAdmin
- 17.6.2.2 Using Server Commands

The following sections describe procedures that use each of these methods to replace the disk and migrate resources at the data storage destination.

Point

It is recommended that you back up the database cluster following recovery. Backup deletes obsolete archive logs (transaction logs copied to the backup data storage destination), freeing up disk space and reducing the recovery time.

17.6.2.1 Using WebAdmin

Follow the procedure below to replace the disk and migrate resources at the data storage destination by using WebAdmin.

1. Back up files

   If the disk at the data storage destination contains any required files, back up the files. It is not necessary to back up the data storage destination.

2. Stop applications

   Stop applications that are using the database.

3. Back up the database cluster

   Back up the latest resources at the data storage destination. Refer to "3.2.1 Using WebAdmin" for details.

4. Stop the instance

   Stop the instance. Refer to "2.1.1 Using WebAdmin" for information on how to stop an instance.

5. Replace with a larger capacity disk

   Replace the disk. Then, recover the volume configuration information.

6. Recover the database cluster

   Log in to WebAdmin, and perform recovery operations. Refer to steps 4 ("Create a tablespace directory ") to 7 ("Run recovery") under "If failure occurred in the data storage disk or the transaction log storage disk" in "17.1.1 Using WebAdmin" for information on the procedure. An instance is automatically started when recovery is successful.

7. Resume applications

   Resume applications that are using the database.

8. Restore the files

   Restore the files backed up in step 1.

17.6.2.2 Using Server Commands

Follow the procedure below to replace the disk and migrate resources at the data storage destination by using server commands.

1. Back up files

   If the disk at the data storage destination contains any required files, back up the files. It is not necessary to back up the data storage destination.

2. Stop applications

   Stop applications that are using the database.

3. Back up the database cluster

   Back up the latest resources at the data storage destination. Refer to "3.2.2 Using Server Commands" for details.

4. Stop the instance

   After backup is complete, stop the instance. Refer to "2.1.2 Using Server Commands" for information on how to stop an instance.

   If the instance fails to stop, refer to "17.11 Actions in Response to Failure to Stop an Instance".

5. Replace with a larger capacity disk

   Replace the disk. Then, recover the volume configuration information.

6. Create a data storage destination

   Create a data storage destination. If a tablespace was defined, also create a directory for it.

   Example

   ```
   $ mkdir /database/inst1
   $ chown fsepuser:fsepuser /database/inst1
   $ chmod 700 /database/inst1
   ```

7. Recover the keystore, and enable automatic opening of the keystore

   When the data in the database has been encrypted, restore the keystore to its state at the time of the database backup. Configure automatic opening of the keystore as necessary.

8. Recover the database cluster

   Use the pgx_rcvall command to recover the database cluster.

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

   - Specify the backup storage directory in the -B option.

   Example

   ```
   > pgx_rcvall -D /database/inst1 -B /backup/inst1
   ```

   Note

   If recovery fails, remove the cause of the error in accordance with the displayed error message and then re-execute the pgx_rcvall command.

   If the message "pgx_rcvall: an error occurred during recovery" is displayed, then the log recorded when recovery was executed is output after this message. The cause of the error is output in around the last fifteen lines of the log, so remove the cause of the error in accordance with the message and then re-execute the pgx_rcvall command.

The following message displayed during recovery is output as part of normal operation of pgx_rcvall (therefore the user does not need not be concerned).

```
FATAL: the database system is starting up
```

See

Refer to "pgx_rcvall" in the Reference for information on the pgx_rcvall command.

9. Start the instance

   Start the instance.

   Refer to "2.1.2 Using Server Commands" for information on how to start an instance.

10. Resume applications

    Resume applications that are using the database.

11. Restore files

    Restore the files backed up in step 1.

17.7 Actions in Response to Insufficient Space on the Backup Data Storage Destination

If space runs out on the backup data storage destination, check if the disk contains any unnecessary files and delete them, and then make a backup as required.

If deleting unnecessary files does not solve the problem, take the following action:

- 17.7.1 Temporarily Saving Backup Data
- 17.7.2 Replacing the Disk with a Larger Capacity Disk

17.7.1 Temporarily Saving Backup Data

This method involves temporarily moving backup data to a different directory, saving it there, and securing disk space on the backup data storage destination so that a backup can be made normally.

Use this method if you need time to prepare a larger capacity disk.

If space runs out on the backup data storage destination, archive logs can no longer be stored in the backup data storage destination. As a result, transaction logs continue to accumulate in the data storage destination or the transaction log storage destination.

If action is not taken soon, the transaction log storage destination will become full, and operations may not be able to continue.

To prevent this, secure space in the backup data storage destination, so that archive logs can be stored.

There are two methods of taking action:

- 17.7.1.1 Using WebAdmin
- 17.7.1.2 Using Server Commands

17.7.1.1 Using WebAdmin

Follow the procedure below to recover the backup data storage disk.

1. Temporarily save backup data

   Move backup data to a different directory and temporarily save it, and secure space in the backup data storage destination directory.

The reason for saving the backup data is so that the data in the data storage destination can be recovered even if it is corrupted before you perform recovery. If there is no disk at the save destination and you consider that there is no risk of corruption at the data storage destination, delete the backup data.

The following example saves backup data from the backup data storage destination directory (/backup/inst1) under /mnt/usb/backup.

Example

```
> mkdir /mnt/usb/backup/
> mv /backup/inst1/* /mnt/usb/backup/
```

2. Back up the database cluster

   Back up the latest resources at the data storage destination. Refer to "3.2.1 Using WebAdmin" for details.

3. Delete temporarily saved backup data

   If backup completes normally, the temporarily saved backup data becomes unnecessary and is deleted.

   The following example deletes backup data that was temporarily saved in /mnt/usb.

   Example

```
> rm -rf /mnt/usb/backup
```

17.7.1.2 Using Server Commands

The following describes the procedure for recovering the backup storage disk.

There are two methods of taking action:

- Performing recovery while the instance is active
- Stopping the instance before performing recovery

The following table shows the different steps to be performed depending on whether you stop the instance.

No	Step	Instance stopped	
		No	Yes
1	Stop transaction log mirroring	Y	N
2	Stop output of archive logs	Y	N
3	Stop applications	N	Y
4	Stop the instance	N	Y
5	Temporarily save backup data	Y	Y
6	Resume output of archive logs	Y	N
7	Resume transaction log mirroring	Y	N
8	Start an instance	N	Y
9	Run backup	Y	Y
10	Resume applications	N	Y
11	Delete temporarily saved backup data	Y	Y

Y: Required
N: Not required

The procedure is as follows:

Performing recovery while the instance is active

1. Stop transaction log mirroring

   Stop transaction log mirroring.

   ```
   postgres=# SELECT pgx_pause_wal_multiplexing();
   LOG:  multiplexing of transaction log files has been stopped
   pgx_pause_wal_multiplexing
   ----------------------------

   (1 row)
   ```

2. Stop output of archive logs

   Transaction logs may accumulate during replacement of backup storage disk, and if the data storage disk or the transaction log storage disk becomes full, there is a risk that operations may not be able to continue.

   To prevent this, use the following methods to stop output of archive logs.

   - Changing the archive_command parameter

     Specify a command that will surely complete normally, such as "echo skipped archiving WAL file %f" or "/bin/true", so that archive logs will be regarded as having been output.

     If you specify echo, a message is output to the server log, so it may be used as a reference when you conduct investigations.

   - Reloading the configuration file

     Run the pg_ctl reload command or the pg_reload_conf SQL function.

   If you simply want to stop output of errors without the risk that operations will not be able to continue, specify an empty string ('') in archive_command and reload the configuration file.

3. Temporarily save backup data

   Move backup data to a different directory and temporarily save it, and secure space in the backup data storage destination directory.

   The reason for saving the backup data is so that the data in the data storage destination can be recovered even if it is corrupted before you perform the next step. If there is no disk at the save destination and you consider that there is no risk of corruption at the data storage destination, delete the backup data.

   The following example saves backup data from the backup data storage destination directory (/backup/inst1) under /mnt/usb/backup.

   Example

   ```
   > mkdir /mnt/usb/backup/
   > mv /backup/inst1/* /mnt/usb/backup/
   ```

4. Resume output of archive logs

   Return the archive_command setting to its original value, and reload the configuration file.

5. Resume transaction log mirroring

   Execute the pgx_resume_wal_multiplexing SQL function.

   Example

   ```
   SELECT pgx_resume_wal_multiplexing()
   ```

6. Run backup

   Use the pgx_dmpall command to back up the database cluster.

   Specify the following option in the pgx_dmpall command:

   - Specify the directory of the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

     Example

```
> pgx_dmpall -D /database/inst1
```

7. Delete temporarily saved backup data

   If backup completes normally, the temporarily saved backup data becomes unnecessary and is deleted.

   The following example deletes backup data that was temporarily saved in /mnt/usb.

   Example

   ```
   > rm -rf /mnt/usb/backup
   ```

If an instance has been stopped

1. Stop applications

   Stop applications that are using the database.

2. Stop the instance

   Stop the instance. Refer to "2.1.2 Using Server Commands" for details.

   If the instance fails to stop, refer to "17.11 Actions in Response to Failure to Stop an Instance".

3. Temporarily save backup data

   Move backup data to a different directory and temporarily save it, and secure space in the backup data storage destination directory.

   The reason for saving the backup data is so that the data in the data storage destination can be recovered even if it is corrupted before you perform recovery. If there is no disk at the save destination and you consider that there is no risk of corruption at the data storage destination, delete the backup data.

   The following example saves backup data from the backup data storage destination directory (/backup/inst1) under /mnt/usb/backup.

   Example

   ```
   > mkdir /mnt/usb/backup/
   > mv /backup/inst1/* /mnt/usb/backup/
   ```

4. Start the instance

   Start the instance. Refer to "2.1.2 Using Server Commands" for information on how to start an instance.

5. Run backup

   Use the pgx_dmpall command to back up the database cluster.

   Specify the following value in the pgx_dmpall command:

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

     Example

     ```
     > pgx_dmpall -D /database/inst1
     ```

6. Resume applications

   Resume applications that are using the database.

7. Delete temporarily saved backup data

   If backup completes normally, the temporarily saved backup data becomes unnecessary and is deleted.

   The following example deletes backup data that was temporarily saved in /mnt/usb.

   Example

   ```
   > rm -rf /mnt/usb/backup
   ```

See

- Refer to "pgx_rcvall" and "pgx_dmpall" in the Reference for information on the pgx_rcvall command and pgx_dmpall command.
- Refer to "Write Ahead Log" under "Server Administration" in the PostgreSQL Documentation for information on archive_command.
- Refer to "B.1 WAL Mirroring Control Functions" for information on the pgx_is_wal_multiplexing_paused and pgx_resume_wal_multiplexing.

17.7.2 Replacing the Disk with a Larger Capacity Disk

This method involves replacing the disk at the backup data storage destination with a larger capacity disk, so that it does not run out of free space again. After replacing the disk, back up data to obtain a proper backup.

There are two methods of performing backup:

- 17.7.2.1 Using WebAdmin
- 17.7.2.2 Using Server Commands

17.7.2.1 Using WebAdmin

Follow the procedure below to recover the backup storage disk.

1. Back up files

   If the disk at the backup data storage destination contains any required files, back up the files. It is not necessary to back up the backup data storage destination.

2. Temporarily save backup data

   Save the backup data to a different directory.

   The reason for saving the backup data is so that the data in the data storage destination can be recovered even if it is corrupted before you perform the next step. If there is no disk at the save destination and you consider that there is no risk of corruption at the data storage destination, delete the backup data.

   The following example saves backup data from the backup data storage destination directory (/backup/inst1) under /mnt/usb/backup.

   Example

   ```
   > mkdir /mnt/usb/backup/
   > mv /backup/inst1/* /mnt/usb/backup/
   ```

3. Stop applications

   Stop applications that are using the database.

4. Replace with a larger capacity disk

   Replace the disk. Then, recover the volume configuration information.

5. Run backup

   Log in to WebAdmin, and perform recovery operations. Refer to steps 2 ("Recover the backup data") and 3 ("Run backup") under "If failure occurred on the backup storage disk" in "17.1.1 Using WebAdmin".

6. Restore files

   Restore the files backed up in step 1.

7. Delete temporarily saved backup data

   If backup completes normally, the temporarily saved backup data becomes unnecessary and is deleted.

   The following example deletes backup data that was temporarily saved in /mnt/usb.

Example

```
> rm -rf /mnt/usb/backup
```

17.7.2.2 Using Server Commands

The procedure for recovering the backup data storage disk is described below.

There are two methods of taking action:

- Performing recovery while the instance is active
- Stopping the instance before performing recovery

The following table shows the different steps to be performed depending on whether you stop the instance.

No	Step	Instance stopped	
		No	Yes
1	Back up files	Y	Y
2	Temporarily save backup data	Y	Y
3	Confirm that transaction log mirroring has stopped	Y	N
4	Stop output of archive logs	Y	N
5	Stop applications	N	Y
6	Stop the instance	N	Y
7	Replace with a larger capacity disk	Y	Y
8	Create a backup storage directory	Y	Y
9	Resume output of archive logs	Y	N
10	Resume transaction log mirroring	Y	N
11	Start the instance	N	Y
12	Run backup	Y	Y
13	Resume applications	N	Y
14	Restore files	Y	Y
15	Delete temporarily saved backup data	Y	Y

Y: Required
N: Not required

The procedure is as follows:

If an instance has not been stopped

1. Back up files

   If the disk at the backup data storage destination contains any required files, back up the files. It is not necessary to back up the backup data storage destination.

2. Temporarily save backup data

   Save the backup data to a different directory.

   The reason for saving the backup data is so that the data in the data storage destination can be recovered even if it is corrupted before you perform the next step. If there is no disk at the save destination and you consider that there is no risk of corruption at the data storage destination, delete the backup data.

   The following example saves backup data from the backup data storage destination directory (/backup/inst1) under /mnt/usb/backup.

Example

```
> mkdir /mnt/usb/backup/
> mv /backup/inst1/* /mnt/usb/backup/
```

3. Confirm that transaction log mirroring has stopped

   Use the following SQL function to confirm that transaction log mirroring has stopped.

```
 postgres=# SELECT pgx_is_wal_multiplexing_paused();
 pgx_is_wal_multiplexing_paused
-------------------
 t
(1 row)
```

   If transaction log mirroring has not stopped, then stop it using the following SQL function.

```
postgres=# SELECT pgx_pause_wal_multiplexing();
LOG:  multiplexing of transaction log files has been stopped
pgx_pause_wal_multiplexing
----------------------------

(1 row)
```

4. Stop output of archive logs

   Transaction logs may accumulate during replacement of backup storage disk, and if the data storage destination disk or the transaction log storage destination disk becomes full, there is a risk that operations may not be able to continue.

   To prevent this, use the following methods to stop output of archive logs.

   - Changing the archive_command parameter

     Specify a command that will surely complete normally, such as "echo skipped archiving WAL file %f" or "/bin/true", so that archive logs will be regarded as having been output.

     If you specify echo, a message is output to the server log, so it may be used as a reference when you conduct investigations.

   - Reloading the configuration file

     Run the pg_ctl reload command or the pg_reload_conf SQL function.

   If you simply want to stop output of errors without the risk that operations will not be able to continue, specify an empty string ('') in archive_command and reload the configuration file.

5. Replace with a larger capacity disk

   Replace the disk. Then, recover the volume configuration information.

6. Create a backup data storage destination

   Create a backup data storage destination.

   Example

```
# mkdir /backup/inst1
# chown fsepuser:fsepuser /backup/inst1
# chmod 700 /backup/inst1
```

   Refer to "3.2.2 Using Server Commands" for details.

7. Resume output of archive logs

   Return the archive_command setting to its original value, and reload the configuration file.

8. Resume transaction log mirroring

   Execute the pgx_resume_wal_multiplexing SQL function.

Example

```
SELECT pgx_resume_wal_multiplexing()
```

9. Run backup

   Use the pgx_dmpall command to back up the database cluster.

   Specify the following value in the pgx_dmpall command:

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

   #### Example

   ```
   > pgx_dmpall -D /database/inst1
   ```

10. Restore files

    Restore the files backed up in step 1.

11. Delete temporarily saved backup data

    If backup completes normally, the temporarily saved backup data becomes unnecessary and is deleted.

    The following example deletes backup data that was temporarily saved in /mnt/usb.

    #### Example

    ```
    > rm -rf /mnt/usb/backup
    ```

If an instance has been stopped

1. Back up files

   If the disk at the backup data storage destination contains any required files, back up the files. It is not necessary to back up the backup data storage destination.

2. Temporarily save backup data

   Save the backup data to a different directory.

   The reason for saving the backup data is so that the data in the data storage destination can be recovered even if it is corrupted before you perform the next step. If there is no disk at the save destination and you consider that there is no risk of corruption at the data storage destination, delete the backup data.

   The following example saves backup data from the backup data storage destination directory (/backup/inst1) under /mnt/usb/backup.

   #### Example

   ```
   > mkdir /mnt/usb/backup/
   > mv /backup/inst1/* /mnt/usb/backup/
   ```

3. Stop applications

   Stop applications that are using the database.

4. Stop the instance

   Stop the instance. Refer to "2.1.2 Using Server Commands" for information on how to stop an instance.

   If the instance fails to stop, refer to "17.11 Actions in Response to Failure to Stop an Instance".

5. Replace with a larger capacity disk

   Replace the disk. Then, recover the volume configuration information.

6. Create a backup data storage destination

   Create a backup data storage destination.

Example

```
# mkdir /backup/inst1
# chown fsepuser:fsepuser /backup/inst1
# chmod 700 /backup/inst1
```

Refer to "3.2.2 Using Server Commands" for details.

7. Start the instance

   Start the instance. Refer to "2.1.2 Using Server Commands" for information on how to start an instance.

8. Run backup

   Use the pgx_dmpall command to back up the database cluster.

   Specify the following value in the pgx_dmpall command:

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

   Example

   ```
   > pgx_dmpall -D /database/inst1
   ```

9. Resume applications

   Resume applications that are using the database.

10. Restore files

    Restore the files backed up in step 1.

11. Delete temporarily saved backup data

    If backup completes normally, the temporarily saved backup data becomes unnecessary and is deleted.

    The following example deletes backup data that was temporarily saved in /mnt/usb.

    Example

    ```
    > rm -rf /mnt/usb/backup
    ```

See

- Refer to "pgx_rcvall" and "pgx_dmpall" in the Reference for information on the pgx_rcvall command and pgx_dmpall command.
- Refer to "Write Ahead Log" under "Server Administration" in the PostgreSQL Documentation for information on archive_command.
- Refer to "B.1 WAL Mirroring Control Functions" for information on the pgx_is_wal_multiplexing_paused and pgx_resume_wal_multiplexing.

17.8 Actions in Response to Insufficient Space on the Transaction Log Storage Destination

If the transaction log storage destination runs out of space, check if the disk contains any unnecessary files and delete them so that operations can continue.

If deleting unnecessary files does not solve the problem, you must migrate data to a disk with larger capacity.

17.8.1 Replacing the Disk with a Larger Capacity Disk

Before replacing the disk with a larger capacity disk, migrate resources at the transaction log storage destination using the backup and recovery features.

There are two methods of performing backup and recovery:

- 17.8.1.1 Using WebAdmin
- 17.8.1.2 Using Server Commands

The following sections describe procedures that use each of these methods to replace the disk and migrate resources at the transaction log storage destination.

Point

It is recommended that you back up the database cluster following recovery. Backup deletes obsolete archive logs (transaction logs copied to the backup data storage destination), freeing up disk space and reducing the recovery time.

17.8.1.1 Using WebAdmin

Follow the procedure below to replace the disk and migrate resources at the transaction log storage destination by using WebAdmin.

1. Back up files

   If the disk at the transaction log storage destination contains any required files, back up the files. It is not necessary to back up the transaction log storage destination.

2. Back up the database cluster

   Back up the latest data storage destination resources and transaction log storage destination resources (refer to "3.2.1 Using WebAdmin" for details).

3. Stop applications

   Stop applications that are using the database.

4. Stop the instance

   Stop the instance. Refer to "2.1.1 Using WebAdmin" for information on how to stop an instance. WebAdmin automatically stops instances if recovery of the database cluster is performed without stopping the instance.

5. Replace with a larger capacity disk

   Replace the disk. Then, recover the volume configuration information.

6. Create a tablespace directory

   If a tablespace was defined after backing up, create a directory for it.

7. Recover the keystore, and enable automatic opening of the keystore

   Do the following if the data in the database has been encrypted:

   - Restore the keystore to its state at the time of the database backup.
   - Enable automatic opening of the keystore.

8. Recover the database cluster

   Log in to WebAdmin, and perform recovery operations. Refer to steps 4 ("Create a tablespace directory ") to 7 ("Run Recovery") under " If failure occurred in the data storage disk or the transaction log storage disk " in "17.1.1 Using WebAdmin" for information on the procedure. An instance is automatically started when recovery is successful.

9. Resume applications

   Resume applications that are using the database.

10. Restore files

    Restore the files backed up in step 1.

17.8.1.2 Using Server Commands

Follow the procedure below to replace the disk and migrate resources at the transaction log storage destination by using server commands.

1. Back up files

   If the disk at the transaction log storage destination contains any required files, back up the files. It is not necessary to back up the transaction log storage destination.

2. Back up the database cluster

   Use server commands to back up the latest data storage destination resources and transaction log storage destination resources. Refer to "3.2.2 Using Server Commands" for information on how to perform backup.

3. Stop applications

   Stop applications that are using the database.

4. Stop the instance

   After backup is complete, stop the instance. Refer to "2.1.2 Using Server Commands" for information on how to stop an instance.

   If the instance fails to stop, refer to "17.11 Actions in Response to Failure to Stop an Instance".

5. Replace with a larger capacity disk

   Replace the disk. Then, recover the volume configuration information.

6. Create a transaction log storage destination

   Create a transaction log storage destination. If a tablespace was defined, also create a directory for it.

   Example

   ```
   # mkdir /tranlog/inst1
   # chown fsepuser:fsepuser /tranlog/inst1
   # chmod 700 /tranlog/inst1
   ```

7. Recover the keystore, and enable automatic opening of the keystore

   When the data in the database has been encrypted, restore the keystore to its state at the time of the database backup. Configure automatic opening of the keystore as necessary.

8. Recover the database cluster

   Use the pgx_rcvall command to recover the database cluster.

   - Specify the data storage destination in the -D option. If the -D option is omitted, the value of the PGDATA environment variable is used by default.

   - Specify the backup storage directory in the -B option.

   Example

   ```
   > pgx_rcvall -D /database/inst1 -B /backup/inst1
   ```

   Note

   If recovery fails, remove the cause of the error in accordance with the displayed error message and then re-execute the pgx_rcvall command.

   If the message "pgx_rcvall: an error occurred during recovery" is displayed, then the log recorded when recovery was executed is output after this message. The cause of the error is output in around the last fifteen lines of the log, so remove the cause of the error in accordance with the message and then re-execute the pgx_rcvall command.

   The following message displayed during recovery is output as part of normal operation of pgx_rcvall command (therefore the user does not need not be concerned).

   ```
   FATAL: the database system is starting up
   ```

See

Refer to "pgx_rcvall" in the Reference for information on the pgx_rcvall command.

9. Start the instance

   Start the instance.

   Refer to "2.1.2 Using Server Commands" for information on how to start an instance.

10. Resume applications

    Resume applications that are using the database.

11. Restore files

    Restore the files backed up in step 1.

17.9 Errors in More Than One Storage Disk

If an error occurs in the storage destination disks or resources are corrupted, determine the cause of the error from system logs and server logs and remove the cause.

If errors occur in either of the following combinations, you cannot recover the database.

Recreate the instance, and rebuild the runtime environment.

Data storage disk	Transaction log storage disk	Backup data storage disk
Error	-	Error
-	Error	Error

See

Refer to "Setup" in the Installation and Setup Guide for Server for information on how to create an instance and build the runtime environment.

17.10 Actions in Response to Instance Startup Failure

If an instance fails to start, refer to the system log and the server log, and determine the cause of the failure.

If using WebAdmin, remove the cause of the error. Then, click [Solution] and [Recheck the status] and confirm that the instance is in the normal state.

The following sections describe common causes of errors and the actions to take.

17.10.1 Errors in the Configuration File

If you have directly edited the configuration file using a text editor or changed the settings using WebAdmin, refer to the system log and the server log, confirm that no messages relating to the files below have been output.

- postgresql.conf
- pg_hba.conf

See

Refer to the following for information on the parameters in the configuration file:

- "Configuring Parameters" in the Installation and Setup Guide for Server

- "Appendix A Parameters"
- "Server Configuration" and "Client Authentication" under "Server Administration" in the PostgreSQL Documentation

17.10.2 Errors Caused by Power Failure or Mounting Issues

If mounting is cancelled after restarting the server, for example, because the disk device for each storage destination disk was not turned on, or because automatic mounting has not been set, then starting an instance will fail.

Refer to "17.14.2 Errors Caused by Power Failure or Mounting Issues", and take actions accordingly.

17.10.3 Other Errors

This section describes the recovery procedure to be used if you cannot take any action or the instance cannot start even after you have referred to the system log and the server log.

There are two methods of recovery:

- 17.10.3.1 Using WebAdmin
- 17.10.3.2 Using Server Commands

Note that recovery will not be possible if there is an error at the backup data storage destination. If the problem cannot be resolved, contact Fujitsu technical support.

17.10.3.1 Using WebAdmin

Follow the procedure below to perform recovery.

1. Delete the data storage destination directory and the transaction log storage destination directory

   Back up the data storage destination directory and the transaction log storage destination directory before deleting them.

2. Reconfirm the status

   Log in to WebAdmin, and in the [Instances] tab, click [Solution] for the error message.

   Click [Recheck the status] to reconfirm the storage destination resources.

3. Run recovery

   Restore the database cluster after WebAdmin detects an error.

   Refer to "17.2.1 Using WebAdmin" for details.

17.10.3.2 Using Server Commands

Follow the procedure below to recover the database.

1. Delete the data storage destination directory and the transaction log storage destination directory

   Save the data storage destination directory and the transaction log storage destination directory, and then delete them.

2. Execute recovery

   Use the pgx_rcvall command to recover the database cluster.

   Refer to "17.2.2 Using the pgx_rcvall Command" for details.

17.11 Actions in Response to Failure to Stop an Instance

If an instance fails to stop, refer to the system log and the server log, and determine the cause of the failure.

If the instance cannot stop despite taking action, perform the following operation to stop the instance.

There are two methods of recovery:

- 17.11.1 Using WebAdmin

- 17.11.2 Using Server Commands

17.11.1 Using WebAdmin

In the [Instances] tab, click ■ and select the Fast stop mode or the Immediate stop mode to stop the instance. Forcibly terminate the server process from WebAdmin if the instance cannot be stopped.

Refer to "2.1.1 Using WebAdmin" for information on the stop modes.

17.11.2 Using Server Commands

There are three methods:

- Stopping the Instance Using the Fast Mode

  If backup is in progress, then terminate it, roll back all executing transactions, forcibly close client connections, and then stop the instance.

- Stopping the Instance Using the Immediate Mode

  Forcibly terminate the instance immediately. A crash recovery is run when the instance is restarted.

- Forcibly Stopping the Server Process

  Reliably stops the server process when the other methods are unsuccessful.

17.11.2.1 Stopping the Instance Using the Fast Mode

Specify "-m fast" in the pg_ctl command to stop the instance.

If the instance fails to stop when you use this method, stop the instance as described in "17.11.2.2 Stopping the Instance Using the Immediate Mode" or "17.11.2.3 Forcibly Stopping the Server Process".

Example

```
> pg_ctl stop -D /database/inst1 -m fast
```

17.11.2.2 Stopping the Instance Using the Immediate Mode

Specify "-m immediate " in the pg_ctl command to stop the instance.

If the instance fails to stop when you use this method, stop the instance as described in "17.11.2.3 Forcibly Stopping the Server Process".

Example

```
> pg_ctl stop -D /database/inst1 -m immediate
```

17.11.2.3 Forcibly Stopping the Server Process

If both the Fast mode and the Immediate mode fail to stop the instance, use the kill command or the kill parameter of the pg_ctl command to forcibly stop the server process.

The procedure is as follows:

1. Execute the ps command
   Note that "<*x*>" indicates the product version.

```
> ps axwfo user,pid,ppid,tty,command | grep postgres
fsepuser 19174 18027 pts/1                 \_ grep postgres
fsepuser 20517     1 ?        /opt/fsepv<x>server64/bin/postgres -D /database/inst1
fsepuser 20518 20517 ?         \_ postgres: logger
fsepuser 20520 20517 ?          \_ postgres: checkpointer
fsepuser 20521 20517 ?          \_ postgres: background writer
fsepuser 20522 20517 ?          \_ postgres: walwriter
fsepuser 20523 20517 ?          \_ postgres: autovacuum launcher
fsepuser 20524 20517 ?          \_ postgres: archiver
fsepuser 20525 20517 ?          \_ postgres: logical replication launcher
```

The process ID (20517) indicates the server process.

2. Forcibly stop the server process

   As instance manager, forcibly stop the server process.

   Using the pg_ctl command

   ```
   > pg_ctl kill SIGQUIT 20517
   ```

   Using the kill command

   ```
   > kill -s SIGQUIT 20517
   ```

17.12 Actions in Response to Failure to Create a Streaming Replication Standby Instance

When creating a streaming replication standby instance using WebAdmin, if the instance creation fails, refer to the system log and the server log, and determine the cause of the failure.

When an error occurs in the creation of the standby instance using WebAdmin, it is unlikely that the partially created standby instance can be resumed to complete the operation.

In such a scenario, fix the cause of the error, delete the partially created standby instance, and then create a new standby instance. This recommendation is based on the following assumptions:

- As the instance is yet to be created completely, there are no applications connecting to the database.
- The standby instance is in error state and is not running.
- There are no backups for the standby instance and as a result, it cannot be recovered.

See

Refer to "Deleting Instances" in the Installation and Setup Guide for details on how to delete an instance.

17.13 Actions in Response to Error in a Distributed Transaction

If a system failure (such as server failure) occurs in an application that uses distributed transactions, then transactions may be changed to the in-doubt state.

At that point, resources accessed by the transaction will be locked, and rendered unusable by other transactions.

The following describes how to check for in-doubt transactions, and how to resolve them.

How to check for in-doubt transactions

The following shows how to check for them:

If the server fails

1. An in-doubt transaction will have occurred if a message similar to the one below is output to the log when the server is restarted.

   Example

   ```
   LOG: Restoring prepared transaction 2103.
   ```

2. Refer to system view pg_prepared_xacts to obtain information about the prepared transaction.

   If the transaction identifier of the prepared transaction in the list (in the transaction column of pg_prepared_xacts) is the same as the identifier of the in-doubt transaction obtained from the log output when the server was restarted, then that row is the information about the in-doubt transaction.

   Example

   ```
   postgres=# select * from pg_prepared_xacts;
    transaction |   gid     |   prepared |  owner   | database
   -------------+-----------+------------+----------+----------
   2103 | 374cc221-f6dc-4b73-9d62-d4fec9b430cd | 2022-03-06 16:28:48.471+08 | postgres |
   postgres (1 row)
   ```

   Information about the in-doubt transaction is output to the row with the transaction ID 2103 in the transaction column.

If the client fails

If there are no clients connected and there is a prepared transaction in pg_prepared_xacts, then you can determine that the transaction is in the in-doubt state.

If at least one client is connected and there is a prepared transaction in pg_prepared_xacts, you cannot determine whether there is a transaction in the in-doubt state. In this case, use the following query to determine the in-doubt transaction from the acquired database name, user name, the time PREPARE TRANSACTION was executed, and the information about the table name accessed.

```
select gid,x.database,owner,prepared,l.relation::regclass as relation from pg_prepared_xacts x
left join pg_locks l on l.virtualtransaction = '-1/'||x.transaction and l.locktype='relation';
```

If it still cannot be determined from this information, wait a few moments and then check pg_prepared_xacts again.

If there is a transaction that has continued since the last time you checked, then it is likely that it is the one in the in-doubt state.

Point

As you can see from the explanations in this section, there is no one way to definitively determine in-doubt transactions.

Consider collecting other supplementary information (for example, logging on the client) or performing other operations (for example, allocating database users per job).

How to resolve in-doubt transactions

From the system view pg_prepared_xacts mentioned above, obtain the global transaction identifier (in the gid column of pg_prepared_xacts) for the in-doubt transaction, and issue either a ROLLBACK PREPARED statement or COMMIT PREPARED statement to resolve the in-doubt transaction.

Example

- Rolling back in-doubt transactions

```
postgres=# rollback prepared '374cc221-f6dc-4b73-9d62-d4fec9b430cd';
ROLLBACK PREPARED
```

- Committing in-doubt transactions

```
postgres=# commit prepared '374cc221-f6dc-4b73-9d62-d4fec9b430cd';
COMMIT PREPARED
```

17.14 I/O Errors Other than Disk Failure

Even if a disk is not defective, the same input-output error messages, as those generated when the disk is defective, may be output.

A few examples of such errors are given below. The appropriate action for each error is explained respectively.

- 17.14.1 Network Error with an External Disk
- 17.14.2 Errors Caused by Power Failure or Mounting Issues

17.14.1 Network Error with an External Disk

This is an error that occurs in the network path to/from an external disk.

Determine the cause of the error by checking the information in the system log and the server log, the disk access LED, network wiring, and network card status. Take appropriate action to remove the cause of the error, for example, replace problematic devices.

17.14.2 Errors Caused by Power Failure or Mounting Issues

These are errors that occur when the disk device is not turned on, automatic mounting of the disk was not set, or mounting was accidentally cancelled.

In this case, check the information in the system log and the server log, the disk access LED, and whether the disk is mounted correctly. If problems are detected, take appropriate action.

If mounting has been cancelled, it is possible that mounting was accidentally cancelled, or automatic mounting at the time of starting the operating system is not set. In this case, set the mounting to be performed automatically.

17.15 Anomaly Detection and Resolution

The following operations performed via the command line interface will result in an anomaly in WebAdmin:

- Changes to the port and backup_destination parameters in postgresql.conf
- Changes to Mirroring Controller configuration of cluster replication added via WebAdmin

This section describes when WebAdmin checks for such anomalies, and what takes place when an anomaly is detected.

17.15.1 Port Number and Backup Storage Path Anomalies

An anomaly occurs when the value of [Port number] and/or [Backup storage path] in WebAdmin is different from the value of its corresponding parameter in postgresql.conf - port and backup_destination, respectively.

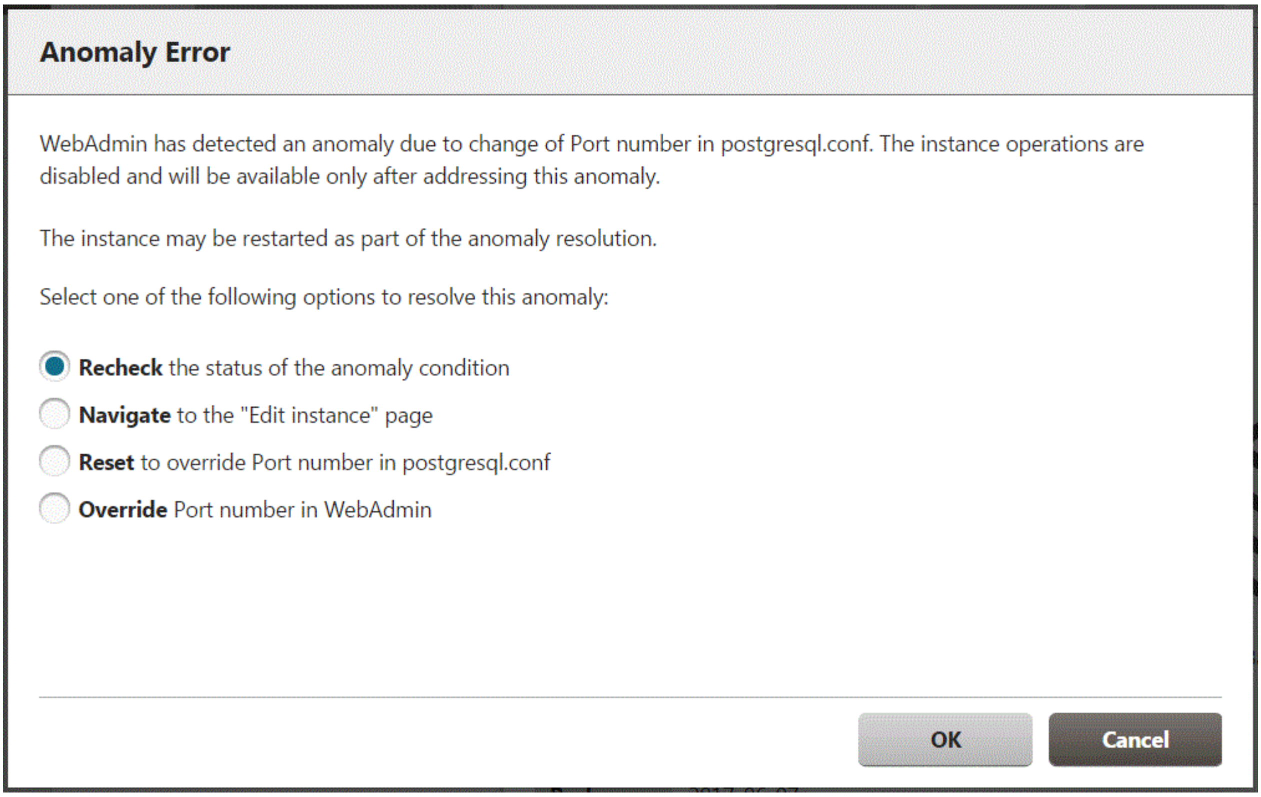
WebAdmin checks for anomalies when an instance is selected for viewing or any instance operation is performed. Anomalies will be identified for the selected instance only.

The following occurs when an anomaly is detected in port number and/or backup storage path:

- All instance operation buttons are disabled, except for "Edit instance", "Refresh instance", and "Delete Mirroring Controller"
- A red error status indicator is displayed on the instance icon
- For an anomaly specific to backup storage path, a red error status indicator is displayed on the [Backup storage] disk icon, and [Backup storage status] is set to "Error"
- The message, "WebAdmin has detected an anomaly with...", is displayed in the [Message] section along with an associated [Solution] button

Critical errors encountered during anomaly resolution will be displayed, however, rollback of the instance to its previous state is not supported.

Click [Solution]. The [Anomaly Error] dialog box is displayed.



Anomaly Error

WebAdmin has detected an anomaly due to change of Port number in postgresql.conf. The instance operations are disabled and will be available only after addressing this anomaly.

The instance may be restarted as part of the anomaly resolution.

Select one of the following options to resolve this anomaly:

● **Recheck** the status of the anomaly condition

○ **Navigate** to the "Edit instance" page

○ **Reset** to override Port number in postgresql.conf

○ **Override** Port number in WebAdmin

Select the required option, click [OK], and then resolve the anomaly error.

Refer to "Editing instance information" in the Installation and Setup Guide for Server for information on the [Edit instance] page.

17.15.2 Mirroring Controller Anomalies

The following conditions will cause a Mirroring Controller anomaly:

- The Mirroring Controller management folder or configuration files have been deleted
- The permissions to the Mirroring Controller management folder or configuration files have been changed such that:
  - The instance administrator's access to Mirroring Controller configuration is denied
  - Users other than an instance administrator have access privileges to Mirroring Controller configuration files

WebAdmin checks for anomalies when Mirroring Controller status check is performed.

The following occurs when a Mirroring Controller anomaly is detected:

- All Mirroring Controller functionality is disabled for the replication cluster, except for "Delete Mirroring Controller"
- [Mirroring Controller status] is set to "Error"
- Either of the following messages is displayed in the [Message] section

  "Failed to access the Mirroring Controller management folder or configuration files '*path*'. Mirroring Controller functionality has been disabled. Consider deleting Mirroring Controller and adding it again."

  "Failed to find the Mirroring Controller management folder or configuration files '*path*'. Mirroring Controller functionality has been disabled. Consider deleting Mirroring Controller and adding it again."

Appendix A Parameters

This appendix describes the parameters to be set in the postgresql.conf file of Fujitsu Enterprise Postgres.

The postgresql.conf file is located in the data storage destination.

Information

The maximum value that can be expressed as a 4-byte signed integer changes according to the operating system. Follow the definition of the operating system in use.

- core_directory (string)

  This parameter specifies the directory where the corefile is to be output. If this parameter is omitted, the data storage destination is used by default. This parameter can only be set when specified on starting an instance. It cannot be changed dynamically, while an instance is active.

- core_contents (string)

  This parameter specifies the contents to be included in the corefile.

  - full: Outputs all contents of the server process memory to the corefile.

  - none: Does not output a corefile.

  - minimum: Outputs only non-shared memory server processes to the corefile. This reduces the size of the corefile. However, in some cases, this file may not contain sufficient information for examining the factor that caused the corefile to be output.

  If this parameter is omitted, "minimum" is used by default. This parameter can only be set when specified on starting an instance. It cannot be changed dynamically, while an instance is active.

- keystore_location (string)

  This parameter specifies the directory that stores the keystore file. Specify a different location from other database clusters. This parameter can only be set when specified on starting an instance. It cannot be changed dynamically, while an instance is active.

  Cannot be specified with the tde_kms.kms_conninfo_file parameter.

- tde_kms.plugin_path (string)

  When connecting to the key management system using a plug-in, specify the directory that stores the plug-in with an absolute path. Only database administrators should be able to store plugins in this directory. It can only be set by specifying a parameter when starting the instance.

- tde_kms.enable_shared_dek (boolean)

  Enables or disables sharing of data encryption keys between backend processes for each encrypted tablespace in transparent data encryption. Default is off. It can only be set by specifying a parameter when starting the instance.

- tde_kms.max_shared_dek (numerical value)

  Specify the maximum number of shared data encryption keys when sharing data encryption keys per tablespace in transparent data encryption. Default is 1000. It can only be set by specifying a parameter when starting the instance.

- tde_kms.kms_conninfo_file (string)

  When using the key management service as a key store, specifies the file that contains the connection information for the key management system. Cannot be specified with the keystore_location parameter.

  Create a connection information file for the key management system in one of the following format:

```
kmip    kms-name    address    port    auth-method    [auth-options]
custom  kms-name    plugin-name      [plugin-options]   [extra-args]
```

  Specify one of the following methods to connect to the key management system.

- kmip

  Access this key management system using the KMIP protocol. Take out and use the encryption key in this key management system.

- custom

  Access the key management system using a plugin module. The encryption/decryption process is done inside the plugin or inside the key management system. Fujitsu Enterprise Postgres does not use cryptographic keys directly.

For type kmip

- kms-name

  The key management system name assigned to the key management system and specified when declaring the master encryption key or opening the keystore. The name of the key management system must be unique within this file. The key management system name must be a string of no more than 63 characters beginning with a-z, consisting of a-z, a number (0 -9), and an underscore. Upper and lower case letters are the same.

- address

  Specifies the host name or IP address of the key management service.

- port

  Specifies the port number on which the key management service listens for services.

- auth-method

  Specifies the authentication method for the key management service.

- auth-options

  The auth-method is followed by the authentication method options. You can specify multiple options in a name = value field.

Authentication method when using a key management service of type kmip

cert

A certificate is used to authenticate the KMIP server and the client, Fujitsu Enterprise Postgres, to each other. The auth-options can be.

- sslcert

  Specifies the file name of the client certificate. The corresponding format is PEM format.

- sslkey

  Specifies the file name of the private key used for the client certificate. The corresponding format is PEM format. If you choose to encrypt the file with a passphrase, use a passphrase that is no more than 1023 bytes long.

- sslkeypassphrase-obf

  Specifies the file that contains the obfuscated passphrase for the private key file specified by sslkey. This option allows the keystore to be opened automatically when the server starts. The pgx _ keystore command creates obfuscated files. It can be omitted.

- sslrootcert

  Specifies the file name of the SSL Certificate Authority certificate. The corresponding format is PEM format. Used to verify the server certificate of the connection destination.

Example

```
kmip   mykmipsvr  mykmipsvr.example.com   5696   cert   sslcert=postgres.crt
sslkey=postgres.key sslrootcert=root.crt
```

Note

cert authentication does not verify that the server you are connecting to is the same server you are trying to connect to. Any server using a server certificate that is signed with the certificate of the certificate authority specified in sslrootcert is considered the correct destination. To avoid problems with this behavior, consider using your own CA or self-signed certificate for the KMIP server.

For type custom

- kms-name

  The key management system name given to the key management system, specified when declaring a master encryption key or opening a keystore. The name of the key management system must be unique within this file. The key management system name must be a string of up to 63 characters starting with a-z and consisting of a-z, digits (0-9) and underscores. Uppercase and lowercase letters are considered the same.

- plugin-name

  Specify the name of the plugin. Use the executable file with the same name as the plugin name in the directory specified by tde_kms.plugin_path as the plugin module.

- plugin-options

  Specify other options for the plugin. Multiple options can be specified in name=value format fields. You can specify the following option names.

  - kms-secret-obf

    Specify the file containing the obfuscated KMS secret when enabling automatic opening of the keystore using transparent data encryption. The obfuscated file is created with the pgx_keystore command. The obfuscated file contents are decrypted by Fujitsu Enterprise Postgres and passed to the plugin. Can be omitted if you are not using transparent data encryption to enable automatic opening of the keystore.

- extra-args

  Specify additional arguments to pass to the specified shell command in the form arg=value. If you specify multiple extra-args, the values are passed to the shell command in that order.

Example

```
custom   mykms  mykms  arg=--profile arg=user1
```

- tablespace_encryption_algorithm (string)

  This parameter specifies the encryption algorithm for tablespaces that will be created. Valid values are "AES128", "AES256", and "none". If you specify "none", encryption is not performed. The default value is "none". To perform encryption, it is recommended that you specify "AES256". Only superusers can change this setting.

- backup_destination (string)

  This parameter specifies the absolute path of the directory where pgx_dmpall will store the backup data. Specify a different location from other database clusters. This parameter can only be set when specified on starting an instance. It cannot be changed dynamically, while an instance is active.

  Place this directory on a different disk from the data directory to be backed up and the tablespace directory. Ensure that users do not store arbitrary files in this directory, because the contents of this directory are managed by the database system.

- search_path (string)

  When using the SUBSTR function compatible with Oracle databases, set "oracle" and "pg_catalog" in the search_path parameter. You must specify "oracle" before "pg_catalog".

Example

```
search_path = '"$user", public, oracle, pg_catalog'
```

Information

- The search_path feature specifies the priority of the schema search path. The SUBSTR function in Oracle database is defined in the oracle schema.

- Refer to "Statement Behavior" under "Server Administration" in the PostgreSQL Documentation for information on search_path.

- track_waits (string)

  This parameter enables collection of statistics for pgx_stat_lwlock and pgx_stat_latch.

  - on: Enables collection of statistics.

  - off: Disables collection of statistics.

  If this parameter is omitted, "on" is assumed.

  Only superusers can change this setting.

- track_sql (string)

  This parameter enables collection of statistics for pgx_stat_sql.

  - on: Enables collection of statistics.

  - off: Disables collection of statistics.

  If this parameter is omitted, "on" is assumed.

  Only superusers can change this setting.

Parameters for the in-memory feature

- reserve_buffer_ratio (numerical value)

  This parameter specifies the proportion of shared memory to be used for a stable buffer table.

  - Minimum value: 0

  - Maximum value: 80

  If this parameter is omitted, 0 will be used.

- vci.cost_threshold (numerical value)

  This parameter specifies the lowest cost that selects an execution plan that uses a VCI. If the cost of the best execution plan that does not use a VCI is lower than this value, that execution plan will be selected.

  - Minimum value: 0

  - Maximum value: Maximum value that can be expressed as a 4-byte signed integer

  If this parameter is omitted or a value outside this range is specified, 18000 will be used.

- vci.control_max_workers (numerical value)

  This parameter specifies the number of background workers that manage VCI. The number of workers for the entire instance is limited by max_worker_processes, so add the value specified here to max_worker_processes.

  - Minimum value: 1

  - Maximum value: 8388607

  If this parameter is omitted or a value outside this range is specified, 8 will be used.

- vci.enable (string)

  This parameter enables or disables VCI.

  - on: Enables VCI.

  - off: Disables VCI.

  If this parameter is omitted, "on" will be used.

- vci.log_query (string)

  This parameter enables or disables log output when VCI is not used due to insufficient memory specified by vci.max_local_ros.

  - on: Enables log output.

  - off: Disables log output.

  If this parameter is omitted, "off" will be used.

- vci.maintenance_work_mem (numerical value)

  This parameter specifies the maximum memory size used for maintenance of VCI (when executing CREATE INDEX, for example).

  - Minimum value: 1 MB

  - Maximum value: Maximum value that can be expressed as a 4-byte signed integer

  If this parameter is omitted or a value outside this range is specified, 256 MB will be used.

- vci.max_local_ros (numerical value)

  This parameter specifies the maximum memory size used for VCI scan.

  - Minimum value: 64 MB

  - Maximum value: Maximum value that can be expressed as a 4-byte signed integer

  If this parameter is omitted or a value outside this range is specified, 64 MB will be used.

- vci.max_parallel_degree (numerical value)

  This parameter specifies the maximum number of background workers used for parallel scan. The number of workers for the entire instance is limited by max_worker_processes, so add the value specified here to max_worker_processes.

  A value from -8388607 to 8388607 can be specified.

  - Integer (1 or greater): Parallel scan is performed using the specified degree of parallelism.

  - 0: Stops the parallel scan process.

  - Negative number: The specified value minus the maximum number of CPUs obtained from the environment is used as the degree of parallelism and parallel scan is performed.

  If this parameter is omitted or a value outside this range is specified, 0 will be used.

- vci.shared_work_mem (numerical value)

  This parameter specifies the maximum memory size used for VCI parallel scan.

  - Minimum value: 32 MB

  - Maximum value: Maximum value that can be expressed as a 4-byte signed integer

  If this parameter is omitted or a value outside this range is specified, 1 GB will be used.

Parameters for the Global Meta Cache feature

- pgx_global_metacache (numerical value)

  Specifies the memory size of the GMC area.

  Specify a value calculated by the formula below.

A value lower than the calculated value will still work, but the meta cache may not be able to fit into the GMC area.

In this case, the system will discard the meta cache it thinks it is no longer needed, but if it is needed again, the meta cache will need to be expanded and will not perform well.

If the value is less than 10 MB and is set to a nonzero value that disables the feature, the database startup fails because the Global Meta Cache feature cannot operate.

A setting of 0 disables the Global Meta Cache feature. The default is 0.

Changing this setting requires restarting the database.

```
Size of GMC area
 = Max(10MB,
     (All user table x 0.4 KB
      + All user Indexes x 0.3 KB
      + All user columns x 0.8 KB) x 1.5 (*1) )

*1) Safety Factor (1.5)
This value takes into account the case where both GMC before and after the change temporarily exist
at the same time in shared memory when the table definition is changed or the row of the system
catalog is changed.
```

- track_gmc (string)

  This parameter enables collection of statistics for pgx_stat_gmc.

  - on: Enables collection of statistics.

  - off: Disables collection of statistics.

  If this parameter is omitted, "on" is used.

  Only superusers can change this setting.

Parameters for the Local Meta Cache Limit feature

- pgx_catalog_cache_max_size(numerical value)

  Specifies the maximum amount of memory that the backend process should use as the catalog cache.

  You can enable catalog cache deletion by setting it to 8 KB or more.

  A setting of 0 disables the catalog cache removal. The default is 0.

  If no units are specified, they are treated as KB.

  - Minimum value: 8KB

  - Maximum value: Maximum value that can be expressed as a 4-byte signed integer

  When calculating the parameter settings, the factors that determine the cache size are calculated as the number of tables, the number of indexes, and the number of columns. What is kept as a catalog cache or relation cache also includes objects such as databases, roles, or procedures, but these are small compared to the above factors and do not need to be factored into them. It also includes a calculation method for pgx_relacion_cache_max_size because the given memory is distributed between the catalog cache and the relation cache.

  Note

  The calculation method here assumes that all backends have similar access and that the transaction also has access to a similar number of resources. If you have a small number of singular backends or transactions, consider excluding them as errors.

  1. Determine how much memory a backend process can use. Decide by subtracting the memory size required by the entire system such as the database cache from the installed memory and dividing the rest by the number of connections.

  2. For best performance, use the following formula to calculate the total memory size of the catalog cache when the backend holds the catalog cache for all resources accessed during its lifetime.

The amount of memory varies depending on whether Global Meta Cache is enabled or disabled. Enabling Global Meta Cache reduces the amount of memory required because most of the cache is located on shared memory.

```
When Global Meta Cache is enabled:
  (Number of tables to access + Number of indexes to access + Number of columns to access)
      × 0.1KB × 1.5 (*1)
When Global Meta Cache is disabled:
  { Number of tables to access × 0.5KB(pg_class tuple size)
      + Number of indexes to access × 0.5KB(pg_index tuple size)
      + Number of columns to access × 1.0KB(pg_statistic tuple size)} × 1.5 (*1)

*1) Safety Factor (1.5)
The system catalog contains columns with variable-length types. For example, the tuple size in
pg_class is a constant value multiplied by the number of tables, while relname in pg_class is
variable length data.
It is not practical to calculate every definition in detail, so we added 50% to the above
formula.
```

3. In the same way as in 2., calculate the relation cache using the following formula.

```
  (1.4KB × Number of tables to access + 2.4KB × Number of indexes to access) × 1.5 (*1)

*1) Safety Factor (1.5)
The relation cache is structured to facilitate the use of table and index definitions, holds
pointers to various objects, and is sized to include them. It is variable length because the
type of object allocated by the table definition and its size change. Since it is not
realistic to calculate for all definitions, 50% is added.
```

4. If the value of 1. ≥ the value of 2. + the value of 3., the backend process can keep all caches to the extent allowed, so there is no need to limit the caches. If you want to cap for safety, set the value of 2. to pgx_catalog_cache_max_size and the value of 3. to pgx_relation_cache_max_size.

5. If the value of 1. < the value of 2. + the value of 3. then you need to limit the cache. However, this parameter does not limit the size of the cache used by a transaction. Therefore, take the following steps.

6. Calculate the catalog cache used by a transaction using the formula in 2.

7. Calculate the relation cache used by a transaction using the formula in 3.

8. If the value of 1. < the value of 6. + the value of 7., then the value of 1. needs to be increased. In other words, in some cases, it may be necessary to increase the installed memory or reduce the number of connections.

9. If the value of 1. ≥ the value of 6. + the value of 7., the condition of 1. can be satisfied by limiting the cache with this parameter. Divide the value of 1. by the ratio of 2. and 3. and set it as a parameter. Set the value distributed to 2. to pgx_catalog_cache_max_size and the value distributed to 3. to pgx_relation_cache_max_size.

10. The value calculated in 9. is a provisional value. If you cannot meet your target performance, first try to shift the focus of allocation to the relation cache. This is because when executing SQL, the relation cache generated based on the catalog cache is mainly referenced, so it is advantageous to leave a large amount of relation cache. If the performance is still not satisfied, adjust the parameters by referring to "15.1.4 Performance Impact and Parameter Tuning of the Local Meta Cache Limit Feature".

Note

Be careful when partitioning the table.

The cached definition changes depending on whether the parent table is specified in the SQL statement or the child table is specified. In particular, note that if you specify a parent table, the definitions of all child tables are cached. This is because when you specify a parent table in an SQL statement, you need to know the definitions of all the child tables in order to determine which child table will contain the desired data. Note that the column information of the parent table is not cached.

When specifying the parent table:

```
Number of tables to access = Number of parent tables to access + Number of defined child tables
Number of columns = Number of defined columns x number of defined child tables
```

When specifying the child table directly:

```
Number of tables to access = Number of child tables actually accessed
Number of columns ＝ Number of defined columns x number of child tables actually accessed
```

Example)

Suppose the parent table T (1 index, 3 columns) is split from child tables T1 to T5 (1 index, 3 columns, respectively). If the parent table T is specified in SQL, when the child tables that contain the data to be queried are limited to T1 and T2, and when accessing the data using the indexes defined by T1 and T2, calculate as follows.

```
Number of tables = 1(parent table) + 5(child table) = 6
Number of indexes = 2 (index to access)
Number of columns = 3 (number of columns) x 5 (child table) = 15
```

If you specify child tables T1 and T2 in SQL and use the indexes defined on T1 and T2 when accessing data, the calculation is as follows.

```
Number of tables = 2(child table)
Number of indexes = 2 (index to access)
Number of columns = 3 (number of columns) x 2 (child table) = 6
```

- pgx_relation_cache_max_size(numerical value)

  Specifies the maximum amount of memory that the backend process should use as the relation cache.

  You can enable catalog cache deletion by setting it to 8 KB or more.

  A setting of 0 disables the relation cache removal. The default is 0.

  If no units are specified, they are treated as KB.

  - Minimum value: 8KB
  - Maximum value: Maximum value that can be expressed as a 4-byte signed integer

  For the calculation method for parameter setting, refer to the calculation method of pgx_catalog_cache_max_size.

- pgx_cache_hit_log_interval(numerical value)

  Specifies the time interval to output a message indicating the cache reference status for each backend process.

  When the transaction ends, if the time set in this parameter has elapsed since the previous message was output, the message is output.

  If set to 0, a message will be output each time the transaction ends.

  Setting -1 disables the output. The default value is 10min.

  If no units are specified, they are treated as ms.

  Even if pgx_catalog_cache_max_size and pgx_relation_cache_max_size are disabled, the message output of the corresponding cache will be invalid.

  Immediately after connecting to the server, a small transaction occurs before the request from the user application, such as for user authentication. Since it is meaningless to know the hit rate for these, a message will be output at the end of the transaction that started after the time set in this parameter has elapsed after connecting to the server.

  For the same reason, setting a small value such as 0 may result in a message being printed at the end of such a small transaction.

  You can check which transaction the message corresponds to from the information output at the beginning.

  This information depends on the setting of the parameter log_line_prefix.

  - Minimum value: 0
  - Maximum value: 2147483647ms

See

Refer to "Server Configuration" under "Server Administration" in the PostgreSQL Documentation for information on other postgresql.conf parameters.

Parameters for the Policy-based Password Management

- userprofile_database(string)

  Specifies the name of the database to which the background worker process connects. This parameter must specify a connectable database name. You cannot use the pseudo database "replication". If you omit this parameter, then it is assumed to be "postgres".

  If you change this parameter, reload the configuration file.

Parameters for using OpenSSL legacy algorithms

- openssl_conf(string)

  Specifies the OpenSSL configuration file. Specifying a valid configuration file makes legacy algorithms available. Use the example below to prepare the configuration file in any directory.

  If this parameter is not specified, the empty string is assumed.

  This parameter can only be set by specifying the parameter at instance startup.

  You cannot make dynamic changes during instance startup.

  Example

  openssl_conf = '/path/to/openssl.conf'

  [OpenSSL Configuration File (openssl.conf) Example]

  ```
  =====================
  openssl_conf = openssl_init

  [openssl_init]
  providers = provider_sect

  [provider_sect]
  default = default_sect
  legacy = legacy_sect

  [default_sect]
  activate = 1

  [legacy_sect]
  activate = 1
  =====================
  ```

- openssl_modules(string)

  Specifies the directory that contains additional OpenSSL modules.

  Legacy algorithms are available by specifying 'server installation directory/lib/ossl-modules'.

  If this parameter is not specified, the empty string is assumed.

  This parameter can only be set by specifying the parameter at instance startup.

  You cannot make dynamic changes during instance startup.

  This parameter sets the OPENSSL_MODULES environment variable to be applied to the server process. Do not set the OPENSSL_MODULES environment variable in any way other than setting this parameter, as this may cause abnormal behavior.

Example

openssl_modules = '/opt/fsepv<*x*>server64/lib/ossl-modules'

Note that "<*x*>" indicates the product version.

Information

Legacy algorithms include the following encryption algorithms:

- BF
- CAST5
- DES-ECB
- DES-CBC
- MD4
- Whirlpool

Appendix B System Administration Functions

This appendix describes the system administration functions of Fujitsu Enterprise Postgres.

See

Refer to "System Administration Functions" under "The SQL Language" in the PostgreSQL Documentation for information on other system administration functions.

B.1 WAL Mirroring Control Functions

The following table lists the functions that can be used for backup and recovery based on WAL mirroring.

Table B.1 WAL mirroring control functions

Name	Return type	Description
pgx_pause_wal_multiplexing()	void	Stops WAL multiplexing
pgx_resume_wal_multiplexing()	void	Resumes WAL multiplexing
pgx_is_wal_multiplexing_paused()	boolean	Returns true if WAL multiplexing has stopped

If WAL multiplexing has not been configured, these functions return an error. Setting the backup_destination parameter in postgresql.conf configures WAL multiplexing.

Only superusers can execute these functions.

B.2 Transparent Data Encryption Control Functions

The following table lists the functions that can be used for transparent data encryption.

Table B.2 Transparent data encryption control functions

Name	Return type	Description

pgx_open_keystore(*passphrase*)
pgx_open_keystore(sslpassphrase => text)
	void	Opens the keystore

pgx_set_master_key(*passphrase*)	void	Sets the master encryption key
pgx_declare_external_master_key(kms_name => text, key_id => text, sslpassphrase => text)	void	To set an encryption key existing in a key management system as a master encryption key for transparent data encryption.
pgx_set_keystore_passphrase(*oldPassphrase*, *newPassphrase*)	void	Changes the keystore passphrase

B.2.1 pgx_open_keystore

pgx_open_keystore opens the keystore.

Only superusers can execute this function. Also, this function cannot be executed within a transaction block.

File-based keystores:

The pgx_open_keystore function uses the specified passphrase to open the keystore. When the keystore is opened, the master encryption key is loaded into the database server memory. In this way, you can access the encrypted data and create encrypted tablespaces. If the keystore is already open, this function returns an error.

Using the key management system as a keystore

pgx_open_keystore makes available (opens a keystore) a master encryption key on a key management system that has already been declared for use. The keystore cannot be opened unless it has been declared to use a master encryption key.

If the keystore is already open, use the credentials you entered to reconnect to the key management system.

Specify the authentication information for connecting to the key management system. Arguments must be specified in naming notation. The information you pass in the argument depends on the key management system you use.

If the key management system information file specifies an obfuscated credentials file, the file is recreated with the new credentials.

Using the key management service of type kmip

The following arguments are specified in naming notation.

- sslpassphrase text

  Specifies the passphrase of the client certificate private key file when connecting to the KMIP server. This can be omitted if no passphrase is set in the private key file.

Using the key management service of type custom

The following arguments are specified in naming notation.

- kms_secret text

  Confidential information passed to the plugin. It can be omitted if it is not necessary for using the key management system. Whether or not it can be omitted depends on the implementation of the plugin.

Example

To specify the passphrase mykmippassphrase for the client certificate private key file in naming notation:

```
SELECT pgx_open_keystore( sslpassphrase => 'mykmippassphrase' );
```

B.2.2 pgx_set_master_key

The pgx_set_master_key function generates a master encryption key and stores it in the file-based keystore.

If the keystore does not exist, this function creates a keystore. If the keystore already exists, this function modifies the master encryption key. If the keystore has not been opened, this function opens it.

The passphrase is a string of 8 to 200 bytes.

Only superusers can execute this function. Also, this function cannot be executed within a transaction block. Processing is not affected by whether the keystore is open.

B.2.3 pgx_declare_external_master_key

pgx_declare_external_master_key declares the use of an encryption key that exists in the key management system as the master encryption key for transparent data encryption. If the master encryption key already exists, change the master encryption key. If the master encryption key already exists, the keystore must be open.

The argument specifies information that identifies the master encryption key. Arguments must be specified in naming notation. The information you pass in the argument depends on the key management system you use.

This function can only be executed by superuser. Also, you cannot execute this function within a transaction block.

This function is available if you have installed the extension 'tde_kms'.

The following arguments are specified in naming notation:

- kms_name text

  Specify the key management system name specified in the key management system connection information file. Required.

- key_id text

  Specify the key ID assigned to the encryption key. Cannot be omitted.

- sslpassphrase text

  Specify the passphrase of the client certificate private key file when connecting to the KMIP server. This can be omitted if the private key file does not have a passphrase. Ignored if the key management system type specified by kms_name is not kmip.

- kms_secret text

  Confidential information passed to the plugin. It can be omitted if it is not necessary for using the key management system. Whether or not it can be omitted depends on the implementation of the plugin. Ignored if the key management system type specified by kms_name is not custom.

Example

```
SELECT pgx_declare_external_master_key( kms_name => 'mykmipsvr', key_id =>
'a0eebc99-9c0b-0000-0000-000000000000', sslpassphrase => 'mykmippassphrase' );
```

B.2.4 pgx_set_keystore_passphrase

The pgx_set_keystore_passphrase function changes the file-based keystore passphrase.

Specify the current passphrase in *oldPassphrase*, and a new passphrase in *newPassphrase*.

The passphrase is a string of 8 to 200 bytes.

Only superusers can execute this function. Also, this function cannot be executed within a transaction block. Processing is not affected by whether the keystore is open.

B.3 Profile Management Functions and User Management Functions

The table below lists the profile management functions and user management functions used in policy-based password management.

B.3.1 Profile Management Functions

Name	Return type	Description

pgx_create_profile(profile_name name, password_parameter json)	void	Create a new profile.
		For profile_name, specify the profile name. An error will occur if an existing profile name is specified.
		For password_parameter, specify parameters and values in key-value format as follows.
		`'{`
		`    "PASSWORD_LIFE_TIME": 30,`
		`    "PASSWORD_GRACE_TIME": "UNLIMITED",`
		`    "PASSWORD_REUSE_TIME": 10,`
		`    "PASSWORD_REUSE_MAX": 5,`
		`    "PASSWORD_LOCK_TIME": 0.5,`
		`    "FAILED_LOGIN_ATTEMPTS": "DEFAULT",`
		`    "PASSWORD_ALLOW_HASHED": true`
		`}');`
		The json value accepts integer, numeric, boolean depending on the parameter, but only accepts strings "DEFAULT" and "UNLIMITED" as special values.

Name	Return type	Description
		For parameters that are omitted in json or that specify null for value, they follow the values in the default profile.Also, password_parameter can be omitted, and if omitted, a profile with all parameters conforming to the default profile will be created.

pgx_alter_profile(profile_name name, alter_parameter json)	void	Update the contents of an existing profile.
		For profile_name, specify the name of the profile to update. If you specify a profile name that does not exist, an error will occur.
		In alter_parameter, specify the variable name and value you want to change in key-value format as follows. Does not change the profile contents for parameters that are omitted or for which value is null.
		`'{`
		`    "name": "new_name",`
		`    "PASSWORD_LIFE_TIME": 50,`
		`    "PASSWORD_GRACE_TIME": 10`
		`}');`
		name: Specify the modified profile name. null cannot be specified.
		If the default profile is specified in profile_name, the following changes cannot be made.
		- Change name
		- Change each password_parameter to "DEFAULT"

pgx_drop_profile(profile_name name, if_exists boolean, cascade boolean)	void	Delete an existing profile. Specify the profile name to be deleted in profile_name. Specifying the default profile will result in an error. if_exists specifies whether an error occurs when a nonexistent profile name is specified in profile_name. If true, no error. if_exists is optional. Default value is false. cascade specifies whether an error occurs when a profile assigned to a user is specified in profile_name. If false, an error will occur. If true, unassign all of the profiles before deleting them, and any unassigned users will be assigned the default profile instead. cascade is optional. Default value is false. [Example when specifying true] Specify arguments explicitly. (Example of named notation) `SELECT pgx_drop_profile('test_profile', cascade => true);`

B.3.2 User Management Functions

Name	Return type	Description
pgx_assign_profile_to_user(user_name name, profile_name name)	void	Assigns an existing profile to a user. Specify the assignee user in user_name. If you specify a user name that does not exist, an error will occur. Specify the profile name to be assigned in profile_name. If you specify a profile name that does not exist, an error will occur.
pgx_lock_user(user_name name)	void	Lock the user explicitly.

Name	Return type	Description
		Specify the user name in user_name. If you specify a user name that does not exist, an error will occur. If locked by this function, the lock period is indefinite. If you want to unlock it, you need to unlock it with the pgx_unlock_user function. If you lock a user that is already locked, the lock state is overwritten and becomes an indefinite lock even if it was previously a finite lock.
pgx_unlock_user(user_name name)	void	Explicitly unlock the user. The type of lock (profile reference lock or permanent lock) does not matter. Specify the user name in user_name. If you specify a user name that does not exist, an error will occur. Specifying an unlocked user does not result in an error.
pgx_make_password_expire(user_name name, expireat timestamptz)	void	Expires the specified user's password immediately. Alternatively, specify the time to expire. Specify the user name in user_name. If you specify a user name that does not exist, an error will occur. expireat can be a time that expires.expireat is optional. If omitted, execution time is specified.

B.4 Data Masking Control Functions

The table below lists the functions that can be used for data masking.

Table B.3 Data masking control functions

Name	Return type	Description
pgx_alter_confidential_policy	boolean	Changes masking policies
pgx_create_confidential_policy	boolean	Creates masking policies
pgx_drop_confidential_policy	boolean	Deletes masking policies
pgx_enable_confidential_policy	boolean	Enables or disables masking policies
pgx_update_confidential_values	boolean	Changes replacement characters when full masking is specified for masking type

B.4.1 pgx_alter_confidential_policy

Description

Changes masking policies

Format

The format varies depending on the content to be changed. The format is shown below.

- Common format

```
common_arg:
[schema_name    := 'schemaName',]
table_name    := 'tableName',
policy_name     := 'policyName'
```

- Add a masking target to a masking policy

```
pgx_alter_confidential_policy(
commonArg,
[action        := 'ADD_COLUMN', ]
column_name    := 'colName'
[, function_type    := 'FULL'] |
[, function_type    := 'PARTIAL', partialOpt] |
[, function_type    := 'REGEXP', regexpOpt]
)


partialOpt:
function_parameters    := 'maskingFmt'


regexpOpt:
regexp_pattern        := 'regexpPattern',
regexp_replacement    := 'regexpReplacementChar',
[, regexp_flags        := 'regexpFlags']
```

- Delete a masking target from a masking policy

```
pgx_alter_confidential_policy(
commonArg,
action        := 'DROP_COLUMN',
column_name    := 'colName'
)
```

- Change the masking condition

```
pgx_alter_confidential_policy(
commonArg,
action        := 'MODIFY_EXPRESSION',
expression    := 'expr'
)
```

- Change the content of a masking policy set for a masking target

```
pgx_alter_confidential_policy(
commonArg,
action        := 'MODIFY_COLUMN',
column_name    := 'colName'
[, function_type    := 'FULL'] |
[, function_type    := 'PARTIAL', partialOpt] |
[, function_type    := 'REGEXP', regexpOpt]
)


partialOpt:
function_parameters    := 'maskingFmt'


regexpOpt:
regexp_pattern        := 'regexpPattern',
regexp_replacement        := 'regexpReplacementChar',
[, regexp_flags        := 'regexpFlags']
```

- Change the masking policy description

```
pgx_alter_confidential_policy(
commonArg,
action        := 'SET_POLICY_DESCRIPTION',
policy_description    := 'policyDesc'
)
```

- Change the masking target description

```
pgx_alter_confidential_policy(
commonArg,
action        := 'SET_COLUMN_DESCRIPTION',
column_name    := 'colName',
column_description    := 'colDesc'
)
```

Argument

The argument varies depending on the content to be changed. Details are as follows.

- Common arguments

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
All	schema_name	varchar(63)	Schema name of table for which a masking policy is applied	'public'
	table_name	varchar(63)	Name of table for which a masking policy is applied	Mandatory
	policy_name	varchar(63)	Masking policy name	Mandatory

- Add a masking target to a masking policy

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
All	action	varchar(63)	'ADD_COLUMN'	'ADD_COLUMN'
	column_name	varchar(63)	Masking target name	Mandatory
	function_type	varchar(63)	Masking type - 'FULL': Full masking - 'PARTIAL': Partial masking - 'REGEXP': Regular expression masking	'FULL'
Partial masking	function_parameters	varchar(1024)	Masking format for partial masking	Mandatory
Regular expression masking	regexp_pattern	varchar(1024)	Search pattern for regular expression masking	Mandatory
	regexp_replacement	varchar(1024)	Replacement character/string for regular expression masking	Mandatory
	regexp_flags	varchar(20)	Regular expression flags	NULL

- Delete a masking target from a masking policy

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
All	action	varchar(63)	'DROP_COLUMN'	Mandatory
	column_name	varchar(63)	Masking target name	Mandatory

- Change the masking condition

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
All	action	varchar(63)	'MODIFY_EXPRESSION'	Mandatory
	expression	varchar(1024)	Masking condition to be changed	Mandatory

- Change the content of a masking policy set for a masking target

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
All	action	varchar(63)	'MODIFY_COLUMN'	Mandatory
	column_name	varchar(63)	Masking target name	Mandatory
	function_type	varchar(63)	Masking type - 'FULL': Full masking - 'PARTIAL': Partial masking - 'REGEXP': Regular expression masking	'FULL'
Partial masking	function_parameters	varchar(1024)	Masking format for partial masking	Mandatory
Regular expression masking	regexp_pattern	varchar(1024)	Search pattern for regular expression masking	Mandatory
	regexp_replacement	varchar(1024)	Replacement character/string for regular expression masking	Mandatory
	regexp_flags	varchar(20)	Regular expression flags	NULL

- Change the masking policy description

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
All	action	varchar(63)	'SET_POLICY_DESCRIPTION'	Mandatory
	policy_description	varchar(1024)	Masking policy description	Mandatory

- Change the masking target description

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
All	action	varchar(63)	'SET_COLUMN_DESCRIPTION'	Mandatory
	column_name	varchar(63)	Masking target name	Mandatory
	column_description	varchar(1024)	Masking target description	Mandatory

Details about whether arguments can be omitted are as follows.

Argument	Mandatory or optional									
	ADD_COLUMN			DROP_COLUMN	MODIFY_EXPRESSION	MODIFY_COLUMN			SET_POLICY_DESCRIPTION	SET_COLUMN_DESCRIPTION
	Full masking	Partial masking	Regular expression masking			Full masking	Partial masking	Regular expression masking		
schema_name	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
table_name	N	N	N	N	N	N	N	N	N	N
policy_name	N	N	N	N	N	N	N	N	N	N
action	Y	Y	Y	N	N	N	N	N	N	N
column_name	N	N	N	N	-	N	N	N	-	N
function_type	Y	N	N	-	-	Y	N	N	-	-
expression	-	-	-	-	N	-	-	-	-	-
policy_description	-	-	-	-	-	-	-	-	N	-
column_description	-	-	-	-	-	-	-	-	-	N
function_parameters	-	N	-	-	-	-	N	-	-	-
regexp_pattern	-	-	N	-	-	-	-	N	-	-
regexp_replacement	-	-	N	-	-	-	-	N	-	-
regexp_flags	-	-	Y	-	-	-	-	Y	-	-

Y: Can be omitted; N: Cannot be omitted; -: Ignored when specified

Return value

Return value	Description
TRUE	Ended normally
FALSE	Ended abnormally

Execution example 1

Adding masking policy p1 to masking target c2

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action :=
'ADD_COLUMN', column_name := 'c2', function_type := 'PARTIAL', function_parameters := 'VVVFVVVVFVVVV,
VVV-VVVV-VVVV, *, 4, 11');
 pgx_alter_confidential_policy
--------------------------------
 t
(1 row)
```

Execution example 2

Deleting masking target c1 from masking policy p1

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action :=
'DROP_COLUMN', column_name := 'c1');
 pgx_alter_confidential_policy
--------------------------------
```

```
 t
(1 row)
```

Execution example 3

Changing the masking condition for masking policy p1

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action :=
'MODIFY_EXPRESSION', expression := 'false');
 pgx_alter_confidential_policy
--------------------------------
 t
(1 row)
```

Execution example 4

Changing the content of masking policy p1 set for masking target c2

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action :=
'MODIFY_COLUMN', column_name := 'c2', function_type := 'FULL');
 pgx_alter_confidential_policy
--------------------------------
 t
(1 row)
```

Execution example 5

Changing the description of masking policy p1

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action :=
'SET_POLICY_DESCRIPTION', policy_description := 'this policy is an example.');
 pgx_alter_confidential_policy
--------------------------------
 t
(1 row)
```

Execution example 6

Changing the description of masking target c2

```
postgres=# select pgx_alter_confidential_policy(table_name := 't1', policy_name := 'p1', action :=
'SET_COLUMN_DESCRIPTION', column_name := 'c2', column_description := 'c2 column is FULL.');
 pgx_alter_confidential_policy
--------------------------------
 t
(1 row)
```

Description

- The arguments for the pgx_alter_confidential_policy system management function can be specified in any order.
- The action parameters below can be specified. When action parameters are omitted, ADD_COLUMN is applied.

Parameter	Description
ADD_COLUMN	Adds a masking target to a masking policy.
DROP_COLUMN	Deletes a masking target from a masking policy.
MODIFY_EXPRESSION	Changes expression.
MODIFY_COLUMN	Changes the content of a masking policy set for a masking target.
SET_POLICY_DESCRIPTION	Changes policy_description.

Parameter	Description
SET_COLUMN_DESCRIPTION	Changes column_description.

- The function_parameters argument is enabled when the function_type is PARTIAL. If the function_type is other than PARTIAL, it will be ignored.

- The arguments below are enabled when the function_type is REGEXP. If the function_type is other than REGEXP, these arguments will be ignored.

  - regexp_pattern

  - regexp_replacement

  - regexp_flags

See

- Refer to "String Constants" in the PostgreSQL Documentation for information on the strings to specify for arguments.

- Refer to "POSIX Regular Expressions" in the PostgreSQL Documentation and check pattern, replacement, and flags for information on the values that can be specified for regexp_pattern, regexp_replacement, and regexp_flags.

B.4.2 pgx_create_confidential_policy

Description

Creates masking policies

Format

The format varies depending on the masking type. The format is shown below.

```
pgx_create_confidential_policy(
[schema_name        := 'schemaName',]
table_name        := 'tableName',
policy_name         := 'policyName',
expression        := 'expr'
[, enable            := 'policyStatus']
[, policy_description    := 'policyDesc']
[, column_name        := 'colName'
     [, function_type    := 'FULL'] |
     [, function_type    := 'PARTIAL', partialOpt] |
     [, function_type    := 'REGEXP', regexpOpt]
     [, column_description    := 'colDesc']
])


partialOpt:
function_parameters    := 'maskingFmt'


regexpOpt:
regexp_pattern        := 'regexpPattern',
regexp_replacement        := 'regexpReplacementChar',
[, regexp_flags        := 'regexpFlags']
```

Argument

Details are as follows.

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
All	schema_name	varchar(63)	Schema name of table for which the masking policy is created	'public'
	table_name	varchar(63)	Name of table for which the masking policy is created	Mandatory
	policy_name	varchar(63)	Masking policy name	Mandatory
	expression	varchar(1024)	Masking condition	Mandatory
	enable	boolean	Masking policy status - 't': Enabled - 'f': Disabled	't'
	policy_description	varchar(1024)	Masking policy description	NULL
	column_name	varchar(63)	Masking target name	NULL
	function_type	varchar(63)	Masking type - 'FULL': Full masking - 'PARTIAL': Partial masking - 'REGEXP': Regular expression masking	'FULL'
	column_description	varchar(1024)	Masking target description	NULL
Partial masking	function_parameters	varchar(1024)	Masking format for partial masking	Mandatory
Regular expression masking	regexp_pattern	varchar(1024)	Search pattern for regular expression masking	Mandatory
	regexp_replacement	varchar(1024)	Replacement character/string for regular expression masking	Mandatory
	regexp_flags	varchar(20)	Regular expression flags	NULL

Details about whether arguments can be omitted are as follows.

Argument	Mandatory or optional		
	Full masking	Partial masking	Regular expression masking
schema_name	Y	Y	Y
table_name	N	N	N
policy_name	N	N	N
expression	N	N	N
enable	Y	Y	Y
policy_description	Y	Y	Y
column_name	Y	Y	Y
function_type	Y	Y	Y
column_description	Y	Y	Y
function_parameters	-	N	-
regexp_pattern	-	-	N
regexp_replacement	-	-	N

Argument	Mandatory or optional		
	Full masking	Partial masking	Regular expression masking
regexp_flags	-	-	Y

Y: Can be omitted; N: Cannot be omitted; -: Ignored when specified

Return value

Return value	Description
TRUE	Ended normally
FALSE	Ended abnormally

Execution example 1

Creating masking policy p1 that does not contain a masking target

```
postgres=# select pgx_create_confidential_policy(table_name := 't1', policy_name := 'p1', 
expression := '1=1');
 pgx_create_confidential_policy
--------------------------------
 t
(1 row)
```

Execution example 2

Creating masking policy p1 that contains masking target c1 of which the masking type is full masking

```
postgres=# select pgx_create_confidential_policy(schema_name := 'public', table_name := 't1', 
policy_name := 'p1', expression := '1=1', enable := 't', policy_description := 'this policy is an 
example.', column_name := 'c1', function_type := 'FULL', column_description := 'c1 column is FULL.');
 pgx_create_confidential_policy
--------------------------------
 t
(1 row)
```

Execution example 3

Creating masking policy p1 that contains masking target c2 of which the masking type is partial masking

```
postgres=# select pgx_create_confidential_policy( table_name := 't1', policy_name := 'p1', 
expression := '1=1', column_name := 'c2', function_type := 'PARTIAL', function_parameters := 
'VVVFVVVVFVVVV, VVV-VVVV-VVVV, *, 4, 11');
 pgx_create_confidential_policy
--------------------------------
 t
(1 row)
```

Execution example 4

Creating masking policy p1 that contains masking target c3 of which the masking type is regular expression masking

```
postgres=# select pgx_create_confidential_policy( table_name := 't1', policy_name := 'p1', 
expression := '1=1', column_name := 'c3', function_type := 'REGEXP', regexp_pattern := '(.*)(@.*)', 
regexp_replacement := 'xxx\2', regexp_flags := 'g');
 pgx_create_confidential_policy
--------------------------------
 t
(1 row)
```

Description

- The arguments for the pgx_create_confidential_policy system management function can be specified in any order.
- If column_name is omitted, only masking policies that do not contain masking target will be created.
- One masking policy can be created for each table. Use the pgx_alter_confidential_policy system management function to add a masking target to a masking policy.
- The function_parameters argument is enabled when the function_type is PARTIAL. If the function_type is other than PARTIAL, it will be ignored.
- The arguments below are enabled when the function_type is REGEXP. If the function_type is other than REGEXP, these arguments will be ignored.
  - regexp_pattern
  - regexp_replacement
  - regexp_flags

Note

If a table for which a masking policy is to be applied is deleted, delete the masking policy as well.

See

- Refer to "String Constants" in the PostgreSQL Documentation for information on the strings to specify for arguments.
- Refer to "POSIX Regular Expressions" in the PostgreSQL Documentation and check pattern, replacement, and flags for information on the values that can be specified for regexp_pattern, regexp_replacement, and regexp_flags.

B.4.3 pgx_drop_confidential_policy

Description

Deletes masking policies

Format

```
pgx_drop_confidential_policy(
[schema_name    := 'schemaName', ]
table_name     := 'tableName',
policy_name    := 'policyName'
)
```

Argument

Details are as follows.

Argument	Data type	Description	Default value
schema_name	varchar(63)	Schema name of table for which a masking policy is deleted	'public'
table_name	varchar(63)	Name of table for which a masking policy is deleted	Mandatory
policy_name	varchar(63)	Masking policy name	Mandatory

Details about whether arguments can be omitted are as follows.

Argument	Mandatory or optional
schema_name	Y
table_name	N
policy_name	N

Y: Can be omitted; N: Cannot be omitted

Return value

Return value	Description
TRUE	Ended normally
FALSE	Ended abnormally

Execution example

Deleting masking policy p1

```
postgres=# select pgx_drop_confidential_policy(table_name := 't1', policy_name := 'p1');
 pgx_drop_confidential_policy
-------------------------------
 t
(1 row)
```

Description

The arguments for the pgx_drop_confidential_policy system management function can be specified in any order.

Note

If a table for which a masking policy is to be applied is deleted, delete the masking policy as well.

See

Refer to "String Constants" in the PostgreSQL Documentation for information on the strings to specify for arguments.

B.4.4 pgx_enable_confidential_policy

Description

Enables or disables masking policies

Format

```
pgx_enable_confidential_policy(
[schema_name    := 'schemaName', ]
table_name     := 'tableName',
policy_name    := 'policyName',
enable        := 'policyStatus'
)
```

Argument

Details are as follows.

Argument	Data type	Description	Default value
schema_name	varchar(63)	Schema name of table for which a masking policy is enabled or disabled	'public'
table_name	varchar(63)	Name of table for which a masking policy is enabled or disabled	Mandatory
policy_name	varchar(63)	Masking policy name	Mandatory
enable	boolean	Masking policy status - 't': Enabled - 'f': Disabled	Mandatory

Details about whether arguments can be omitted are as follows.

Argument	Mandatory or optional
schema_name	Y
table_name	N
policy_name	N
enable	N

Y: Can be omitted; N: Cannot be omitted

Return value

Return value	Description
TRUE	Ended normally
FALSE	Ended abnormally

Execution example

Enabling masking policy p1

```
postgres=# select pgx_enable_confidential_policy(table_name := 't1', policy_name := 'p1', enable :=
't');
 pgx_enable_confidential_policy
--------------------------------
 t
(1 row)
```

Description

The arguments for the pgx_enable_confidential_policy system management function can be specified in any order.

See

Refer to "String Constants" in the PostgreSQL Documentation for information on the strings to specify for arguments.

B.4.5 pgx_update_confidential_values

Description

Changes replacement characters when full masking is specified for masking type

Format

```
pgx_update_confidential_values(
[number_value    := 'numberValue']
[, char_value    := 'charValue']
[, varchar_value    := 'varcharValue']
[, date_value    := 'dateValue']
[, ts_value    := 'tsValue']
)
```

Argument

Details are as follows.

Argument	Data type	Description
number_value	integer	Replacement character in numeric type
char_value	varchar(1)	Replacement character in char type
varchar_value	varchar(1)	Replacement character in varchar type
date_value	date	Replacement character in date type
ts_value	timestamp	Replacement character in timestamp type

Return value

Return value	Description
TRUE	Ended normally
FALSE	Ended abnormally

Execution example

Using '*' as a replacement character in char type and varchar type

```
postgres=# select pgx_update_confidential_values(char_value := '*', varchar_value := '*');
 pgx_update_confidential_values
--------------------------------
 t
(1 row)
```

Description

- The arguments for the pgx_update_confidential_values system management function can be specified in any order.
- Specify one or more arguments for the pgx_update_confidential_values system management function. A replacement character is not changed for an omitted argument.

See

Refer to "String Constants" in the PostgreSQL Documentation for information on the strings to specify for arguments.

B.5 VCI Data Load Control Function

The table below lists the function that loads VCI data to buffer cache.

Table B.4 VCI data load control function

Name	Return type	Description
pgx_prewarm_vci(vci_index regclass)	int8	Loads the VCI data to buffer cache.

pgx_prewarm_vci loads the specified VCI data to buffer cache and returns the number of blocks of the loaded VCI data.

The aggregation process using VCI may take time immediately after an instance is started, because the VCI data has not been loaded to buffer cache. Therefore, the first aggregation process can be sped up by executing pgx_prewarm_vci after an instance is started.

The amount of memory required for preloading is the number of blocks returned by pgx_prewarm_vci multiplied by the size of one block.

This function can only be executed if the user has reference privilege to the VCI index and execution privilege to the pg_prewarm function.

B.6 High-Speed Data Load Control Functions

The table below lists the functions that can be used for high-speed data load.

Table B.5 High-speed data load control functions

Name	Return type	Description
pgx_loader	bigint	Creates dynamic shared memory, starts parallel workers and loads data
pgx_loader_recovery	smallint	Resolves in-doubt transactions

The pgx_loader command executes the above functions internally.

Appendix C System Catalogs

Describes the system catalog of Fujitsu Enterprise Postgres.

See

Refer to "System Catalogs" under "Internals" in the PostgreSQL Documentation for information on other system catalogs.

C.1 pgx_profile

A system catalog for managing profile information. -1 is stored if DEFAULT is specified for each parameter in the profile, and -2 is stored if UNLIMITED is specified.

Column	Type	Description
oid	oid	Profile identifier (Primary key constraint)
prfname	name	Profile name (Unique constraint, NOT NULL constraint)
prfpasswordlifetime	integer	Value of PASSWORD_LIFE_TIME (seconds) (NOT NULL constraint)
prfpasswordgracetime	integer	Value of PASSWORD_GRACE_TIME (seconds) (NOT NULL constraint)
prfpasswordreusetime	integer	Value of PASSWORD_REUSE_TIME (seconds) (NOT NULL constraint)
prfpasswordreusemax	integer	Value of PASSWORD_REUSE_MAX (seconds) (NOT NULL constraint)
prfpasswordlocktime	integer	Value of PASSWORD_LOCK_TIME (seconds) (NOT NULL constraint)
prffailedloginattempts	integer	Value of FAILED_LOGIN_ATTEMPTS (seconds) (NOT NULL constraint)
prfpasswordallowhashed	integer	Value of PASSWORD_ALLOW_HASHED (NOT NULL constraint)

C.2 pgx_user_profile

A system catalog that manages profile-related information associated with users. When DROP ROLE, the corresponding row is automatically deleted.

Column	Type	References	Description
userprfroleid	oid	pg_authid.oid	User's identifier (Primary key constraint)
userprfprfid	oid	pgx_profile.oid	Identifier of the profile assigned to the user
userprfaccountlock	smallint		User lock state (NOT NULL constraint) 0: Not locked 1: Lock (Refer to PASSWORD_LOCK_TIME in profile for duration) 2: Lock (Unlimited)

Column	Type	References	Description
userprfpasswordstatus	char		Password status Updated if there are any changes at login (NOT NULL constraint) o: Current valid password g: In grace period e: Expired
userprflockdate	timestamp with time zone		Time the user was locked
userprfpasswordsetat	timestamp with time zone		Time you updated your current password
userprfpasswordexpire	timestamp with time zone		Password expiration time

C.3 pgx_password_history

A system catalog for managing password history. When DROP ROLE is executed, the corresponding row is automatically deleted.

Since this catalog contains passwords, general users cannot refer to it, just like pg_authid. This system catalog is basically updated when the password is updated, and if the password reuse limit is finite, unnecessary history deletion is also performed. Old passwords that have been removed from the catalog are treated as having never been used.

passhistpassword is saved in the format specified by the password_encryption parameter.

Column	Type	Description
passhistroleid	oid	Identifier of the user whose password was set Set unique constraint with (passhistroleid, passhistpassword).
passhistpassword	text	Encrypted password Save rolpassword value in pg_authid as history (NOT NULL constraint)
passhistpasswordsetat	timestamp with time zone	Time when the password was updated (NOT NULL constraint)

Appendix D System Views

This appendix describes how to use the system views in Fujitsu Enterprise Postgres.

See

Refer to "System Views" under "Internals" in the PostgreSQL Documentation for information on other system views.

D.1 pgx_tablespaces

The pgx_tablespaces view provides information related to the encryption of tablespaces.

Table D.1 pgx_tablespaces view

Column	Type	References	Description
spctablespace	oid	pg_tablespace.oid	Tablespace OID
spcencalgo	text		Tablespace encryption algorithm

The spcencalgo string displays one of the following values:

- none: Tablespace is not encrypted
- AES128: AES with key length of 128 bits
- AES256: AES with key length of 256 bits

D.2 pgx_stat_lwlock

The pgx_stat_lwlock view displays statistics related to lightweight locks, with each type of content displayed on a separate line.

Table D.2 pgx_stat_lwlock view

Column	Type	Description
lwlock_name	name	Name of the lightweight lock
total_waits	bigint	Number of waits caused by the lightweight lock
total_wait_time	double precision	Number of milliseconds spent in waits caused by the lightweight lock
stats_reset	timestamp with timezone	Last time at which this statistics was reset

D.3 pgx_stat_latch

The pgx_stat_latch view displays statistics related to latches, with each type of wait information within Fujitsu Enterprise Postgres displayed on a separate line.

Table D.3 pgx_stat_latch view

Column	Type	Description
latch_name	name	Name of the latch
total_waits	bigint	Number of waits caused a wait
total_wait_time	double precision	Number of milliseconds spent in waits caused by the latch
stats_reset	timestamp with timezone	Last time at which this statistic was reset

D.4 pgx_stat_walwriter

The pgx_stat_walwriter view displays statistics related to WAL writing, in a single line.

Table D.4 pgx_stat_walwriter view

Column	Type	Description
dirty_writes	bigint	Number of times old WAL buffers were written to the disk because the WAL buffer was full when WAL records were added
writes	bigint	Number of WAL writes
write_blocks	bigint	Number of WAL write blocks
total_write_time	double precision	Number of milliseconds spent on WAL writing
stats_reset	timestamp with timezone	Last time at which this statistic was reset

D.5 pgx_stat_sql

The pgx_stat_sql view displays statistics related to SQL statement executions, with each type of SQL statement displayed on a separate line.

Table D.5 pgx_stat_sql view

Column	Type	Description
selects	bigint	Number of SELECT statements executed In database multiplexing mode, this number includes the SELECT statements executed in Mirroring Controller. Mirroring Controller executes the SELECT statement using the interval specified for the heartbeat_interval of the server definition file (milliseconds).
inserts	bigint	Number of INSERT statements executed
deletes	bigint	Number of DELETE statements executed
updates	bigint	Number of UPDATE statements executed
selects_with_parallelism	bigint	Number of times parallel scan was used in SELECT statements
inserts_with_parallelism	bigint	Not used
deletes_with_parallelism	bigint	Not used
updates_with_parallelism	bigint	Not used
copies_with_parallelism	bigint	Not used
declares	bigint	Number of DECLARE statements executed (number of cursor OPENs)
fetches	bigint	Number of FETCH statements executed
checkpoints	bigint	Number of CHECKPOINT statements executed
clusters	bigint	Number of CLUSTER statements executed
copies	bigint	Number of COPY statements executed
reindexes	bigint	Number of REINDEX statements executed
truncates	bigint	Number of TRUNCATE statements executed
locks	bigint	Number of times a lock occurred
stats_reset	timestamp with timezone	Last time at which this statistic was reset

D.6 pgx_stat_gmc

The pgx_stat_gmc view provides information about the GMC areas.

Table D.6 pgx_stat_gmc view

Column	Type	Description
searches	bigint	Number of times the cache table is searched.
hits	bigint	Number of times the cache table is hit.
size	bigint	The current amount of memory (bytes) used in the GMC area.
stats_reset	timestamp with timezone	Last time these statistics were reset.

D.7 pgx_stat_progress_loader

The pgx_stat_progress_loader view provides overall progress information for pgx_loader command.

The pgx_stat_progress_loader view displays the sum of the progress information of the back-end processes and the number of parallel worker processes when pgx_loader runs.

Table D.7 pgx_stat_progress_loader view

Column	Type	Description
pid	integer	Process ID of the backend.
datid	Oid	Oid of the database to connect to.
datname	name	Name of the database to connect to.
relid	Oid	Oid of the table to load.
command	text	Command executes the load process. (Always "COPY FROM"for pgx_loader)
type	text	Type of data source for the load operation.
bytes_processed	bigint	Size of the data at the end of the load. (Backend and worker totals)
bytes_total	bigint	Size of the data to load. (Backend and worker totals)
tuples_processed	bigint	Number of rows that have completed loading. (Backend and worker totals)
tuples_excluded	bigint	Number of rows skipped during the load process. (Backend and worker totals)

Appendix E Tables Used by Transparent Data Encryption

This appendix explains tables tables used by the transparent data encryption feature.

E.1 pgx_tde_master_key

Provides information about the master encryption key being used when using the key management system as a keystore.

Column	Type	Description
local_key_id	integer	Sequence of encryption keys used internally by Fujitsu Enterprise Postgres
kms_name	text	Key management system name
key_name	text	Not currently used
key_id	text	Key ID in key management system
start_time	timestamp	Time the key was activated
status	enum	in-use: Current master encryption key in use used: Master encryption key used in the past scheduled: A new encryption key that was requested to update the master encryption key and is to be used.

Execution example

```
postgres=# select * from pgx_tde_master_key;
local_key_id | kms_name  | key_name|              key_id                  |    start_time     | status
-------------+-----------+---------+--------------------------------------+-------------------+-----
           1  | mykmipsvr |        | a0eebc99-9c0b-0000-0000-000000000000 |2022-12-16 05:43:49| in-use
(1 row)
```

Note

Do not use queries with "*" in the selection list, as the order of the columns may change or columns may be added.

Appendix F Tables Used by Data Masking

This appendix explains tables used by the data masking feature.

Note

These tables are updated by the data masking control function, so do not use SQL statements to directly update these tables.

F.1 pgx_confidential_columns

This table provides information on masking target for which masking policies are set.

Column	Type	Description
schema_name	varchar(63)	Schema name of table for which a masking policy is applied
table_name	varchar(63)	Name of table for which a masking policy is applied
policy_name	varchar(63)	Masking policy name
column_name	varchar(63)	Masking target name
function_type	varchar(63)	Masking type - 'FULL': Full masking - 'PARTIAL': Partial masking - 'REGEXP': Regular expression masking
function_parameters	varchar(1024)	Masking format for partial masking
regexp_pattern	varchar(1024)	Search pattern for regular expression masking
regexp_replacement	varchar(1024)	Replacement character/string for regular expression masking
regexp_flags	varchar(20)	Regular expression flags
column_description	varchar(1024)	Masking target description

Execution example

```
postgres=# select * from pgx_confidential_columns;
 schema_name | table_name | policy_name | column_name | function_type |
function_parameters           | regexp_pattern | regexp_replacement | regexp_flags |
column_description
-------------+------------+-------------+-------------+---------------
+---------------------------------------+----------------+--------------------+-------------
+--------------------
 public     | t1        | p1         | c1         | FULL         |
|                |                    |             |
 public      | t1         | p1          | c2          | PARTIAL       | VVVFVVVVFVVVV, VVV-VVVV-VVVV,
*, 4, 11 |                |                    |             |
(2  row)
```

F.2 pgx_confidential_policies

This table provides information on masking policies.

Column	Type	Description
schema_name	varchar(63)	Schema name of table for which a masking policy is applied

Column	Type	Description
table_name	varchar(63)	Name of table for which a masking policy is applied
policy_name	varchar(63)	Masking policy name
expression	varchar(1024)	Masking condition
enable	boolean	Masking policy status - 't': Enabled - 'f': Disabled
policy_description	varchar(1024)	Masking policy description

Execution example

```
postgres=# select * from pgx_confidential_policies;
 schema_name | table_name | policy_name | expression | enable | policy_description
-------------+------------+-------------+------------+--------+--------------------
 public      | t1         | p1          | 1=1        | t      |
(1 row)
```

F.3 pgx_confidential_values

This table provides information on replacement characters when full masking is specified for masking type.

Column	Data type	Description	Default value
number_value	integer	Numeric	0
char_value	varchar(1)	char type	Spaces
varchar_value	varchar(1)	varchar type	Spaces
date_value	date	date type	'1970-01-01'
timestamp_value	timestamp	timestamp type	'1970-01-01 00:00:00'

Execution example

```
postgres=# select * from pgx_confidential_values;
 number_value | char_value | varchar_value | date_value |      ts_value
--------------+------------+---------------+------------+---------------------
            0 |            |               | 1970-01-01 | 1970-01-01 00:00:00
(1 row)
```

Appendix G Tables Used by High-Speed Data Load

This appendix describes the tables used by high-speed data load.

G.1 pgx_loader_state

The pgx_loader_state table provides information about transactions prepared by high-speed data load.

Column	Type	Description
id	serial	Unique identifier. This value is assigned from the pgx_loader_state_id_seq sequence.
gid	text	Global transaction identifier assigned to a transaction.
state	text	State of the transaction. The value can be one of the following: - commit: The prepared transaction has been committed. - rollback: The prepared transaction is in in-doubt state.
leader_pid	integer	Process ID of the backend process (leader process) that executed the pgx_loader control function.
role_oid	integer	Role identifier (OID). A prepared transaction can only be completed by the same user who executed the original transaction or by a superuser.
relation_oid	integer	Object identifier (OID).

Note

The pgx_loader_state table and pgx_loader_state_id_seq sequence are updated by high-speed data load. Do not update these database objects directly using SQL.

Appendix H Starting and Stopping the Web Server Feature of WebAdmin

To use WebAdmin for creating and managing a Fujitsu Enterprise Postgres instance on a server where Fujitsu Enterprise Postgres is installed, you must first start the Web server feature of WebAdmin.

- Using WebAdmin in a single-server configuration

  You must start the Web server on the server on which Fujitsu Enterprise Postgres and WebAdmin are installed.

- Using WebAdmin in a multiserver configuration

  You must start the Web server on all servers on which WebAdmin has been installed.

This appendix describes how to start and stop the Web server feature of WebAdmin.

Note that "<*x*>" in paths indicates the product version.

See

Refer to "Installing WebAdmin in a Multiserver Configuration" in the Installation and Setup Guide for Server for information on multiserver installation.

H.1 Starting the Web Server Feature of WebAdmin

Follow the procedure below to start the Web server feature of WebAdmin.

1. Change to superuser

   Acquire superuser privileges on the system.

   Example

   ```
   $ su -
   Password:******
   ```

2. Start the Web server feature of WebAdmin

   Execute the WebAdminStart command to start the Web server feature of WebAdmin.

   Example

   If WebAdmin is installed in /opt/fsepv<*x*>webadmin:

   ```
   # cd /opt/fsepv<x>webadmin/sbin
   # ./WebAdminStart
   ```

H.2 Stopping the Web Server Feature of WebAdmin

This section describes how to stop the Web server feature of WebAdmin.

Follow the procedure below to stop the Web server feature of WebAdmin.

1. Change to superuser

   Acquire superuser privileges on the system.

   Example

   ```
   $ su -
   Password:******
   ```

2. Stop the Web server feature of WebAdmin

   Execute the WebAdminStop command to stop the Web server feature of WebAdmin.

Example

If WebAdmin is installed in /opt/fsepv<*x*>webadmin:

```
# cd /opt/fsepv<x>webadmin/sbin
# ./WebAdminStop
```

Appendix I WebAdmin Wallet

This appendix describes how to use the Wallet feature of WebAdmin.

When a remote instance or a standby instance is created, it is necessary to provide user name and password for authentication with the remote machine or the database instance.

The Wallet feature in WebAdmin is a convenient way to create and store these credentials.

Once created, these credentials can be repeatedly used in one or more instances.

I.1 Creating a Credential

1. In the [My Wallet] tab, click ✚. The [New credential] page will be displayed.

2. Enter the information for the credentials.

   Credential name, User name and Password should not contain hazardous characters. Refer to “Appendix J WebAdmin Disallow User Inputs Containing Hazardous Characters”.

   - [Credential name]: Name of the credential

     The name must meet the conditions below:

     - Maximum of 16 characters
     - The first character must be an ASCII alphabetic character
     - The other characters must be ASCII alphanumeric characters

   - [User name]: The operating system user name or database instance user name that will be used later
   - [Password]: Password for the user
   - [Confirm password]: Reenter the password.

3. Click ✔ to store the credential.

I.2 Using a Credential

Once a credential is created in the Wallet, it can be used during remote instance creation or standby instance creation.

If you select the credential saved in "I.1 Creating a Credential" in [Operating system credential], the user name and password will be automatically populated.

Appendix J WebAdmin Disallow User Inputs Containing Hazardous Characters

WebAdmin considers the following as hazardous characters, which are not allowed in user inputs.

```
| (pipe sign)

& (ampersand sign)

; (semicolon sign)

$ (dollar sign)

% (percent sign)

@ (at sign)

' (single apostrophe)

" (quotation mark)

\' (backslash-escaped apostrophe)

\" (backslash-escaped quotation mark)

<> (triangular parenthesis)

() (parenthesis)

+ (plus sign)

CR (Carriage return, ASCII 0x0d)

LF (Line feed, ASCII 0x0a)

, (comma sign)

\ (backslash)
```

Appendix K Collecting Failure Investigation Data

If the cause of an error that occurs while building the environment or during operations is unclear, data must be collected for initial investigation.

This appendix describes how to collect data for initial investigation.

Use the pgx_fjqssinf command to collect data for initial investigation.

See

Refer to the Reference for information on the pgx_fjqssinf command.

Index