FUJITSU

FUJITSU Enterprise Postgres 9.5

Operation Guide

Linux

J2UL-2118-02ENZ0(00)
June 2016

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 Data Masking

Describes the data masking feature.

Chapter 7 Periodic Operations

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

Chapter 8 Streaming Replication Using WebAdmin

Describes how to create a streaming replication cluster using WebAdmin.

Chapter 9 Installing and Operating the In-memory Feature

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

Chapter 10 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 Views

Describes how to use the system view in FUJITSU Enterprise Postgres.

Appendix D Tables Used by Data Masking

Describes the tables used by the data masking feature.

Appendix E Activating and Stopping the Web Server Feature of WebAdmin

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

Appendix F WebAdmin Wallet

Describes how to use the Wallet feature of WebAdmin.

Appendix G 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

```
Second edition: June 2016
First edition: March 2016
```

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 switchover or failover operation using database multiplexing, refer to "Database Multiplexing Mode" in the Cluster Operation Guide.

Operation management using GUI tools

This involves managing operations using the WebAdmin and pgAdmin GUI tools.

- 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.

- Management using pgAdmin

  When developing applications and maintaining the database, you can use pgAdmin to perform simple operations on database objects, such as:

  - Rebuild indexes and update statistics
  - Create, delete, and update database objects

  In addition, from pgAdmin of FUJITSU Enterprise Postgres, you can use the expanded features provided by FUJITSU Enterprise Postgres on the PostgreSQL SQL commands.

  See

  Refer to pgAdmin Help for information on the expanded features of pgAdmin provided by FUJITSU Enterprise Postgres.

Operation management using commands

You can use commands for configuring and operating the database and managing operations. However, note that if you start managing operations using commands, you cannot switch to WebAdmin-based operation management.

Note

You cannot combine WebAdmin and server commands to perform the following operations:

- Use WebAdmin to operate an instance created using the initdb command
- 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.

- You can perform backup and restoration in pgAdmin, but the backup data obtained with WebAdmin and pgx_dmpall is not compatible with the backup data obtained with pgAdmin.

- Refer to pgAdmin Help for other notes on pgAdmin.

Features used in each phase

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

Operation		GUI-based operation	Command-based operation
Setup	Creating an instance	WebAdmin	initdb command
	Creating a standby instance	WebAdmin WebAdmin performs a base backup of the source instance and creates a standby instance.	pg_basebackup command
	Modifying the configuration file	WebAdmin	Directly edit the configuration file
Instance start		WebAdmin	pg_ctl command
Database creation		pgAdmin	Specify using the DDL statement, and define using psql and applications
Database backup		WebAdmin pgx_dmpall command	pgx_dmpall command
Monitoring	Database failure	WebAdmin(*1)	Messages output to the system log (*1)
	Disk space	WebAdmin (*1) (*2)	OS-provided df command (*1)
	Connection status	pgAdmin	psql command (*3)
Database recovery		WebAdmin	pgx_rcvall command

*1: Operations can be monitored using operation management middleware (such as Systemwalker Centric Manager).

*2: A warning is displayed when disk usage reaches 80%.

*3: This command searches for pg_stat_activity in the standard statistics views and monitors the state.

1.2 Activating WebAdmin

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

1.2.1 Logging in to WebAdmin

This section describes how to log in to WebAdmin.

User environment

One of the following browsers is required for using WebAdmin:

- Internet Explorer 8.0 or later

Activation URL for WebAdmin

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

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

- *hostNameOrIpAddress*: The host name or IP address of the server where FUJITSU Enterprise Postgres is installed.

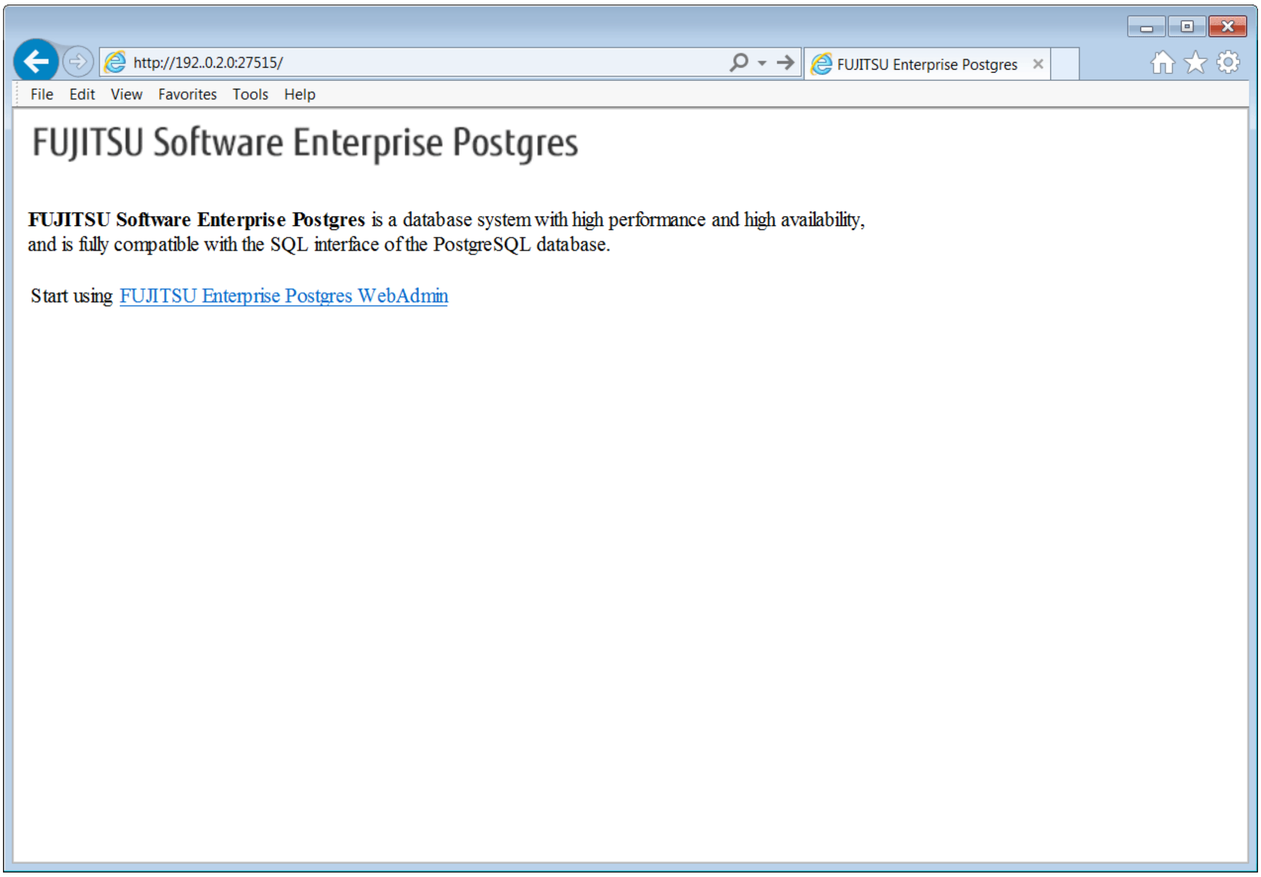
- *portNumber*: The port number of WebAdmin. The default port number is 27515.

Example

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

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

The activation URL window shown below is displayed.



http://192..0.2.0:27515/
FUJITSU Enterprise Postgres
File Edit View Favorites Tools Help

FUJITSU Software Enterprise Postgres

FUJITSU Software Enterprise Postgres is a database system with high performance and high availability, and is fully compatible with the SQL interface of the PostgreSQL database.

Start using FUJITSU Enterprise Postgres WebAdmin

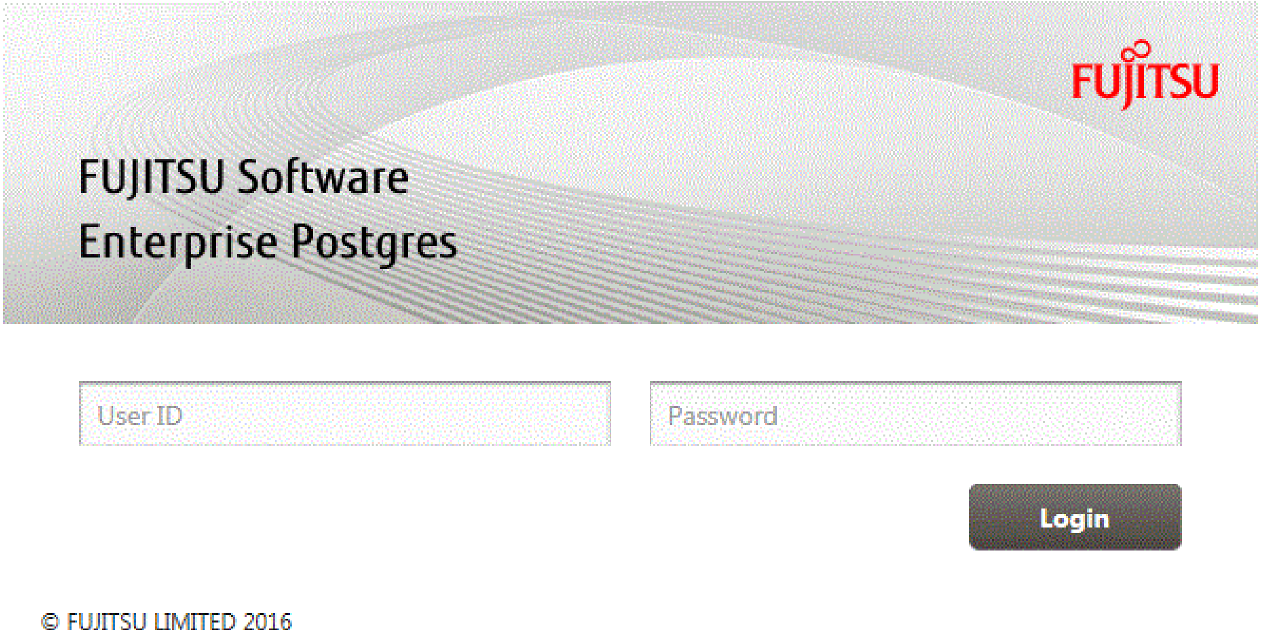
Point

- You must activate the Web server feature of WebAdmin before using WebAdmin.

- Refer to "Appendix E Activating and Stopping the Web Server Feature of WebAdmin" for information on how to activate the Web server feature of WebAdmin.

Log in to WebAdmin

Click [FUJITSU Enterprise Postgres WebAdmin] in the activation URL window to activate WebAdmin and display the [Log in] window.



To log in, specify the following values:

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

Point

Use the OS user account as the user ID of the instance administrator. Refer to "Creating an Instance Administrator" in the Installation and Setup Guide for Server for details.

1.3 Starting pgAdmin

This section describes how to start pgAdmin, how to add an instance required for managing a database, and how to connect to and disconnect from the instance.

You can use pgAdmin on the Windows client.

1.3.1 Starting pgAdmin

This section explains how to start pgAdmin if you are using it from the product "FUJITSU Enterprise Postgres Client (*AA*bit) *x.y* SP*z*" (where *AA* is "32" or "64", *x.y* and *z* are the version numbers (*x.y* SP*z*)).

Windows(R) 8 or Windows Server(R) 2012

From the [Start] screen, start [pgAdmin III (*AA*bit) (*x.y* SP*z*)].

Windows(R) 8.1 or Windows Server(R) 2012 R2

From the [Apps] view, start [pgAdmin III (*AA*bit) (*x.y* SP*z*)].

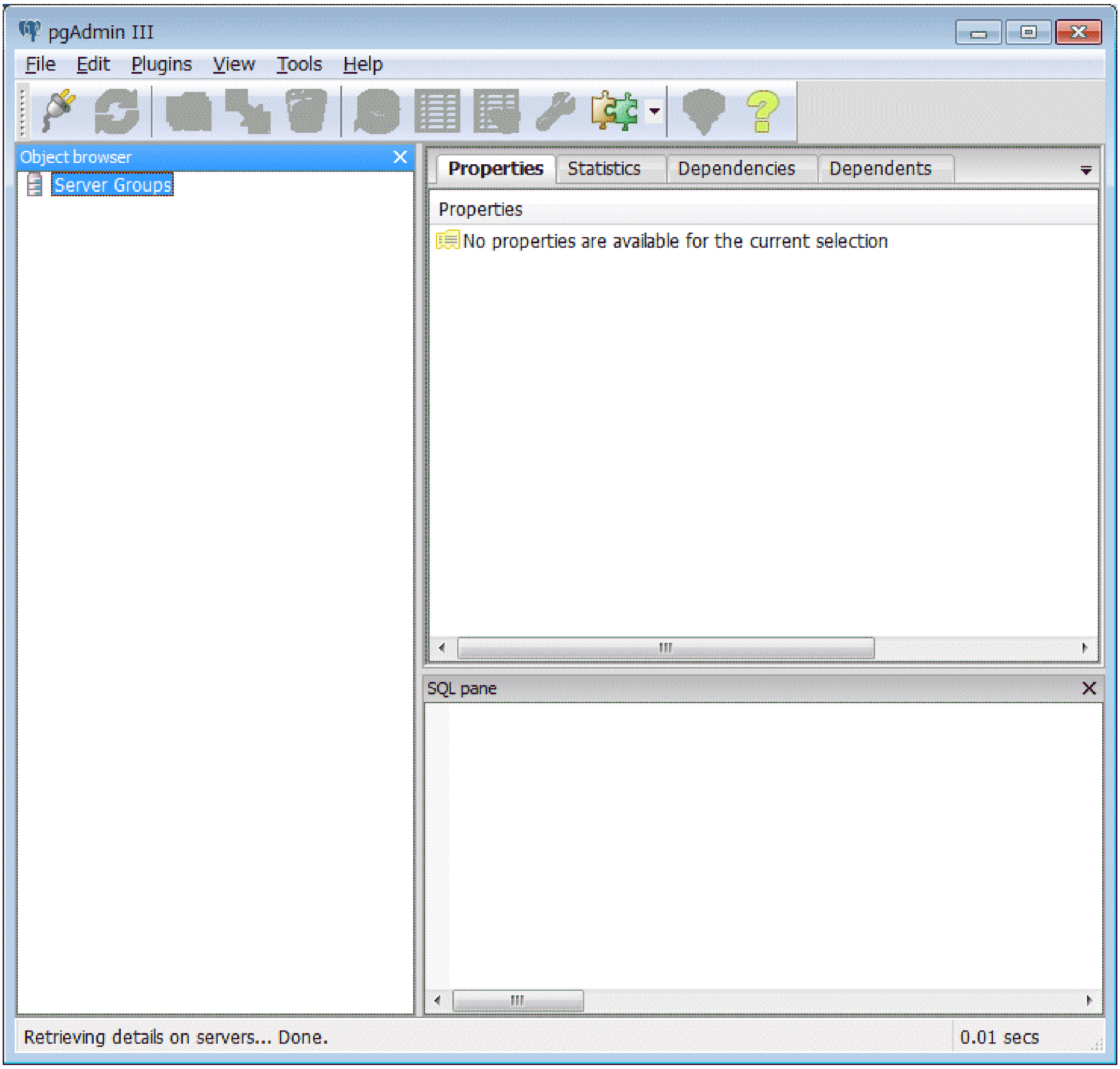
Windows(R) 10

Click [Start] >> [All apps] >> [FUJITSU Enterprise Postgres Client(AAbit)] and start [pgAdmin III (*AA*bit) (*x.y* SP*z*)].

Other operating systems

Click [Start] >> [All Programs] >> [FUJITSU Enterprise Postgres Client(*AA*bit) *x.y* SP*z*] and start [pgAdmin III (*AA*bit) (*x.y* SP*z*)].

The following window is displayed when pgAdmin starts.


pgAdmin III
File Edit Plugins View Tools Help
Object browser
Server Groups
Properties Statistics Dependencies Dependents
Properties
No properties are available for the current selection
SQL pane
Retrieving details on servers... Done.
0.01 secs

Note

- You must start the instance to be connected to before using pgAdmin.

- Refer to "2.1 Starting and Stopping an Instance" for information on how to start an instance.

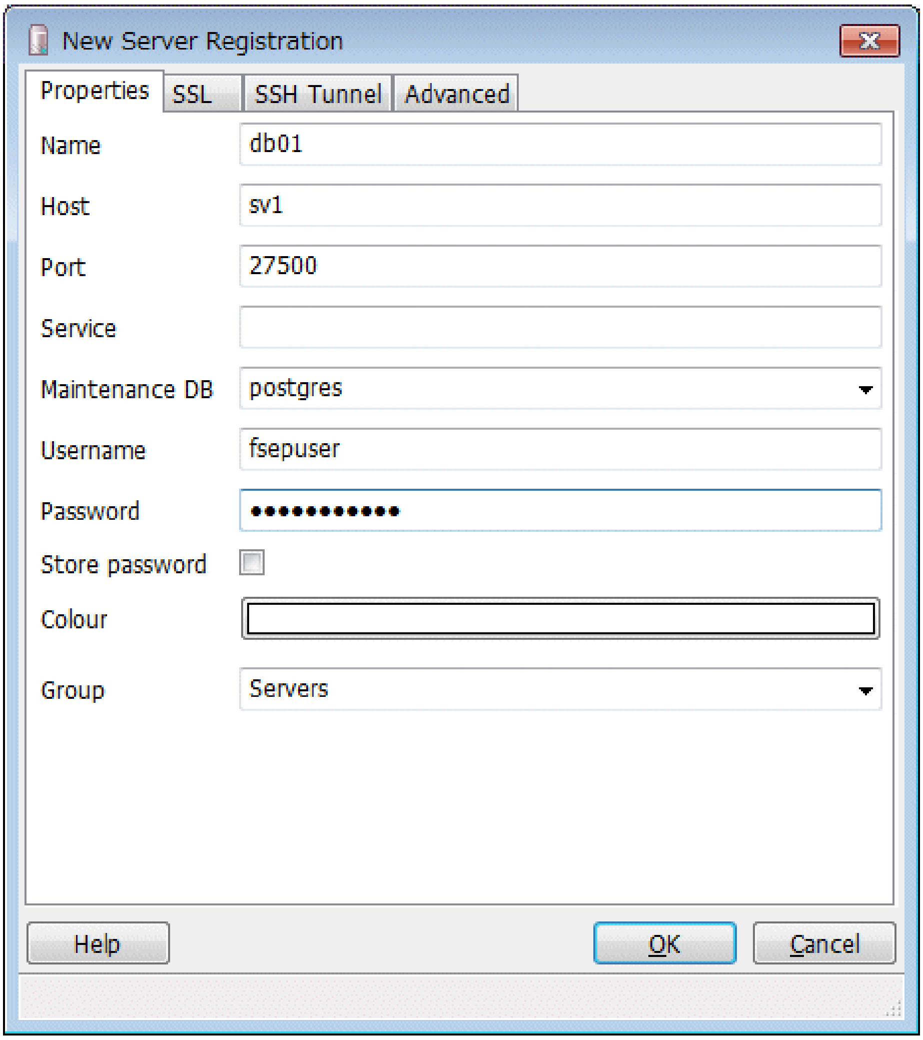
- Adobe(R) Reader(R) X is required for browsing the manual from [FUJITSU Enterprise Postgres Help] in pgAdmin.

1.3.2 Adding an Instance

This section describes how to add an instance to be connected to.

1. From the [File] menu in pgAdmin, click [Add Server].

2. In the [New Server Registration] window, specify a value for each item.



Name: db01

Host: sv1

Port: 27500

Service:

Maintenance DB: postgres

Username: fsepuser

Password: ●●●●●●●●●●●●

Store password: □

Colour:

Group: Servers

([Properties] tab)

- [Name]: Name of the instance to be managed
- [Host]: Host name or IP address of the server where FUJITSU Enterprise Postgres is installed
- [Port]: Port number of the instance
- [Username]: User name of the instance administrator
- [Password]: Password for the user name specified in [Username]

When you add an instance using pgAdmin, the instance is automatically connected to immediately after the addition is completed.

Note

If you select [Store password], a file storing the FUJITSU Enterprise Postgres connection password is created in the following location. Set the appropriate access permissions for the password file to protect it from unauthorized access.

- %APPDATA%\postgresql\pgpass.conf

1.3.3 Connecting/Disconnecting an Instance

This section describes how to connect pgAdmin to an instance, and how to disconnect it.

Note

To connect to an instance created using WebAdmin, you must first configure the settings in the [Client Authentication] window of WebAdmin to permit connection from pgAdmin.

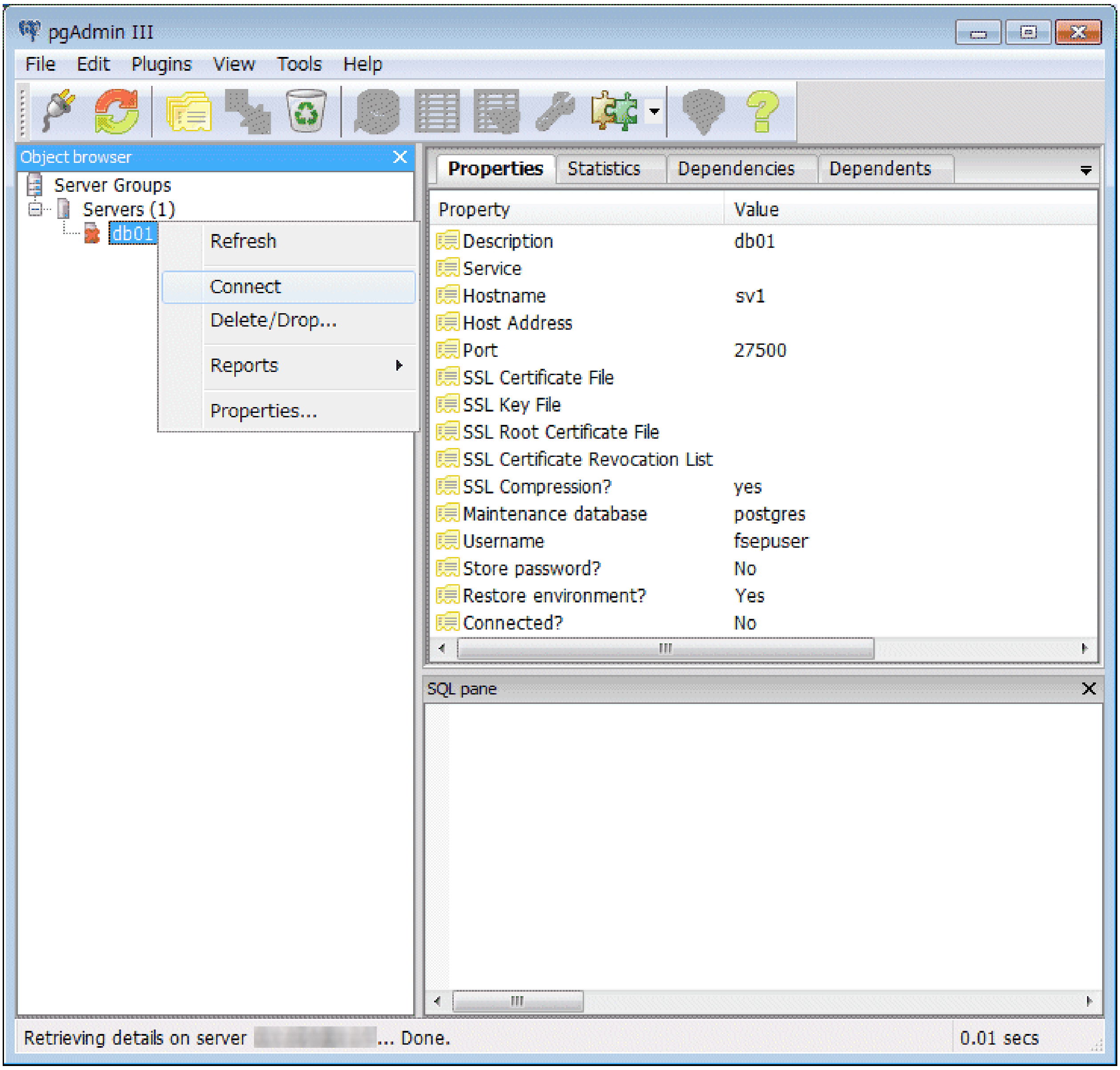
See

Refer to "Changing the settings" in the Installation and Setup Guide for Server for information on the [Client Authentication] window of WebAdmin.

Connecting to an instance

Starting pgAdmin does not connect it to any instance.

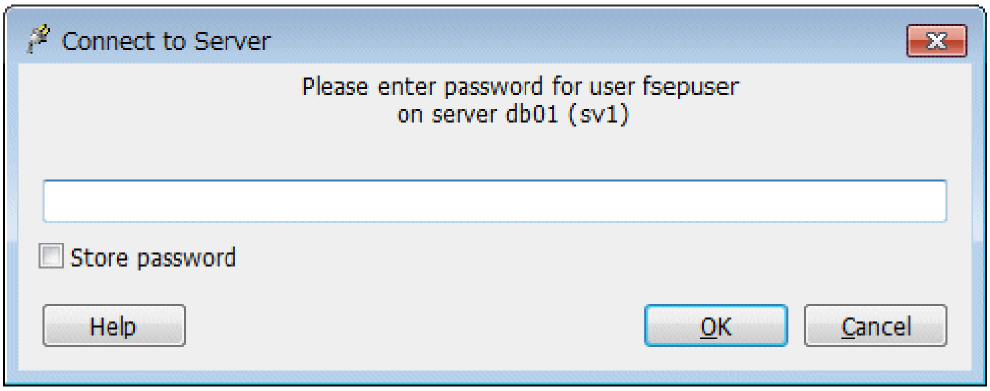
To connect to an instance, right-click the instance in [Object browser] and select [Connect].



Property	Value
Description	db01
Service	
Hostname	sv1
Host Address	
Port	27500
SSL Certificate File	
SSL Key File	
SSL Root Certificate File	
SSL Certificate Revocation List	
SSL Compression?	yes
Maintenance database	postgres
Username	fsepuser
Store password?	No
Restore environment?	Yes
Connected?	No

Retrieving details on server [redacted]... Done. 0.01 secs

If a password was not saved when the instance was added, the following password entry window is displayed.



Connect to Server

Please enter password for user fsepuser
on server db01 (sv1)

□ Store password

Disconnecting from an instance

To disconnect from an instance, right-click the server in [Object browser] in the pgAdmin window and select [Disconnect server].

1.4 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" under the procedure for creating an instance in "Using the initdb Command" in the Installation and Setup Guide for Server for information on the values to be set in 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 need to configure the environment variables.

  See

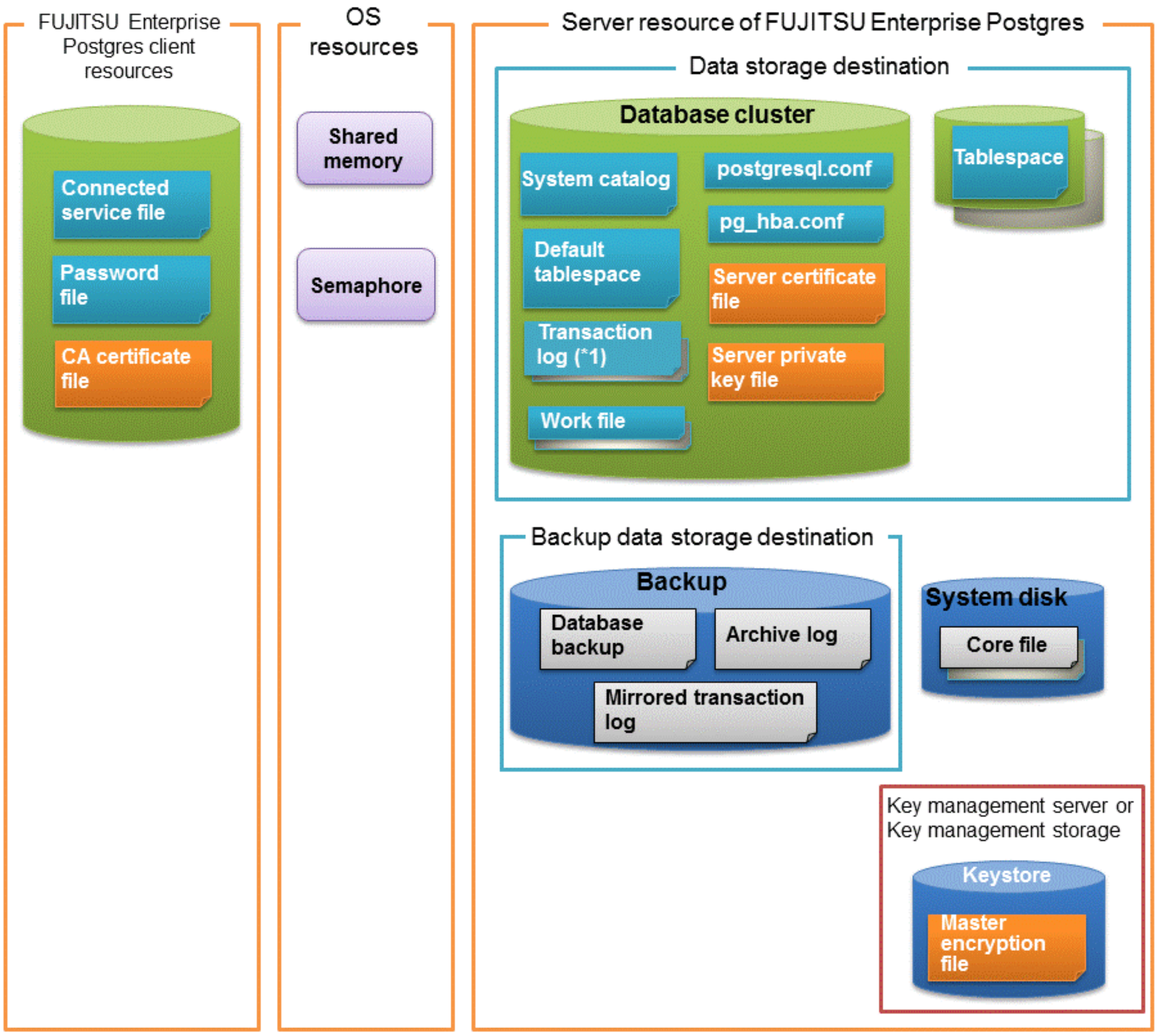
  - Refer to "PostgreSQL Client Applications" under "Reference" in the PostgreSQL Documentation, or "Reference" for information on client commands.
  - Refer to "Configuring Environment Variables" in the Installation and Setup Guide for Client for information on the values to be set in the environment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

1.5 Operating Environment of FUJITSU Enterprise Postgres

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

1.5.1 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
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.
Core file	FUJITSU Enterprise Postgres process core file that is output when an error occurs during an 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.

1.5.2 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 "<*xy*>" indicates the product version and level. **64-bit product** /opt/fsepv<*xy*>server64 **32-bit product** /opt/fsepv<*xy*>server32
Database cluster	Y	1	Specify using WebAdmin or server commands.
Tablespace	Y	Multiple	Specify using pgAdmin or the DDL statement.
Backup	Y	Multiple	Specify using WebAdmin or server commands.

File type	Required	Quantity	How to specify the location
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 use an NFS for UNIX-type files used in FUJITSU Enterprise Postgres except when creating a database space in a storage device on a network.
- When PRIMECLUSTER GDS is used, the PRIMECLUSTER GDS disk class cannot deploy the FUJITSU Enterprise Postgres resources below to the root class.
  Deploy these resources to the local class or the shared class.
  - Database cluster
  - Tablespace
  - Backup directory
- 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.

1.6 Notes on Compatibility of Applications Used for Operations

When you upgrade FUJITSU Enterprise Postgres to a newer version, there may be some affect 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.

Note

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.	

*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.

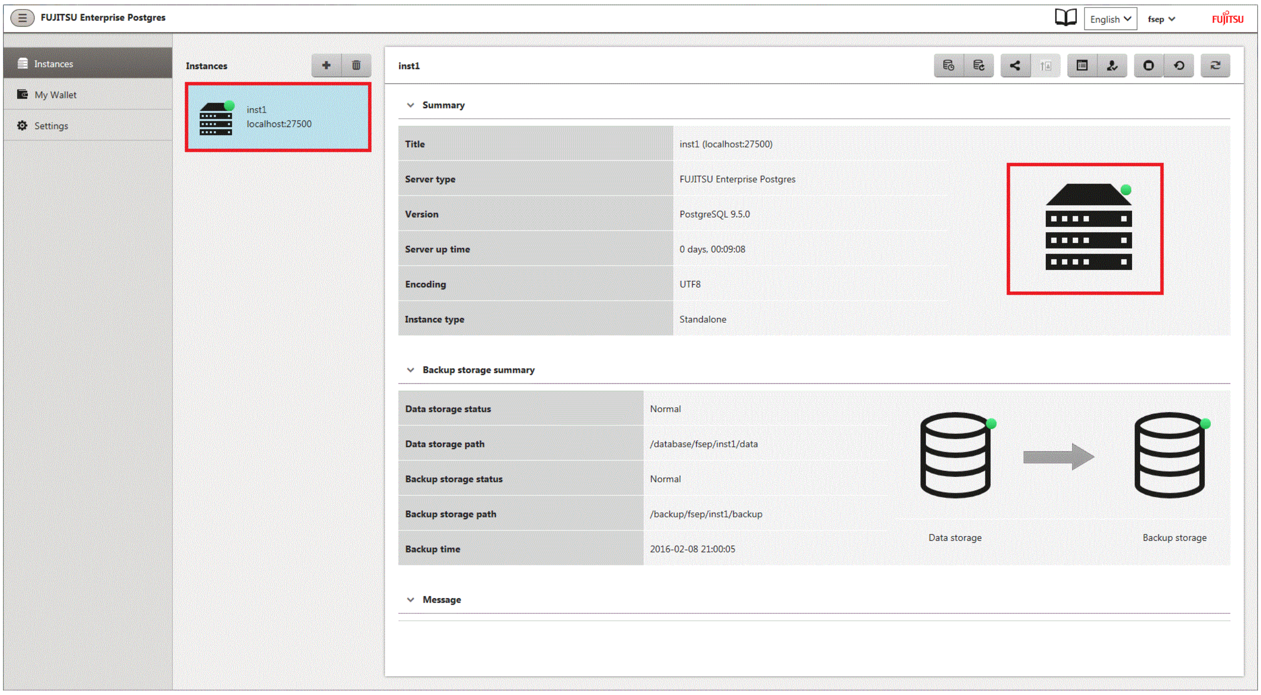
Checking the operating status of an instance

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.

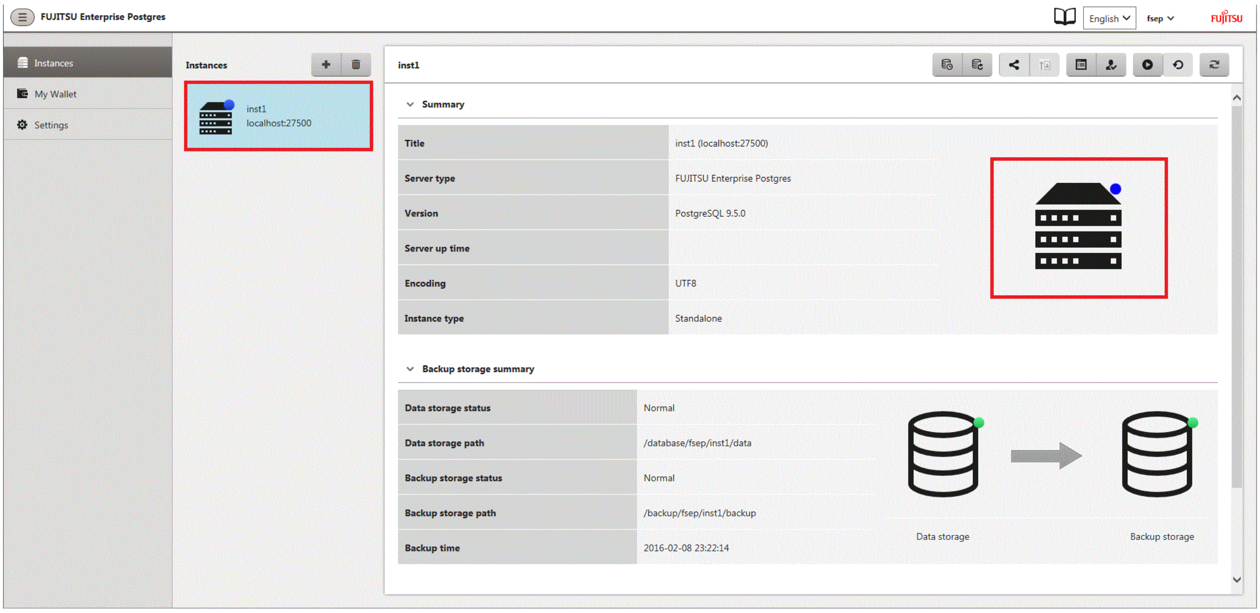
Figure 2.1 Status when an instance is active



Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	0 days, 00:09:08
Encoding	UTF8
Instance type	Standalone

Data storage status	Normal
Data storage path	/database/fsep/inst1/data
Backup storage status	Normal
Backup storage path	/backup/fsep/inst1/backup
Backup time	2016-02-08 21:00:05

Figure 2.2 Status when an instance is stopped



Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	
Encoding	UTF8
Instance type	Standalone

Data storage status	Normal
Data storage path	/database/fsep/inst1/data
Backup storage status	Normal
Backup storage path	/backup/fsep/inst1/backup
Backup time	2016-02-08 23:22:14

Note

- 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.

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.

- It is recommended to specify the -w option, which causes the command to return 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.

  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 with 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 -w -D /database/inst1
```

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.

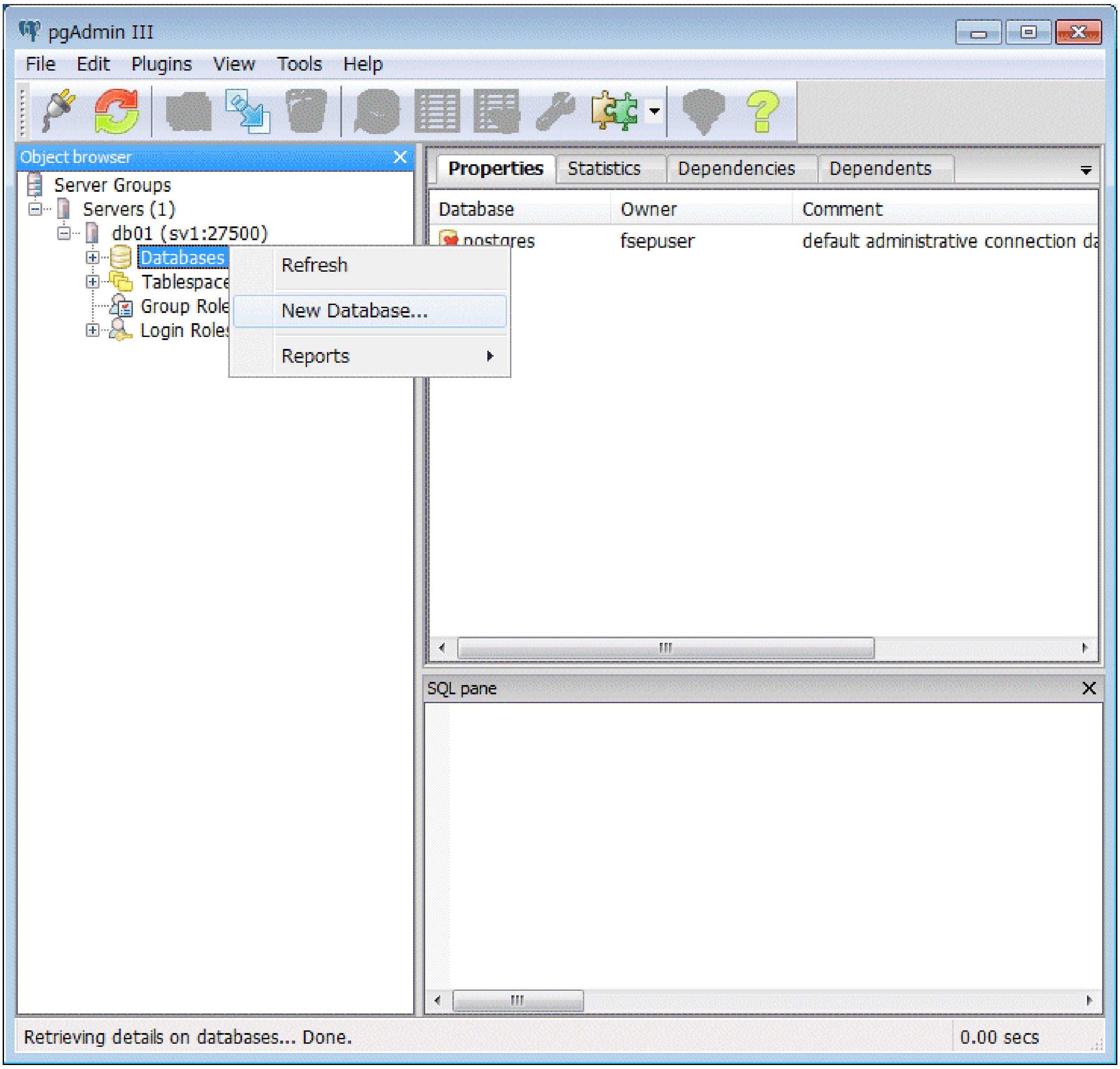
- 2.2.1 Using pgAdmin

- 2.2.2 Using Client Commands

2.2.1 Using pgAdmin

Follow the procedure below to define a database using pgAdmin.

1. In the pgAdmin window, right-click [Database] in [Object browser], and then click [New Database] to display a new database window.



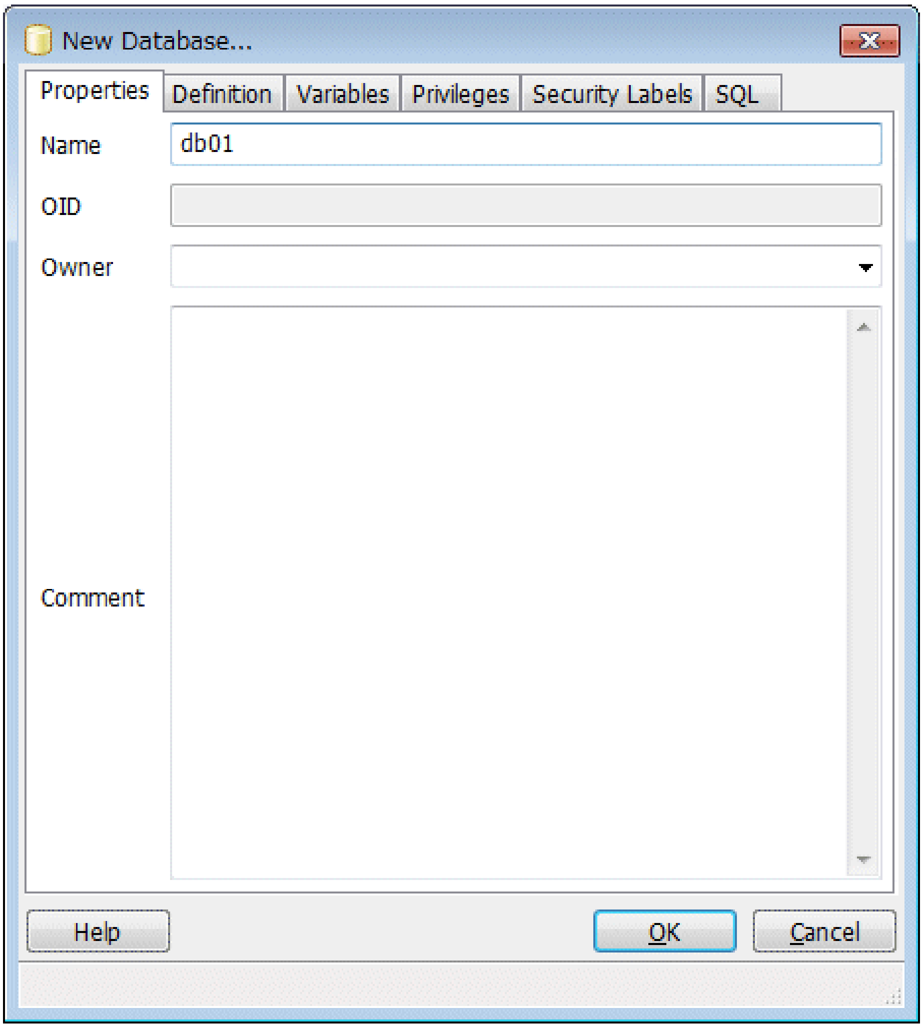
Database	Owner	Comment
postgres	fsepuser	default administrative connection da

Retrieving details on databases... Done. 0.00 secs

2. Specify appropriate values for the following items in the new database window.

   - [Properties] tab

     The following example illustrates creation of the database "db01".



Name: db01

   - [Name]: Name of the database to be managed

3. Click [OK] to create the database.

2.2.2 Using Client Commands

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 (9.5.2)
Type "help" for help.
```

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 the \l+ command, 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.

  The pgx_rcvall command is used for recovery.

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)

Note

- Backup operation cannot be performed on an instance that is part of a streaming replication cluster in standby mode.
- 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.

- 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

- 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.

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.

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.

Note

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.

Note

WebAdmin uses the labels "Data storage path", "Backup storage path" and "Transaction log path" to indicate "data storage destination", "backup data storage destination" and "transaction log storage destination" respectively. In this manual these terms are used interchangeably.

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 .

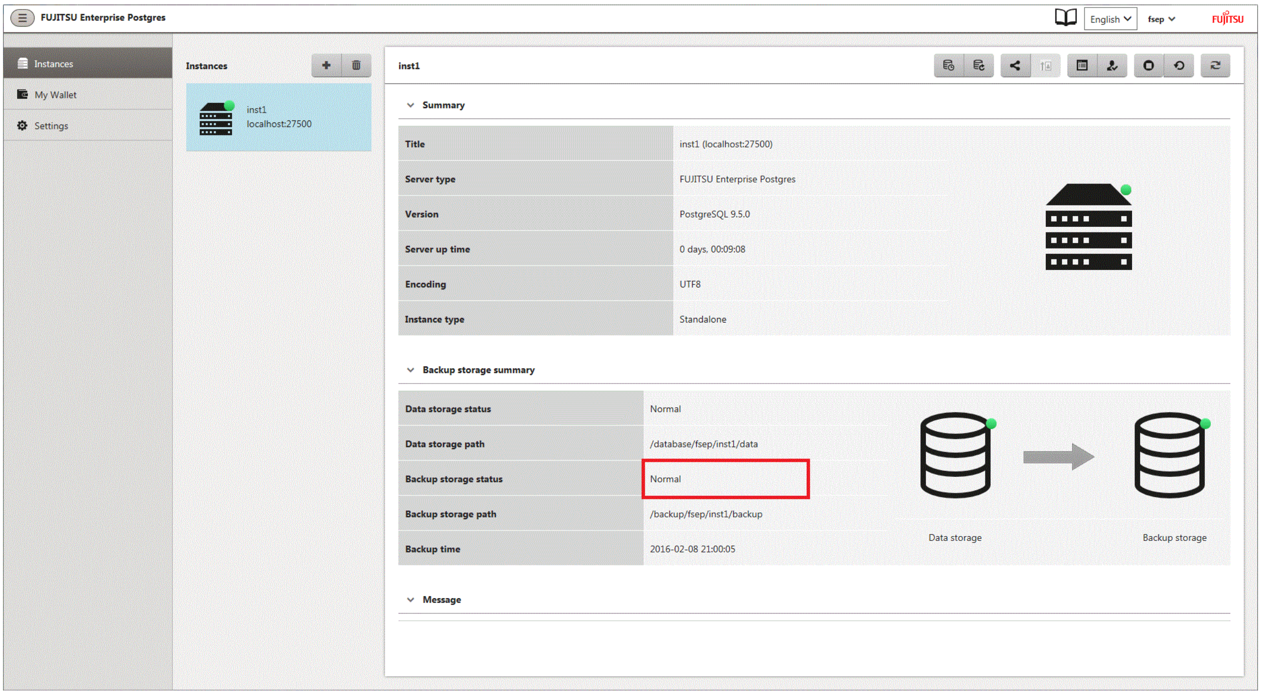
2. Back up the database

   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 destination] or [Backup data storage destination] 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.



Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	0 days, 00:09:08
Encoding	UTF8
Instance type	Standalone

Data storage status	Normal
Data storage path	/database/fsep/inst1/data
Backup storage status	Normal
Backup storage path	/backup/fsep/inst1/backup
Backup time	2016-02-08 21:00:05

Note

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.
wal_level	archive or hot_standby(*1)	Specify the output level for the transaction log. *1: hot_standby is a setting for streaming replication.
archive_mode	on	Specify the archive log mode. Specify [on] (execute).
archive_command	'*installationDirectory*/bin/pgx_xlogcopy.cmd "%p" "*backupDataStorageDestinationDirectory*/archived_xlog/%f"'	Specify the path name of the command that will save the transaction log and the storage destination.

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

Backup operation

Use the pgx_dmpall command to perform 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
2015-05-01 13:30:40     COMPLETE        /backup/inst1/2015-05-01_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_20150503_1');
LOG:  restore point "batch_20150503_1" created at 0/20000E8
STATEMENT:  select pg_create_restore_point('batch_20150503_1');
 pg_create_restore_point
-------------------------
```

```
 0/20000E8
(1 row)
```

Refer to "10.3.2 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

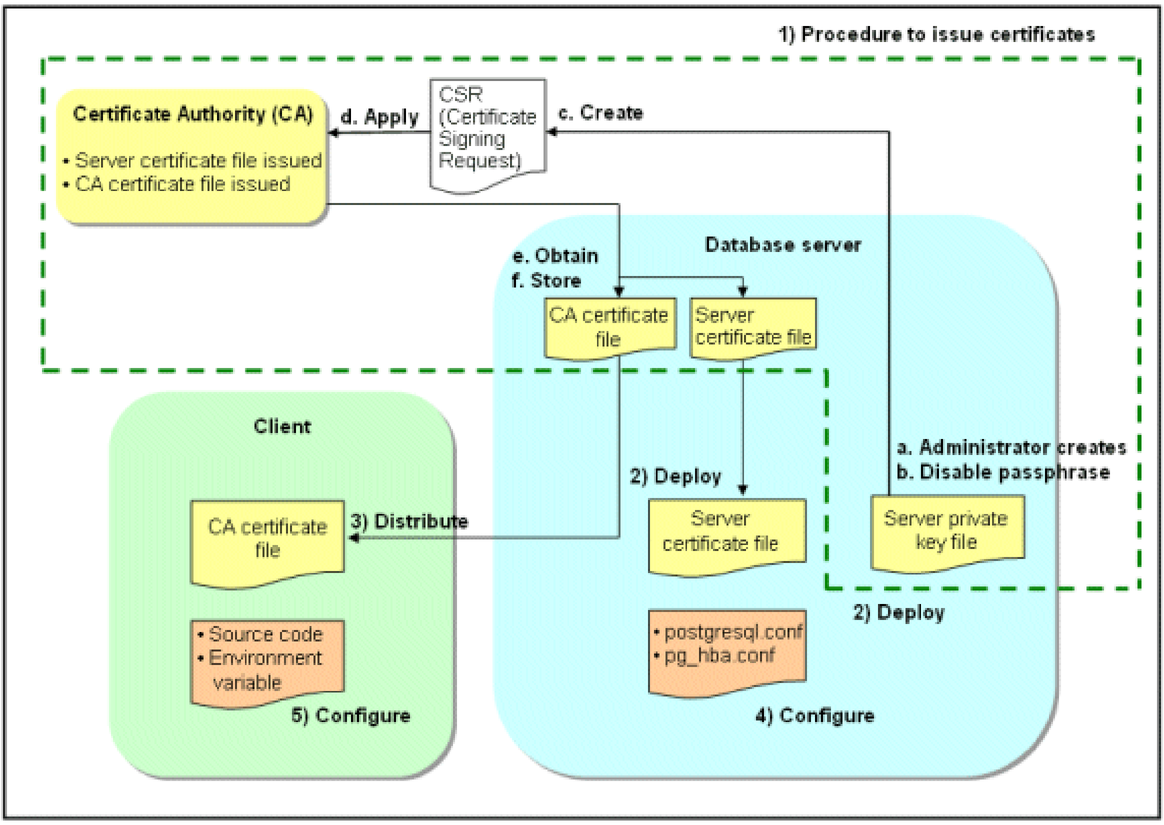
To encrypt communication data transferred between a client and a server, configure communication data encryption as described below. Communication data encryption not only protects the communication content, but it also guards against man-in-the-middle (MITM) attacks (for example, data and password theft 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, certificates with the same "Common Name" must be used. To ensure this, take one of the following actions:

- Create one server certificate, replicate it, and place a copy on each server used for database multiplexing.
- Create a server certificate with the same "Common Name" 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.

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.

Faster encryption and decryption based on hardware

TDE minimizes the overhead of encryption and decryption by using the AES-NI (Advanced Encryption Standard New Instructions) built into Intel(R) Xeon(R) processors since the 5600 series. This means that even in situations where previously the minimum encryption target was selected as a tradeoff between performance and security, it is now possible to encrypt all the data of an application.

You can reference a list of processors equipped with AES-NI on the following page at Intel Corporation's website:

http://ark.intel.com/search/advanced/?s=t&AESTech=true

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.

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.

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

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.

   ```
   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' tablespace_encryption_algorithm =
'AES256';
```

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.

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

Note

An encrypted tablespace cannot be created from the window used for creating the pgAdmin tablespace, or from the query tool. To create an encrypted tablespace, click [PSQL Console] from the [Plugins] menu and create an encrypted tablespace in the psql console window.

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 "C.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
```

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

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

Note

- To use WebAdmin for recovery, you must enable automatic opening of the keystore.
- Refer to "5.7 Backing Up and Restoring/Recovering the Database" after enabling or reconfiguring encryption to back up the database.
- 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

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 May 1, 2015.

  ```
  > pgx_dmpall -D /database/inst1
  > cp -p /key/store/location/keystore.ks /keybackup/keystore_20150501.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 May 5, 2015.

```
> psql -c "SELECT pgx_set_master_key('passphrase')" postgres
> cp -p /key/store/location/keystore.ks /keybackup/keystore_20150505.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 May 5, 2015.

```
> cp -p /keybackup/keystore_20150505.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 May 2015.

```
> cp -p /keybackup/keystore_20150501.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 May 1, 2015.

```
> pgx_dmpall -D /database/inst1
> cp -p /key/store/location/keystore.ks /keybackup/keystore_20150501.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 May 1, 2015.

```
> cp -p /keybackup/keystore_20150501.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 For this reason, the pg_restore 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_restore
  - 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, C, and the Mirroring Controller option.

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.

See

Refer to the Cluster Operation Guide for information on building a cluster system environment using failover operation.

5.10.2 Database Multiplexing Mode

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

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

- 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.

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 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.

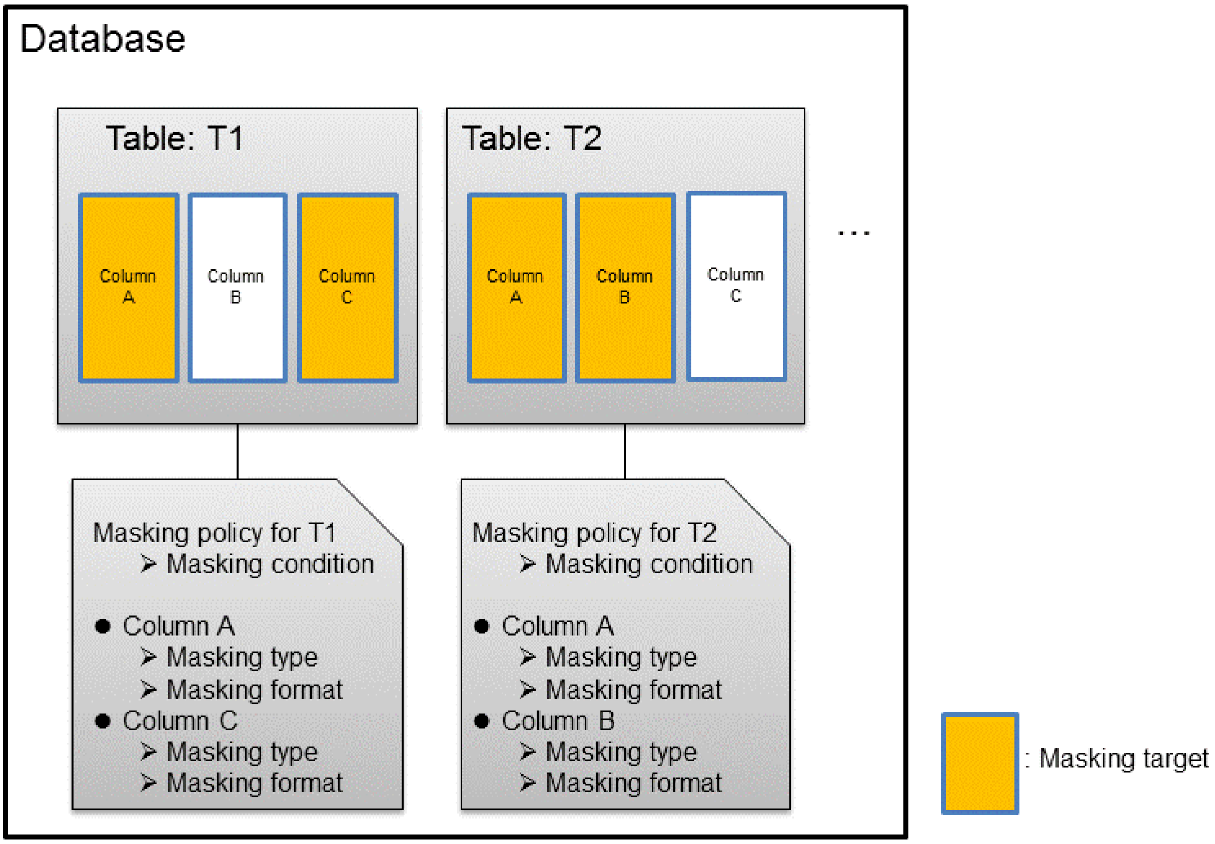
Note

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.

6.1 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 6.1 Masking policy



Note

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

6.1.1 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 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.

6.1.2 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.

6.1.3 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.

6.1.4 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 "6.3 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. - *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*.

Category	Method of specifying function_parameters
	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 "2010-10-10 10:10:10", it will be changed to "2010-10-10 00:00:00".

See

- Refer to "B.3.2 pgx_create_confidential_policy" for information on function_parameters.
- Refer to "6.3 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 "6.3 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.

6.2 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. Run 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.

6.2.1 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.3.2 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.

6.2.2 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.3.1 pgx_alter_confidential_policy" for information on the pgx_alter_confidential_policy system management function.

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

6.2.4 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.3.4 pgx_enable_confidential_policy" for information on the pgx_enable_confidential_policy system management function.

6.2.5 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.3.3 pgx_drop_confidential_policy" for information on the gx_drop_confidential_policy function.

6.3 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
Character type	character varying(*n*)	Y	Y	Y
	varchar(*n*)	Y	Y	Y

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

Chapter 7 Periodic Operations

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

7.1 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.

7.2 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

7.2.1 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%

7.2.2 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 "7.5 Reorganizing Indexes" for details.

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

  Execute backup using WebAdmin or the pgx_dmpall command.

7.3 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.

See

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

7.4 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 two methods are supported for monitoring connections that have been in the waiting status for an extended period:

- 7.4.1 Using the View (pg_stat_activity)
- 7.4.2 Using pgAdmin

7.4.1 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 state='idle in transaction' and current_timestamp >
cast(query_start + interval '60 minutes' as timestamp);
-[ RECORD 1 ]----+------------------------------
datid            | 13003
datname          | db01
pid              | 4638
usesysid         | 10
usename          | fsep
application_name | apl01
client_addr      | 192.33.44.15
client_hostname  |
client_port      | 27500
backend_start    | 2015-04-24 09:09:21.730641+09
xact_start       | 2015-04-24 09:09:23.858727+09
query_start      | 2015-04-24 09:09:23.858727+09
state_change     | 2015-04-24 09:09:23.858834+09
waiting          | f
state            | idle in transaction
backend_xid      |
backend_xmin     |
query            | begin;
```

See

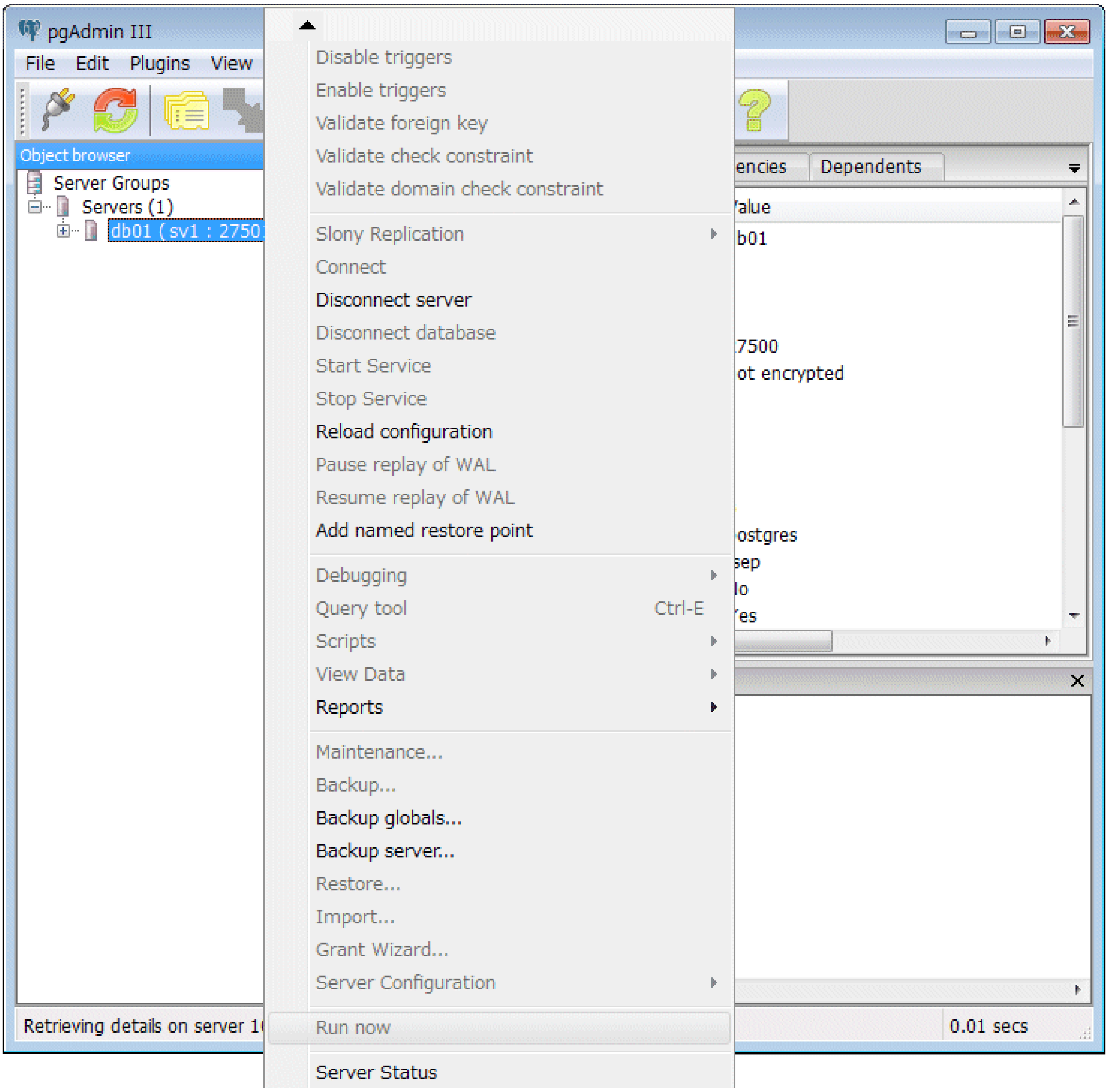
- Refer to "Notes on Application Compatibility" in the Application Development Guide for information on maintaining application compatibility.

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

7.4.2 Using pgAdmin

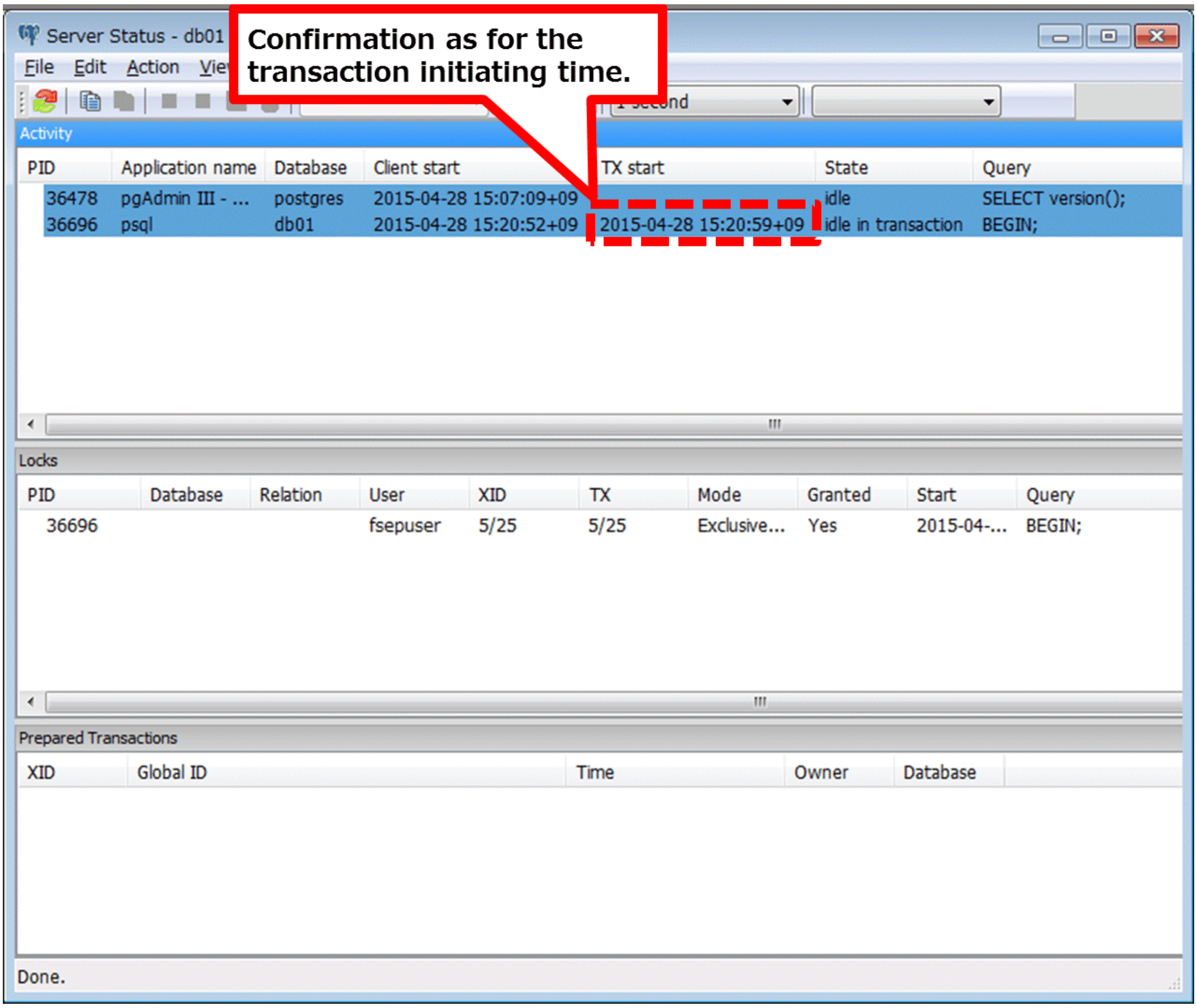
This section describes the procedure for monitoring connections using [Server Status] in pgAdmin.

1. From the [Tools] menu in pgAdmin, click [Server Status].



2. Identify client connections that have been in the waiting state for an extended period.

   From the transaction start time displayed under [TX Start], identify connections that have been in the waiting state for an extended period.



Confirmation as for the transaction initiating time.

PID	Application name	Database	Client start	TX start	State	Query
36478	pgAdmin III - ...	postgres	2015-04-28 15:07:09+09		idle	SELECT version();
36696	psql	db01	2015-04-28 15:20:52+09	2015-04-28 15:20:59+09	idle in transaction	BEGIN;

PID	Database	Relation	User	XID	TX	Mode	Granted	Start	Query
36696			fsepuser	5/25	5/25	Exclusive...	Yes	2015-04-...	BEGIN;

XID	Global ID	Time	Owner	Database

7.5 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 "7.2 Monitoring Disk Usage and Securing Free Space".

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

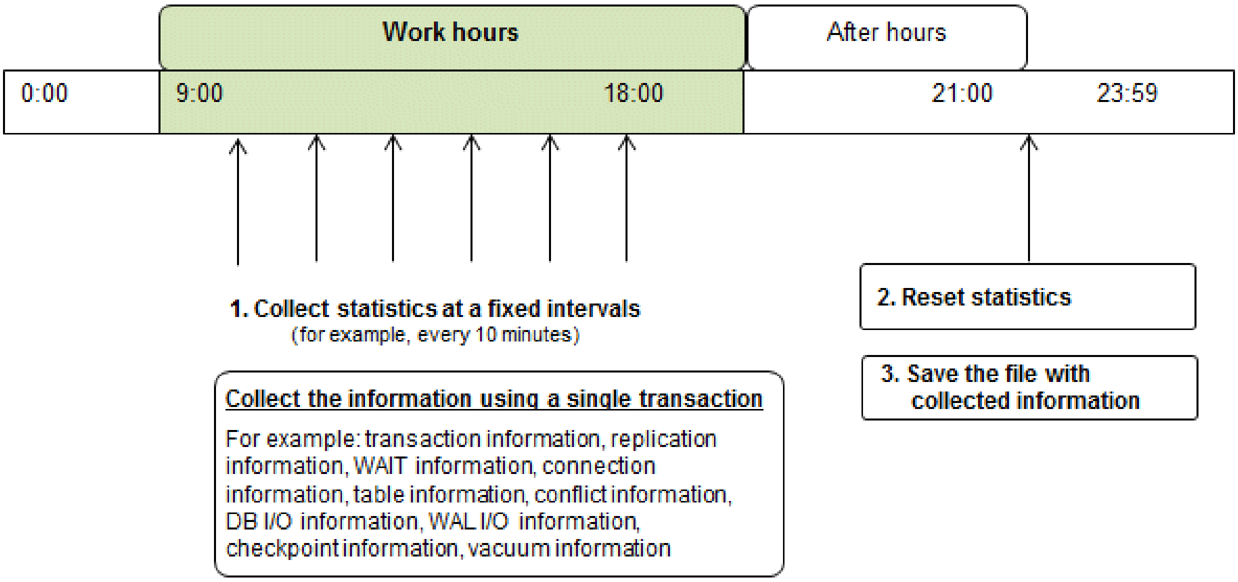
Refer to "Notes on Application Compatibility" in the Application Development Guide for information on maintaining application compatibility.

7.6 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 7.1 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 "7.6.1 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 "7.6.3 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

7.6.1 Information that can be Collected

Information that can be collected is categorized into the following two 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 7.1 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 "C.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 "C.3 pgx_stat_latch" for details.
pgx_stat_walwriter	Displays statistics related to WAL writing, in a single line. Refer to "C.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 "C.5 pgx_stat_sql" for details.

7.6.2 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
pgx_stat_sql	track_sql

Remarks: You cannot change the collection status for pgx_stat_walwriter.

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

7.6.3 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

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.

Chapter 8 Streaming Replication Using WebAdmin

This chapter describes how to create a streaming replication cluster using WebAdmin.

8.1 Streaming Replication

Streaming replication allows the creation of one or more standby instances, which connect to the master instances and replicate the data using WAL records. The standby instance can be used for read-only operations.

WebAdmin can be used to create a streaming replication cluster. WebAdmin allows the creation of a cluster in the following configurations:

- Master-Standby Configuration: This configuration creates a master and standby instance together.
- Standby Only Configuration: This configuration creates a standby instance from an already existing instance.

Point

- A standby instance can be created from a standalone instance, a master instance, or even from another standby instance.
- 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].

See

When a standby instance is created from an existing instance, it is necessary to set the values for the replication-related variables before a standby instance can be created. Refer to "Configuring Parameters" in the Installation and Setup Guide for Server for details.

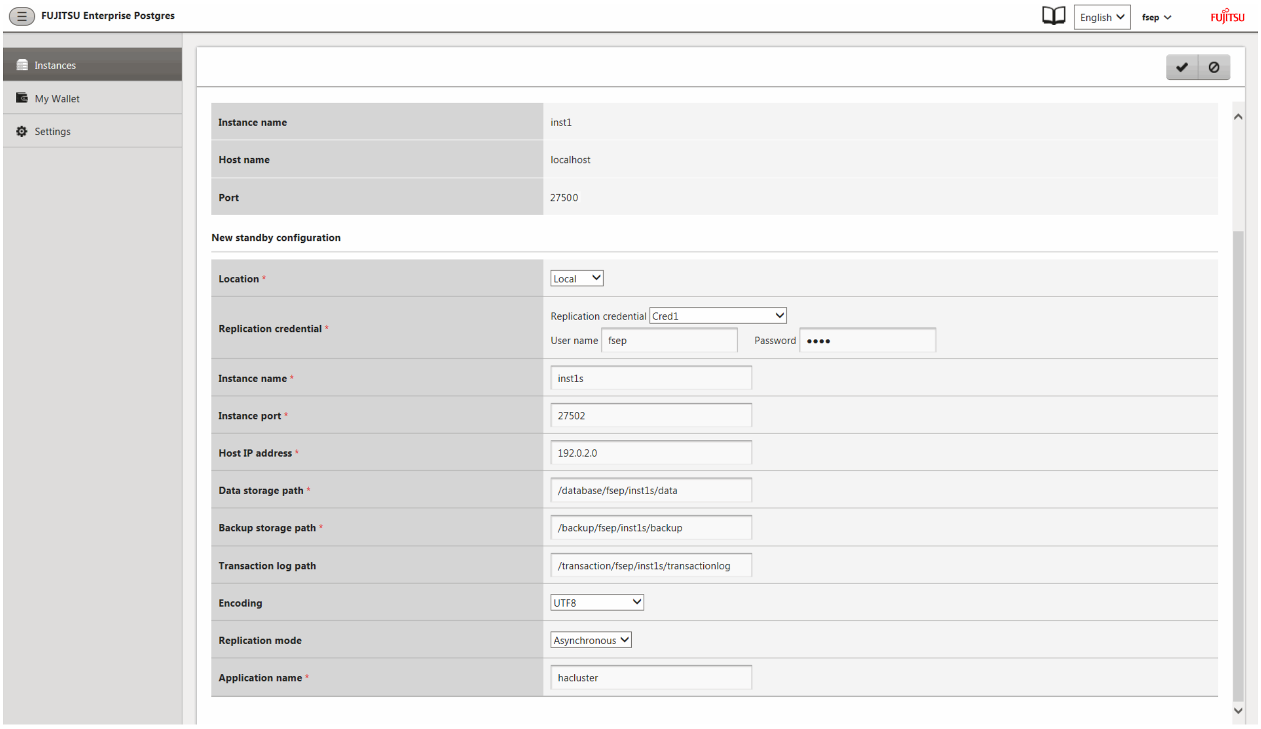
8.1.1 Creating a Standby Instance

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. In the example below, a standby instance is created from instance "inst1".

   The instance name, host address and port of the selected instance are already displayed for easy reference.



Instance name	inst1
Host name	localhost
Port	27500

New standby configuration

Location *	Local
Replication credential *	Replication credential Cred1 User name fsep Password ••••
Instance name *	inst1s
Instance port *	27502
Host IP address *	192.0.2.0
Data storage path *	/database/fsep/inst1s/data
Backup storage path *	/backup/fsep/inst1s/backup
Transaction log path	/transaction/fsep/inst1s/transactionlog
Encoding	UTF8
Replication mode	Asynchronous
Application name *	hacluster

   Enter the following items:

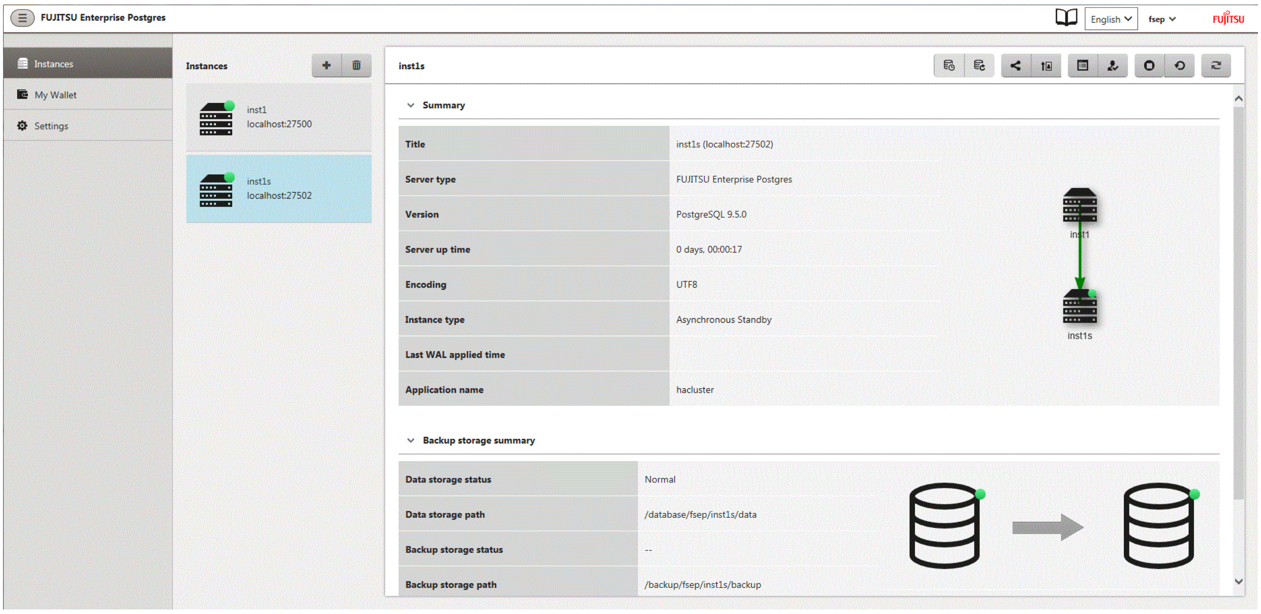
   - [Location]: Whether to create the instance in the server that the current user is logged in to, or in a remote server. The default is "Local", which will create the instance in the server machine where WebAdmin is currently running.
   - [Replication credential]: The user name and password required 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 F 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]: Whether the standby instance created should be in Asynchronous or Synchronous mode.
   - [Application name]: The reference name of the standby instance used to identify it to the master instance.

4. Click ✔ to create the standby instance.

5. Once the standby instance is created successfully, select "inst1s" in the [Instances] tab. The following page will be displayed:



Summary

Title	inst1s (localhost:27502)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	0 days, 00:00:17
Encoding	UTF8
Instance type	Asynchronous Standby
Last WAL applied time	
Application name	hacluster

Backup storage summary

Data storage status	Normal
Data storage path	/database/fsep/inst1s/data
Backup storage status	--
Backup storage path	/backup/fsep/inst1s/backup

Note

- Backups are not possible for standby instances in WebAdmin. As a result, and are disabled and no value is shown for [Backup storage status] and [Backup time].
- 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 (16491)
ERROR:  pgx_rcvall: backup of the database has not yet been performed, or an incorrect backup
storage directory was specified
```

8.1.2 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 .

8.1.3 Converting an Asynchronous Replication to Synchronous

Streaming replication between a master and standby instance can be configured to be in Asynchronous or Synchronous mode. This mode can be changed even after the standby instance was successfully created.

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.

Note

- 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.

- 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.

8.1.4 Converting a Synchronous Replication to Asynchronous

Streaming replication between a master and standby instance can be configured to be in Asynchronous or Synchronous Mode. This mode can be changed even after the standby instance was successfully created.

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.

Note

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.

Chapter 9 Installing and Operating the In-memory Feature

The in-memory feature enables fast aggregation using Vertical Columnar 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.

9.1 Installing Vertical Columnar Index (VCI)

This section describes the installation of VCI.

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

9.1.1 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 aggregation of sum and average

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

See

- Refer to "9.1.4 Data that can Use VCI" for information on VCI restrictions.
- Refer to "Scan Using a Vertical Columnar Index (VCI)" - "Operating Conditions" in the Application Development Guide for information on VCI operating conditions.

9.1.2 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.

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.

9.1.3 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

9.1.3.1 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	Literall '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
```

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.

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

9.1.3.2 Installing the Extensions

Execute the CREATE EXTENSION statement 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 "9.2.1 Commands that cannot be Used for VCI" for details.

9.1.3.3 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 "9.1.4.1 Relation Types" for details.

- Some data types cannot be specified on the column specification of CREATE INDEX. Refer to "9.1.4.2 Data Types" for details.

- Some operations cannot be performed for VCI. Refer to "9.2.1 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.

9.1.3.4 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)
```

9.1.4 Data that can Use VCI

This section describes on which relation types and for which data types VCIs can be created.

9.1.4.1 Relation Types

VCIs cannot be created on some relation types.

The ON clause of CREATE INDEX described in "9.1.3.3 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

9.1.4.2 Data Types

VCIs cannot be created for some data types.

The column specification of CREATE INDEX described in "9.1.3.3 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

Category	Data type	Supported by VCI?
	nchar	N
	nvarchar(n)	N
	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
Bit string	bit(n)	Y
	bit varying(n)	Y
Text search	tsvector	N
	tsquery	N
UUID	uuid	Y
XML	xml	N
JSON	json	N
	jbson	N
Range	int4range	N
	int8range	N
	numrange	N
	tsrange	N
	tstzrange	N
	daterange	N
Object identifier	oid	N
	regproc	N
	regprocedure	N
	regoper	N

Category	Data type	Supported by VCI?
	regoperator	N
	regclass	N
	regtype	N
	regconfig	N
	regdictionary	N
Array type	-	N
User-defined type (Basic type, enumerated type, composite type, and range type)	-	N

9.2 Operating VCI

This section describes how to operate VCI.

9.2.1 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. This operation is not required for VCI.
DROP EXTENSION	-	The VCI extension cannot be specified. (Restriction in this edition.)

- 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	

Command	Subcommand	Description
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}	
	SET WITH OIDS	If WITHOUT OIDS is specified for a table that contains a VCI, the SET WITH OIDS clause cannot be used. If the operation is required, delete the VCI using DROP INDEX, and re-create it using CREATE INDEX after completing the operation.
	SET WITHOUT OIDS	If WITH OIDS is specified for a table that contains a VCI, the SET WITHOUT OIDS clause cannot be used. If the operation is required, delete the VCI using DROP INDEX, and re-create it using CREATE INDEX after completing the operation.
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. This operation is not supported in VCI.
	CONCURRENTLY	
	[ASC | DESC]	
	[NULLS { FIRST | LAST }]	
	WITH	
	WHERE	
CREATE OPERATOR CLASS	-	A VCI cannot be specified. This operation is not supported in VCI.
CREATE OPERATOR FAMILY	-	

Command	Subcommand	Description
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.

9.2.2 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.4 VCI Data Load Control Function" for information on pgx_prewarm_vci.

Chapter 10 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 "10.1 Recovering from Disk Failure (Hardware)", and take actions accordingly.

- If the disk is not defective

  Refer to "10.14 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.

- 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

10.1 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:

- 10.1.1 Using WebAdmin

- 10.1.2 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.

10.1.1 Using WebAdmin

Recover the database cluster by following the appropriate recovery procedure below for the disk where the failure occurred.

Note

Recovery operation cannot be performed on an instance that is part of a streaming replication cluster in standby mode.

If disk failure occurs on a standby instance, it may be necessary to delete and re-create the instance.

Recovery operation can be performed on an instance that is part of a streaming replication cluster in "Master" mode. If a recovery operation is performed on a master instance, it will break the replication cluster and streaming replication will stop between the master instance and all its standby instances. In such an event, the standby instances can be promoted to standalone instances or can be deleted and re-created.

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

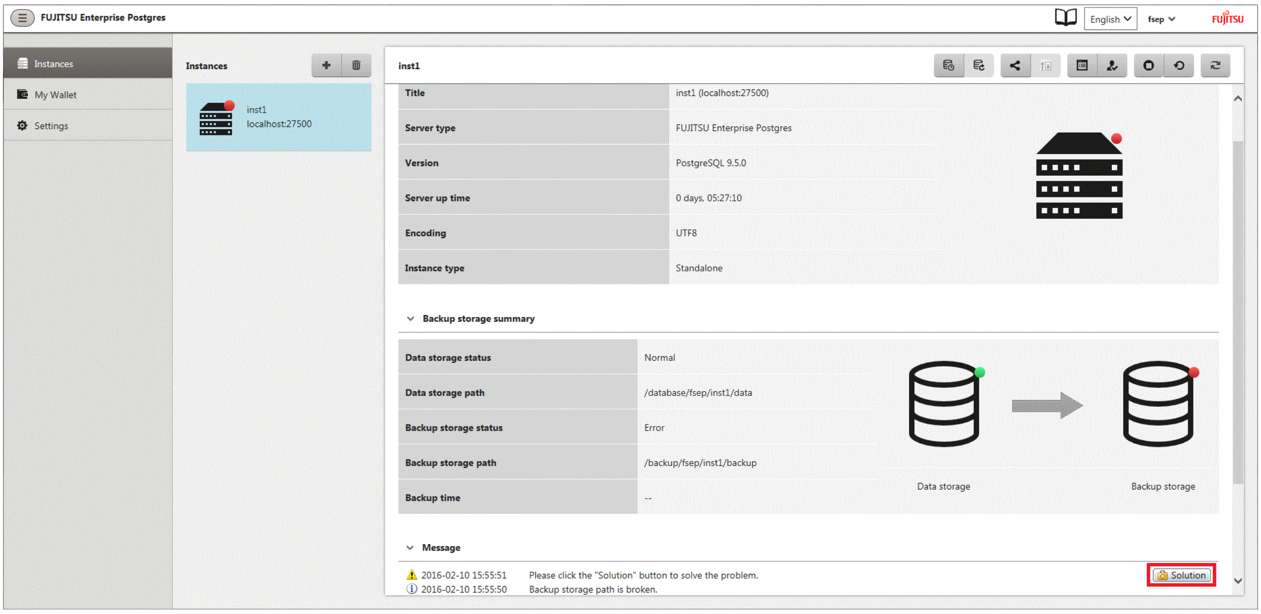
   Do the following if the data in the database has been encrypted:

   - 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.



Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	0 days, 05:27:10
Encoding	UTF8
Instance type	Standalone

Backup storage summary

Data storage status	Normal
Data storage path	/database/fsep/inst1/data
Backup storage status	Error
Backup storage path	/backup/fsep/inst1/backup
Backup time	--

Message

```
2016-02-10 15:55:51   Please click the "Solution" button to solve the problem.
2016-02-10 15:55:50   Backup storage path is broken.
```

7. Run recovery

   In the [Restore Instance] dialog box, click [Yes].

   Instance restore is performed. An instance is automatically started when recovery is successful.

   Note

   WebAdmin does not support recovery of hash index. If you are using a hash index, then after recovery, execute the REINDEX command to rebuild it. Use of hash indexes is not recommended.

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 "10.10.3 Other Errors" to perform recovery.

If failure occurred on the backup data storage disk

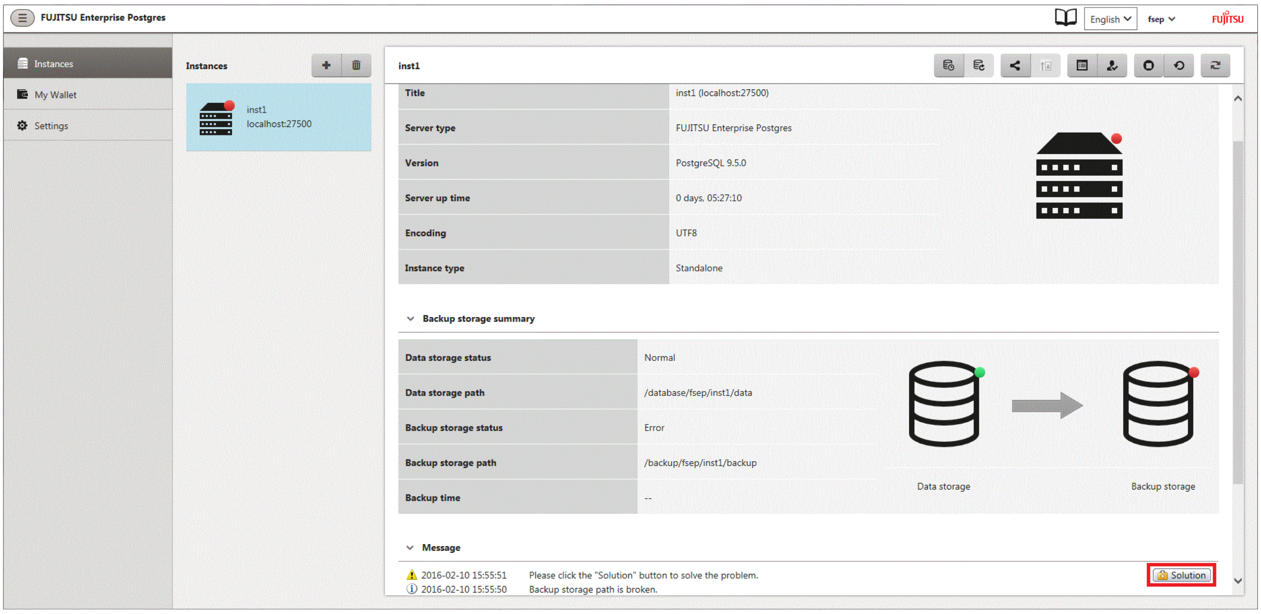
Follow the procedure below to recover the backup data storage disk.

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 [Instance] tab, click [Solution] for the error message.



Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	0 days, 05:27:10
Encoding	UTF8
Instance type	Standalone

Data storage status	Normal
Data storage path	/database/fsep/inst1/data
Backup storage status	Error
Backup storage path	/backup/fsep/inst1/backup
Backup time	--

```
2016-02-10 15:55:51   Please click the "Solution" button to solve the problem.
2016-02-10 15:55:50   Backup storage path is broken.
```

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.

10.1.2 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 "10.11 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
```

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 "10.11 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.

10.2 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:

- 10.2.1 Using WebAdmin
- 10.2.2 Using the pgx_rcvall Command

Note

- 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.

10.2.1 Using WebAdmin

If using WebAdmin, recover the data to the point immediately prior to data corruption by using the backup data.

Refer to "10.1.1 Using WebAdmin" for details.

10.2.2 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 "10.11 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 pg_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
   2015-05-20 10:00:00     COMPLETE        /backup/inst1/2015-05-20_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 pg_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, "May 20, 2015 10:00:00" is specified as the recovery time.

```
> pgx_rcvall -D /database/inst1 -B /backup/inst1 -e '2015-05-20 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
```

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.

   Note

   The pgx_rcvall command cannot accurately recover a hash index. If you are using a hash index, wait for the instance to start and then execute the REINDEX command for the appropriate index.

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.

10.3 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:

- 10.3.1 Using WebAdmin
- 10.3.2 Using the pgx_rcvall Command

Note

- 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.

10.3.1 Using WebAdmin

You can use WebAdmin to recover data to a backup point.

Note

Recovery operation cannot be performed on an instance that is part of a streaming replication cluster in standby mode.

If disk failure occurs on a standby instance, it may be necessary to delete and re-create the instance.

Recovery operation can be performed on an instance that is part of a streaming replication cluster in "Master" mode. If a recovery operation is performed on a master instance, it will break the replication cluster and streaming replication will stop between the master instance and all its 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.

   Note

   WebAdmin cannot accurately recover a hash index. If you are using a hash index, then after recovery, execute the REINDEX command for the appropriate index.

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.

10.3.2 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 "10.11 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 pg_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
2015-05-01 10:00:00     COMPLETE        /backup/inst1/2015-05-01_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 pg_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_20150503_1".

```
> pgx_rcvall -D /database/inst1 -B /backup/inst1 -n batch_20150503_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
```

6. Start the instance

   Start the instance.

   Refer to "2.1.2 Using Server Commands" for information on how to start an instance.

Note

The pgx_rcvall command cannot accurately recover a hash index. If you are using a hash index, wait for the instance to start and then execute the REINDEX command for the appropriate index.

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.

10.4 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 "10.4.1 When using the view (pg_stat_activity)")
- ps command (refer to "10.4.2 Using the ps Command")
- pgAdmin (refer to "10.4.3 Using pgAdmin")

Use the system management function (pg_terminate_backend) to disconnect connections.

10.4.1 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 (9.5.2)
Type "help" for help.
```

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_hostname,pg_terminate_backend(pid) from
pg_stat_activity where 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.

10.4.2 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 "<*xy*>" indicates the product version and level.

```
> ps axwfo user,pid,ppid,tty,command | grep postgres
fsepuser 19174 18027 pts/1                 \_ grep postgres
fsepuser 20517     1 ?        /opt/fsepv<xy>server64/bin/postgres -D /disk01/data
fsepuser 20518 20517 ?         \_ postgres: logger process
fsepuser 20520 20517 ?         \_ postgres: checkpointer process
fsepuser 20521 20517 ?         \_ postgres: writer process
fsepuser 20522 20517 ?         \_ postgres: wal writer process
fsepuser 20523 20517 ?         \_ postgres: autovacuum launcher process
fsepuser 20524 20517 ?         \_ postgres: archiver process
fsepuser 20525 20517 ?         \_ postgres: stats collector process
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.

   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)
```

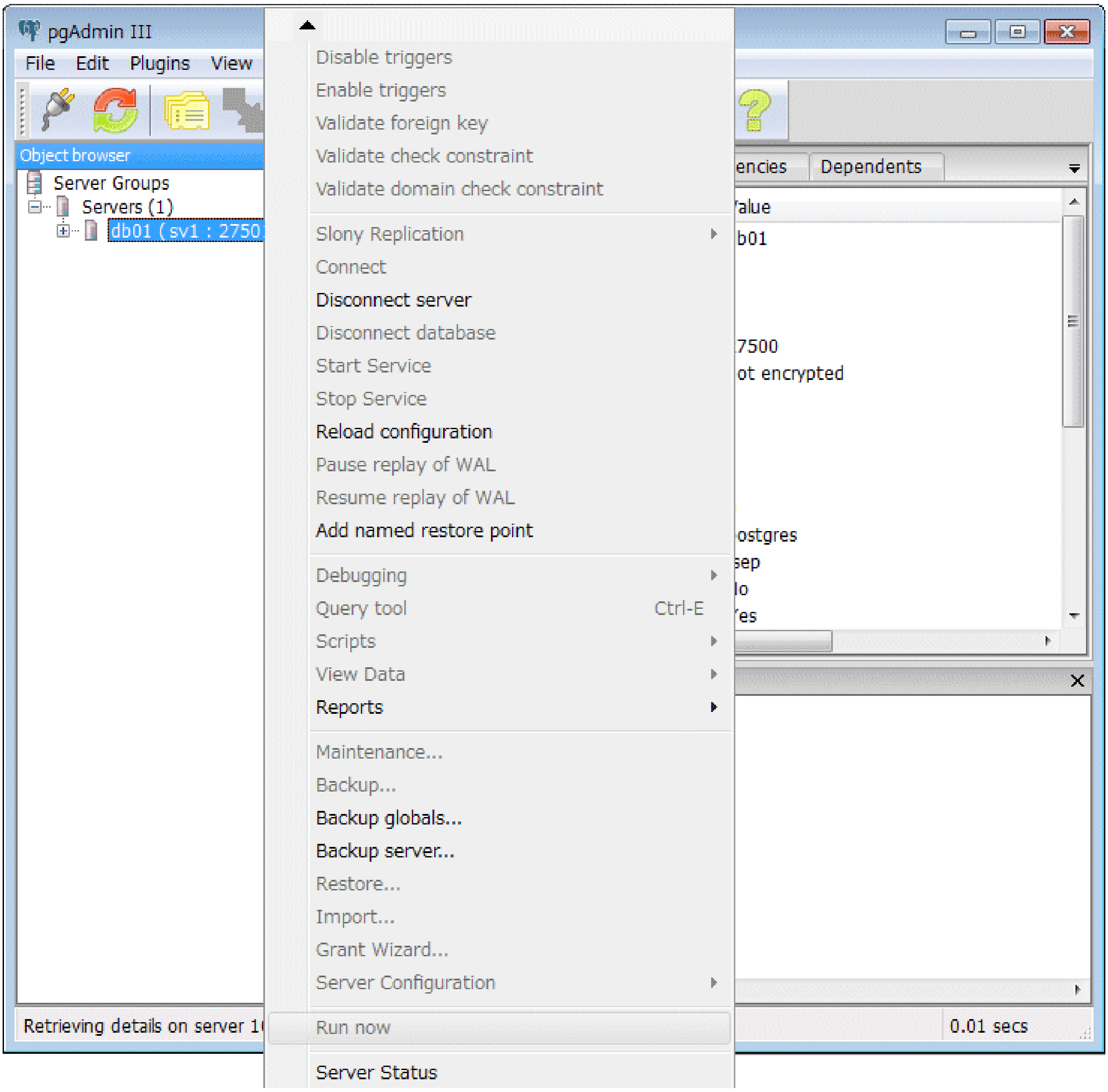
See

- 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.

10.4.3 Using pgAdmin

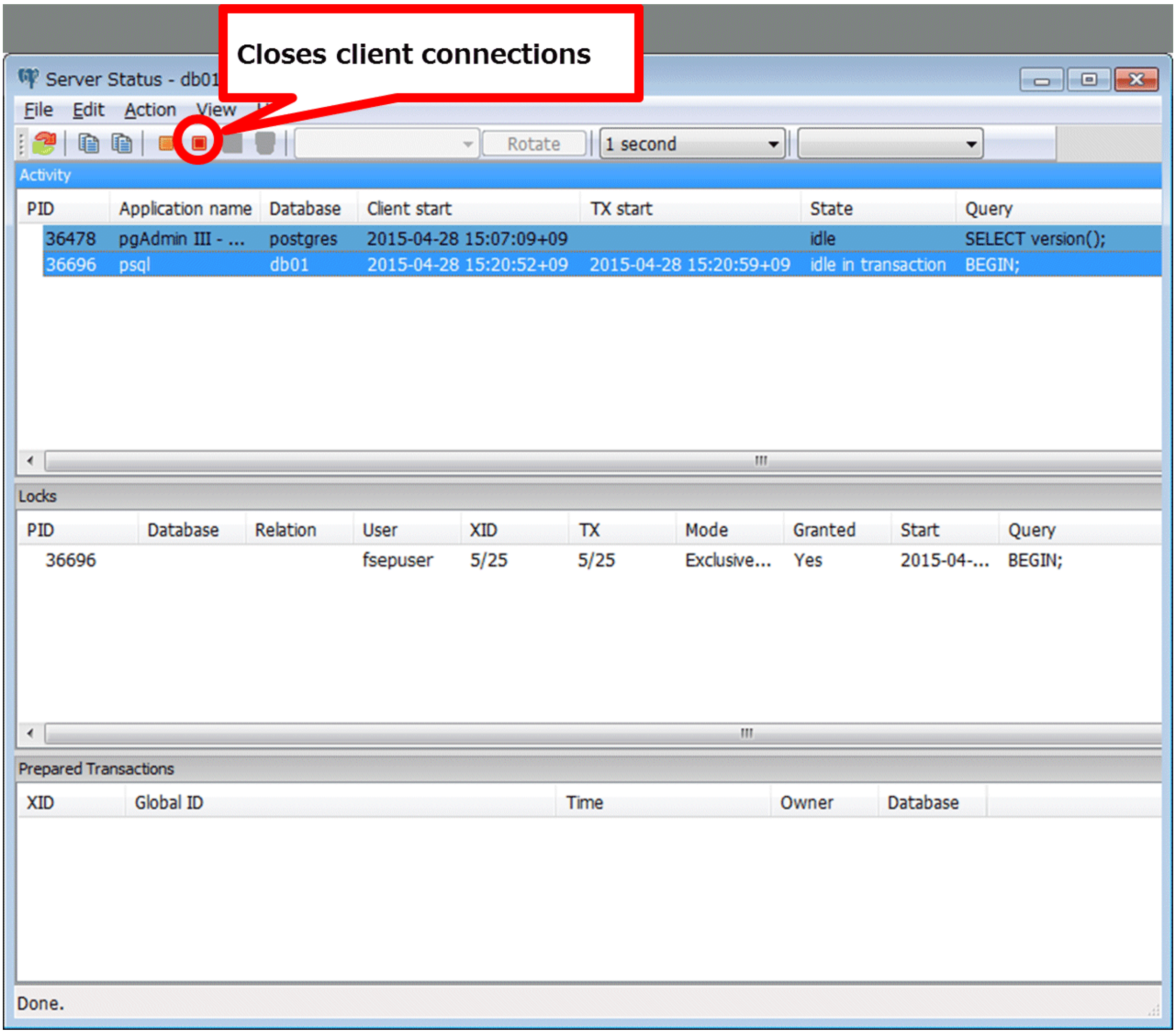
If using pgAdmin, follow the procedure below to close connections.

1. From the [Tools] menu in pgAdmin, click [Server Status].



2. Close client connections that have been in the waiting state for an extended period.

   From the transaction start time displayed under [TX Start], select connections that have been in the waiting state for an extended period. Then click the red square button to close the connections.



Closes client connections

Activity

PID	Application name	Database	Client start	TX start	State	Query
36478	pgAdmin III - ...	postgres	2015-04-28 15:07:09+09		idle	SELECT version();
36696	psql	db01	2015-04-28 15:20:52+09	2015-04-28 15:20:59+09	idle in transaction	BEGIN;

Locks

PID	Database	Relation	User	XID	TX	Mode	Granted	Start	Query
36696			fsepuser	5/25	5/25	Exclusive...	Yes	2015-04-...	BEGIN;

Prepared Transactions

XID	Global ID	Time	Owner	Database

Done.

10.5 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 operations. Also, refer to the system 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 operations.

- 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.

10.6 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:

- 10.6.1 Using a Tablespace
- 10.6.2 Replacing the Disk with a Larger Capacity Disk

10.6.1 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.

10.6.2 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:

- 10.6.2.1 Using WebAdmin
- 10.6.2.2 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.

Note

- Before replacing the disk, stop applications and instances that are using the database.
- 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.

10.6.2.1 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 "10.1.1 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.

10.6.2.2 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 "10.11 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
   ```

   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.

   Note

   The pgx_rcvall command cannot accurately recover a hash index. If you are using a hash index, wait for the pgx_rcvall command to end and then execute the REINDEX command for the appropriate index.

10. Resume applications

    Resume applications that are using the database.

11. Restore files

    Restore the files backed up in step 1.

10.7 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:

- 10.7.1 Temporarily Saving Backup Data

- 10.7.2 Replacing the Disk with a Larger Capacity Disk

10.7.1 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:

- 10.7.1.1 Using WebAdmin
- 10.7.1.2 Using Server Commands

10.7.1.1 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
   ```

10.7.1.2 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

No	Step	Instance stopped	
		No	Yes
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 "10.11 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.

10.7.2 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:

- 10.7.2.1 Using WebAdmin
- 10.7.2.2 Using Server Commands

Note

Before replacing the disk, stop applications that are using the database.

10.7.2.1 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. Replace with a larger capacity disk

   Replace the disk. Then, recover the volume configuration information.

4. 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 "10.1.1 Using WebAdmin".

5. Restore files

   Restore the files backed up in step 1.

6. 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
   ```

10.7.2.2 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

No	Step	Instance stopped	
		No	Yes
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 "10.11 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.

10.8 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.

10.8.1 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:

- 10.8.1.1 Using WebAdmin
- 10.8.1.2 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.

Note

- Before replacing the disk, stop applications that are using the database.
- 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.

10.8.1.1 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 "10.1.1 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.

10.8.1.2 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 "10.11 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
```

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.

Note

The pgx_rcvall command cannot accurately recover a hash index. If you are using a hash index, wait for the instance to start and then execute the REINDEX command for the appropriate index.

10. Resume applications

    Resume applications that are using the database.

11. Restore files

    Restore the files backed up in step 1.

10.9 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.

10.10 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.

10.10.1 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

10.10.2 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 "10.14.2 Errors Caused by Power Failure or Mounting Issues", and take actions accordingly.

10.10.3 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:

- 10.10.3.1 Using WebAdmin
- 10.10.3.2 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.

10.10.3.1 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 "10.2.1 Using WebAdmin" for details.

10.10.3.2 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 "10.2.2 Using the pgx_rcvall Command" for details.

10.11 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:

- 10.11.1 Using WebAdmin
- 10.11.2 Using Server Commands

10.11.1 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.

10.11.2 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.

10.11.2.1 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 "10.11.2.2 Stopping the Instance Using the Immediate Mode" or "10.11.2.3 Forcibly Stopping the Server Process".

Example

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

10.11.2.2 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 "10.11.2.3 Forcibly Stopping the Server Process".

Example

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

10.11.2.3 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 "<*xy*>" indicates the product version and level.

   ```
   > ps axwfo user,pid,ppid,tty,command | grep postgres
   fsepuser 19174 18027 pts/1                 \_ grep postgres
   fsepuser 20517     1 ?        /opt/fsepv<xy>server64/bin/postgres -D /database/inst1
   fsepuser 20518 20517 ?         \_ postgres: logger process
   fsepuser 20520 20517 ?         \_ postgres: checkpointer process
   fsepuser 20521 20517 ?         \_ postgres: writer process
   fsepuser 20522 20517 ?         \_ postgres: wal writer process
   fsepuser 20523 20517 ?         \_ postgres: autovacuum launcher process
   fsepuser 20524 20517 ?         \_ postgres: archiver process
   fsepuser 20525 20517 ?         \_ postgres: stats collector process
   ```

   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
```

10.12 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.

Refer to "Deleting Instances" in the Installation and Setup Guide for details on how to delete a instance.

10.13 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 (such as .NET TransactionScope), 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 | 2015-05-06 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
```

10.14 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.

- 10.14.1 Network Error with an External Disk
- 10.14.2 Errors Caused by Power Failure or Mounting Issues

10.14.1 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.

10.14.2 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.

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.

- 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.

- 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 parallel scan

- enable_parallelagg (boolean)

  This parameter enables or disables the query planner's use of parallel aggregation plan. If set to "off", parallel aggregation plan will not be used. Normally, use the default value when using the parallel scan feature.

    - Valid values: on or off, 1 or 0

    - Default value: on

- enable_parallelscan (boolean)

  This parameter enables or disables the query planner's use of parallel scan plan. If set to "off", parallel scan plan will not be used. Normally, use the default value when using the parallel scan feature.

    - Valid values: on or off, 1 or 0

    - Default value: on

- max_parallel_degree (4-byte signed integer)

  This parameter specifies the maximum number of parallel processes (background processes) to be used per SQL statement. If set to "0", no parallel processes will be generated, and therefore execution plans will not be able to use parallel scan. When using the parallel scan feature, specify this parameter.

    - Minimum value: 0

    - Maximum value: 8388607

    - Default value: 0

- parallel_scan_pages_threshold (4-byte signed integer)

  If the target table size is equal to the specified threshold (number of pages) or more, the query planner considers the parallel scan plan as an option. Normally, use the default value when using the parallel scan feature.

    - Minimum value: 1

    - Maximum value: 4-byte signed integer

    - Default value: 1000

- parallel_setup_cost (double precision floating point)

  This parameter sets the estimated cost of the planner for starting parallel processes. This value is used to calculate the cost for the parallel scan plan.

    - Minimum value: 0

    - Maximum value: Maximum value of double precision floating point

    - Default value: 1000.0

- parallel_tuple_cost (double precision floating point)

  This parameter sets the estimated cost of the planner for transferring rows from parallel processes to backend processes. This value is used to calculate the cost for the parallel scan plan.

  - Minimum value: 0
  - Maximum value: Maximum value of double precision floating point
  - Default value: 0.1

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.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.

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.

- 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.

See

Refer to "Server Configuration" under "Server Administration" in the PostgreSQL Documentation for information on other postgresql.conf parameters.

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*)	void	Opens the keystore
pgx_set_master_key(*passphrase*)	void	Sets the master encryption key
pgx_set_keystore_passphrase(*oldPassphrase*, *newPassphrase*)	void	Changes the keystore passphrase

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.

Only superusers can execute this function. Also, this function cannot be executed within a transaction block.

The pgx_set_master_key function generates a master encryption key and stores it in the 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.

The pgx_set_keystore_passphrase function changes the 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 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.3.1 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    := 'columnName'
[, function_type    := 'FULL'] |
[, function_type    := 'PARTIAL', partialOpt] |
[, function_type    := 'REGEXP', regexpOpt]
)


partialOpt:
function_parameters    := 'maskingFormat'


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    := 'columnName'
)
```

- Change the masking condition

```
pgx_alter_confidential_policy(
commonArg,
action        := 'MODIFY_EXPRESSION',
expression    := 'expression'
)
```

- Change the content of a masking policy set for a masking target

```
pgx_alter_confidential_policy(
commonArg,
action        := 'MODIFY_COLUMN',
column_name    := 'columnName'
[, function_type    := 'FULL'] |
[, function_type    := 'PARTIAL', partialOpt] |
[, function_type    := 'REGEXP', regexpOpt]
)


partialOpt:
function_parameters    := 'maskingFormat'


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    := 'columnName',
column_description    := 'columnDesc'
)
```

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

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		
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.
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.3.2 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        := 'expression'
[, enable            := 'policyStatus']
[, policy_description    := 'policyDesc']
[, column_name        := 'columnName'
     [, function_type    := 'FULL'] |
     [, function_type    := 'PARTIAL', partialOpt] |
     [, function_type    := 'REGEXP', regexpOpt]
     [, column_description    := 'columnDesc']
])


partialOpt:
function_parameters    := 'maskingFormat'


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

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
	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
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.3.3 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.3.4 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	Mandatory

Argument	Data type	Description	Default value
		- 'f': Disabled	

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.3.5 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.4 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.

Appendix C 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.

C.1 pgx_tablespaces

The pgx_tablespaces catalog provides information related to the encryption of tablespaces.

Name	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

C.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 C.1 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

C.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 C.2 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

C.4 pgx_stat_walwriter

The pgx_stat_walwriter view display statistics related to WAL writing, in a single line.

Table C.3 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

C.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 C.4 pgx_stat_sql view

Column	Type	Description
selects	bigint	Number of SELECT statements executed
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

Appendix D Tables Used by Data Masking

This appendix explains tables used by the data masking feature.

D.1 pgx_confidential_columns

This table provides information on masking target for which masking policies are set.

Name	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)
```

D.2 pgx_confidential_policies

This table provides information on masking policies.

Name	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
expression	varchar(1024)	Masking condition
enable	boolean	Masking policy status

Name	Type	Description
		- '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)
```

D.3 pgx_confidential_values

This table provides information on replacement characters when full masking is specified for masking type.

Name	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 E Activating 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 activate the Web server feature of WebAdmin.

This appendix describes how to activate and stop the Web server feature of WebAdmin.

Note that "<*xy*>" in paths indicates the product version and level.

E.1 Activating the Web Server Feature of WebAdmin

Follow the procedure below to activate the Web server feature of WebAdmin.

1. Change to superuser

   Acquire superuser privileges on the system.

   Example

   ```
   $ su -
   Password:******
   ```

2. Activate the Web server feature of WebAdmin

   Execute the WebAdminStart command to activate the Web server feature of WebAdmin.

   Example

   If FUJITSU Enterprise Postgres is installed in /opt/fsepv<*xy*>server64:

```
# cd /opt/fsepv<xy>server64/gui/sbin
# ./WebAdminStart
```

E.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 FUJITSU Enterprise Postgres is installed in /opt/fsepv<*xy*>server64:

```
# cd /opt/fsepv<xy>server64/gui/sbin
# ./WebAdminStop
```

Appendix F 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.

Note

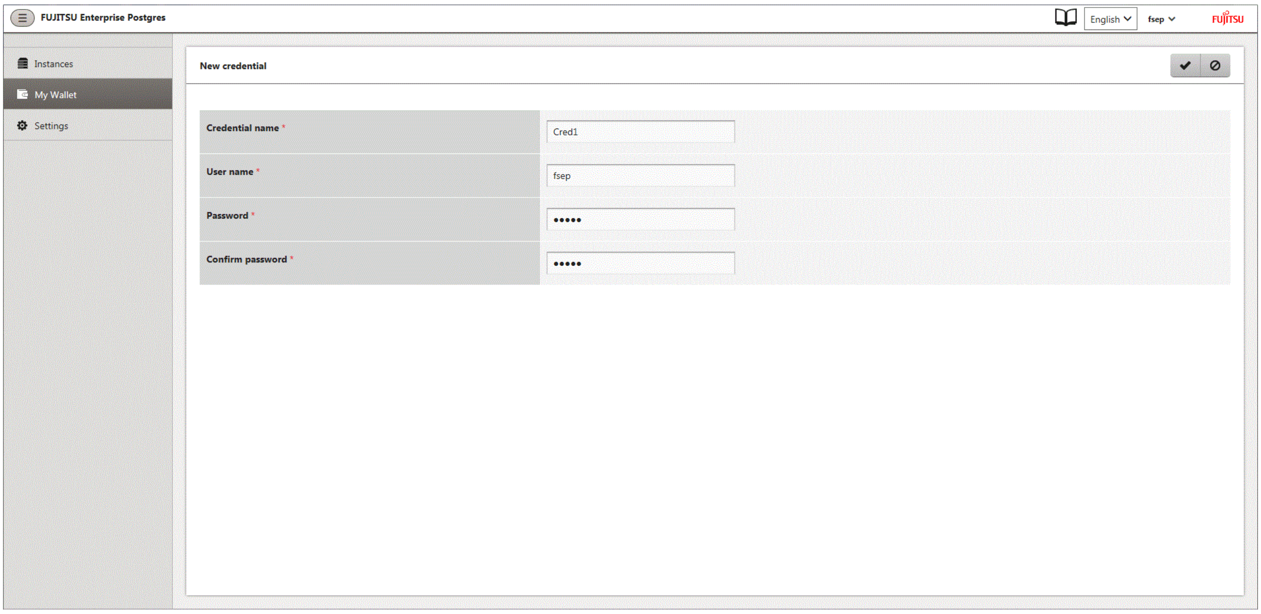
It is not mandatory to create a credential in the Wallet. It is possible to create a remote instance or a standby instance without creating any credential in the Wallet.

If no credential is created beforehand, a user name and password can be entered in the instance creation page. When creating a "Remote" instance, if operating system credentials are entered without using a credential stored in the Wallet, WebAdmin automatically creates a credential with the given user name and password, and stores it in the user’s wallet for future use.

F.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 *	Cred1
User name *	fsep
Password *	•••••
Confirm password *	•••••

Enter the following items:

- [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

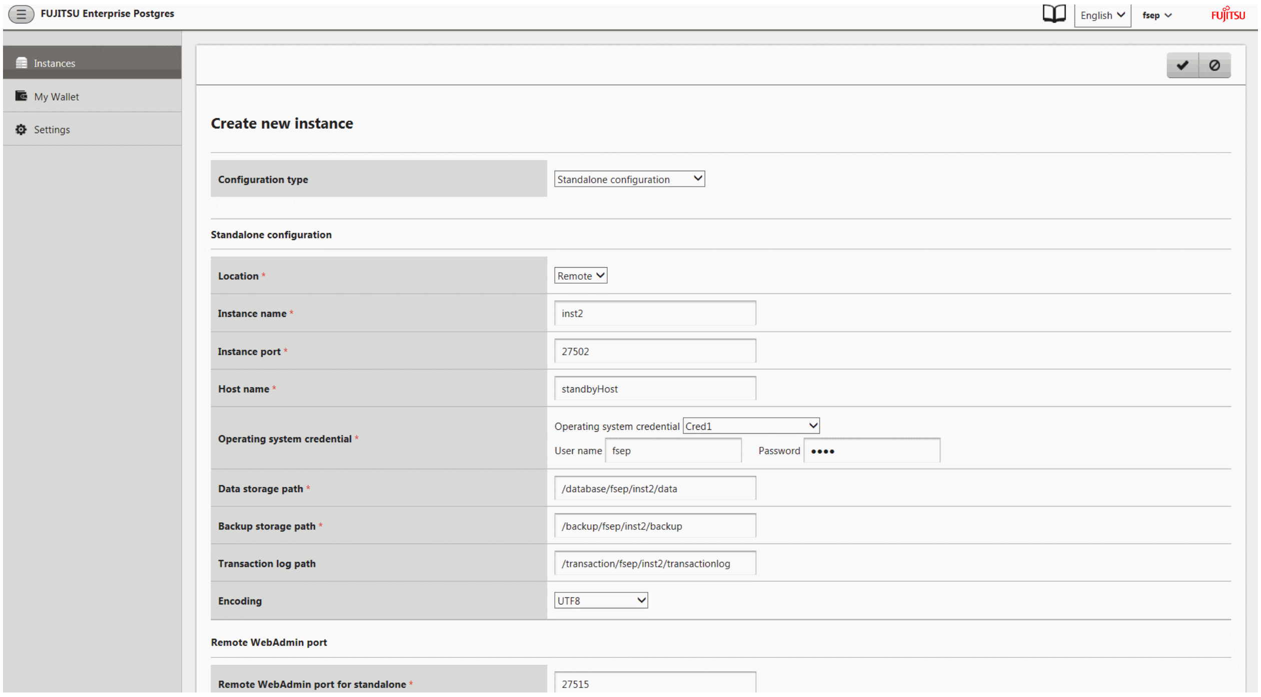
- [Confirm password]: Reenter the password.

3. Click ✓ to store the credential.

F.2 Using a Credential

Once a credential is created in the Wallet, it can be used during remote instance creation or standby instance creation.

The following page uses the credential that was created in the previous section.



Create new instance

Configuration type	Standalone configuration

Standalone configuration

Location *	Remote
Instance name *	inst2
Instance port *	27502
Host name *	standbyHost
Operating system credential *	Operating system credential Cred1 User name fsep Password ••••
Data storage path *	/database/fsep/inst2/data
Backup storage path *	/backup/fsep/inst2/backup
Transaction log path	/transaction/fsep/inst2/transactionlog
Encoding	UTF8

Remote WebAdmin port

Remote WebAdmin port for standalone *	27515

When "Cred1" is selected in [Operating system credential], the user name and password are automatically populated from the credential.

Appendix G 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 FJQSS (Information Collection Tool) to collect data for initial investigation.

See

Refer to the FJQSS manual for information on how to use FJQSS.

Note

When using FJQSS to collect data for initial investigation, you must set the following environment variables:

- Environment variables required for using FUJITSU Enterprise Postgres

  Refer to "Setup" in the Install and Setup Guide for Server for details.

- PGDATA

  Set the data storage destination.

- PGPORT

  Set the instance port number. This does not need to be set if the default port number (27500) has not been changed.

- PGUSER

  Set the database superuser.
  Set the database superuser so that client authentication is possible.
  FJQSS establishes a TCP/IP connection with the template1 database and collects data from the database.

- FSEP_HOME

  Set the FUJITSU Enterprise Postgres installation directory.

In addition, when using database multiplexing, set the following environment variables:

- MCCONTROLDIR

  Refer to "Mirroring Controller Resources" in the Cluster Operation Guide for information on the Mirroring Controller management directory.

Index

FUJITSU Enterprise Postgres 9.5

Operation Guide

Windows

B1WS-1277-02ENZ0(00)
July 2016

Preface

Purpose of this document

The FUJITSU Enterprise Postgres database system extends the PostgreSQL features and runs on the Windows 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
- Windows

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 Data Masking

Describes the data masking feature.

Chapter 7 Periodic Operations

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

Chapter 8 Streaming Replication Using WebAdmin

Describes how to create a streaming replication cluster using WebAdmin.

Chapter 9 Installing and Operating the In-memory Feature

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

Chapter 10 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 Views

Describes how to use the system view in FUJITSU Enterprise Postgres.

Appendix D Tables Used by Data Masking

Describes the tables used by the data masking feature.

Appendix E Activating and Stopping the Web Server Feature of WebAdmin

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

Appendix F WebAdmin Wallet

Describes how to use the Wallet feature of WebAdmin.

Appendix G 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

```
Second edition: July 2016
First edition: April 2016
```

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 switchover or failover operation using database multiplexing, refer to "Database Multiplexing Mode" in the Cluster Operation Guide.

Operation management using GUI tools

This involves managing operations using the WebAdmin and pgAdmin GUI tools.

- 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.

- Management using pgAdmin

  When developing applications and maintaining the database, you can use pgAdmin to perform simple operations on database objects, such as:

  - Rebuild indexes and update statistics
  - Create, delete, and update database objects

  In addition, from pgAdmin of FUJITSU Enterprise Postgres, you can use the expanded features provided by FUJITSU Enterprise Postgres on the PostgreSQL SQL commands.

  See

  Refer to pgAdmin Help for information on the expanded features of pgAdmin provided by FUJITSU Enterprise Postgres.

Operation management using commands

You can use commands for configuring and operating the database and managing operations. However, note that if you start managing operations using commands, you cannot switch to WebAdmin-based operation management.

Note

You cannot combine WebAdmin and server commands to perform the following operations:

- Use WebAdmin to operate an instance created using the initdb command
- 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.

- You can perform backup and restoration in pgAdmin, but the backup data obtained with WebAdmin and pgx_dmpall is not compatible with the backup data obtained with pgAdmin.
- Refer to pgAdmin Help for other notes on pgAdmin.

Features used in each phase

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

Operation		GUI-based operation	Command-based operation
Setup	Creating an instance	WebAdmin	initdb command
	Creating a standby instance	WebAdmin WebAdmin performs a base backup of the source instance and creates a standby instance.	pg_basebackup command
	Modifying the configuration file	WebAdmin	Directly edit the configuration file
Instance start		WebAdmin	OS-provided net command or sc command
Database creation		pgAdmin	Specify using the DDL statement, and define using psql and applications
Database backup		WebAdmin pgx_dmpall command	pgx_dmpall command
Monitoring	Database failure	WebAdmin(*1)	Messages output to the event log (*1)
	Disk space	WebAdmin (*1) (*2)	OS-provided fsutil command (check available capacity) and dir command (check used capacity)
	Connection status	pgAdmin	psql command (*3)
Database recovery		WebAdmin	pgx_rcvall command

*1: Operations can be monitored using operation management middleware (such as Systemwalker Centric Manager).

*2: A warning is displayed when disk usage reaches 80%.

*3: This command searches for pg_stat_activity in the standard statistics views and monitors the state.

1.2 Activating WebAdmin

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

1.2.1 Logging in to WebAdmin

This section describes how to log in to WebAdmin.

User environment

One of the following browsers is required for using WebAdmin:

- Internet Explorer 8.0 or later

Activation URL for WebAdmin

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

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

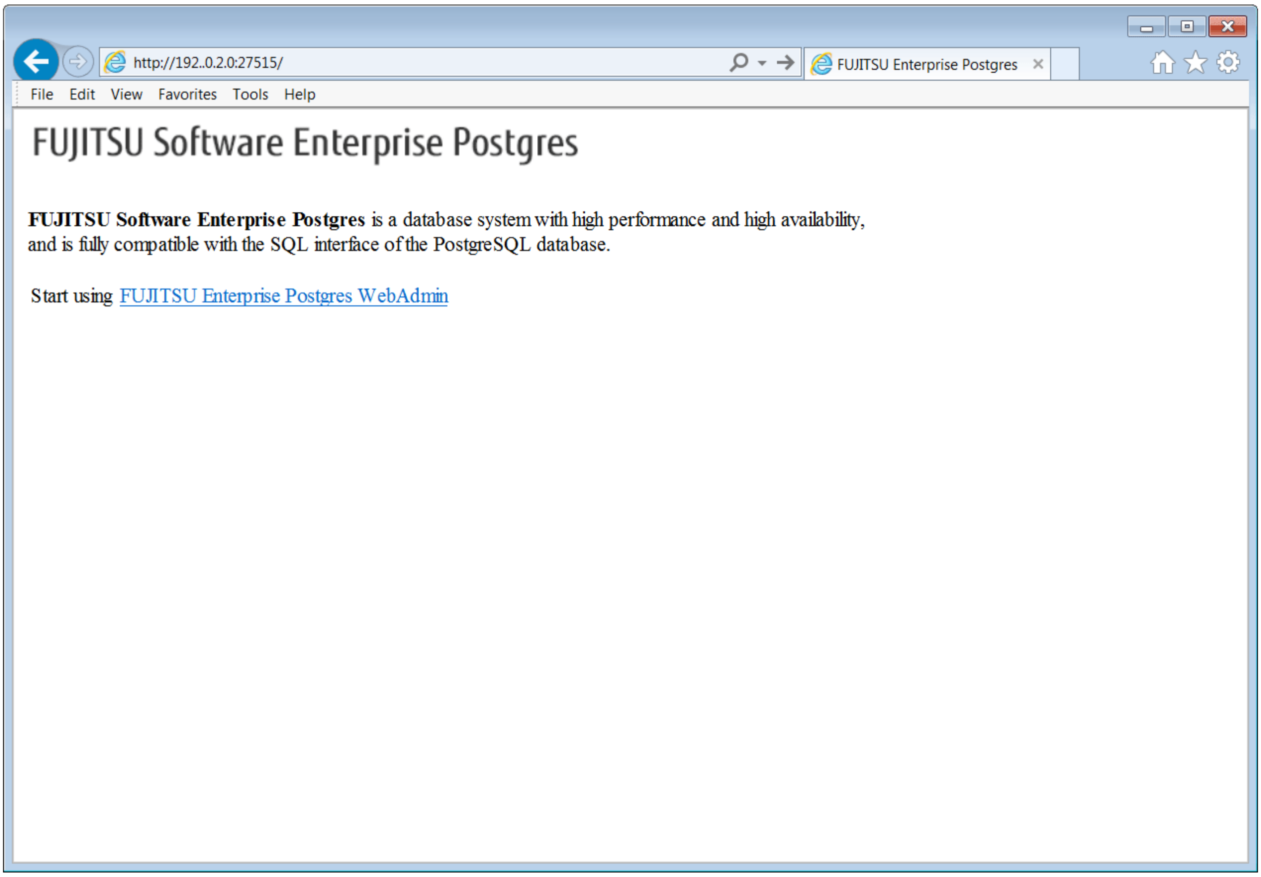
- *hostNameOrIpAddress*: The host name or IP address of the server where FUJITSU Enterprise Postgres is installed.
- *portNumber*: The port number of WebAdmin. The default port number is 27515.

Example

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

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

The activation URL window shown below is displayed.


http://192..0.2.0:27515/
FUJITSU Enterprise Postgres
File Edit View Favorites Tools Help

FUJITSU Software Enterprise Postgres

FUJITSU Software Enterprise Postgres is a database system with high performance and high availability, and is fully compatible with the SQL interface of the PostgreSQL database.

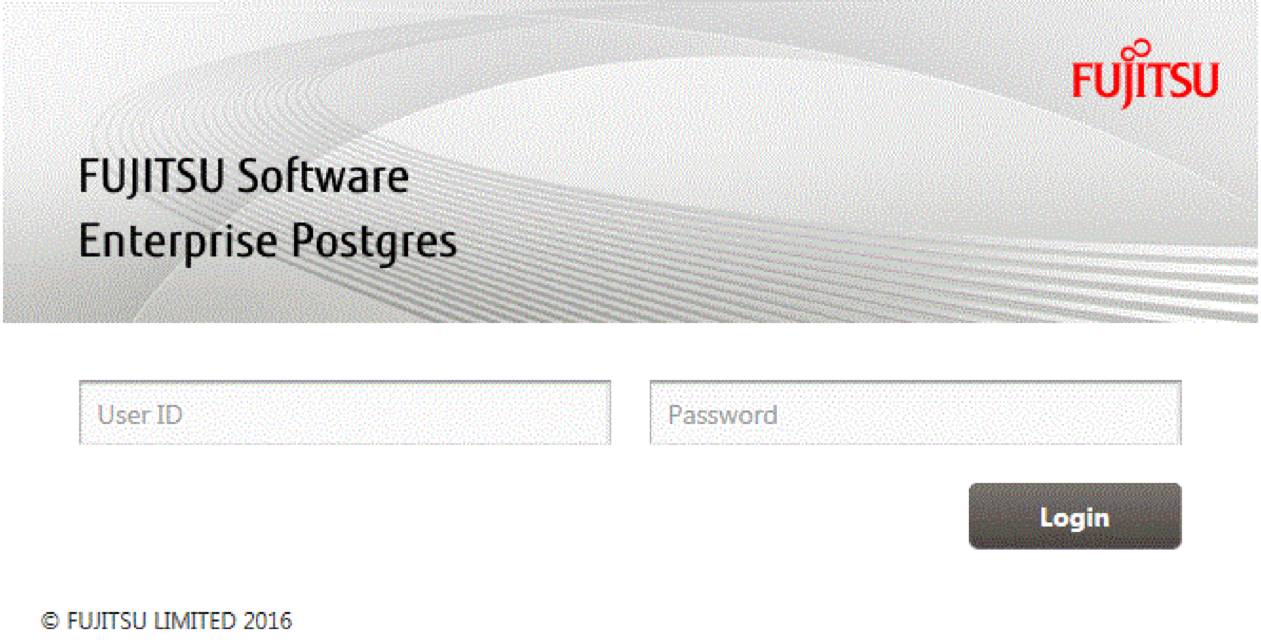
Start using FUJITSU Enterprise Postgres WebAdmin

Point

- You must activate the Web server feature of WebAdmin before using WebAdmin.
- Refer to "Appendix E Activating and Stopping the Web Server Feature of WebAdmin" for information on how to activate the Web server feature of WebAdmin.

Log in to WebAdmin

Click [FUJITSU Enterprise Postgres WebAdmin] in the activation URL window to activate WebAdmin and display the [Log in] window.



To log in, specify the following values:

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

Point

Use the OS user account as the user ID of the instance administrator. Refer to "Creating an Instance Administrator" in the Installation and Setup Guide for Server for details.

1.3 Starting pgAdmin

This section describes how to start pgAdmin, how to add an instance required for managing a database, and how to connect to and disconnect from the instance.

You can use pgAdmin on the Windows client.

1.3.1 Starting pgAdmin

This section explains how to start pgAdmin if you are using it from the product "FUJITSU Enterprise Postgres Client (*AA*bit) *x.y* SP*z*" (where *AA* is "32" or "64", *x.y* and *z* are the version numbers (*x.y* SP*z*)).

Windows(R) 8 or Windows Server(R) 2012

From the [Start] screen, start [pgAdmin III (*AA*bit) (*x.y* SP*z*)].

Windows(R) 8.1 or Windows Server(R) 2012 R2

From the [Apps] view, start [pgAdmin III (*AA*bit) (*x.y* SP*z*)].

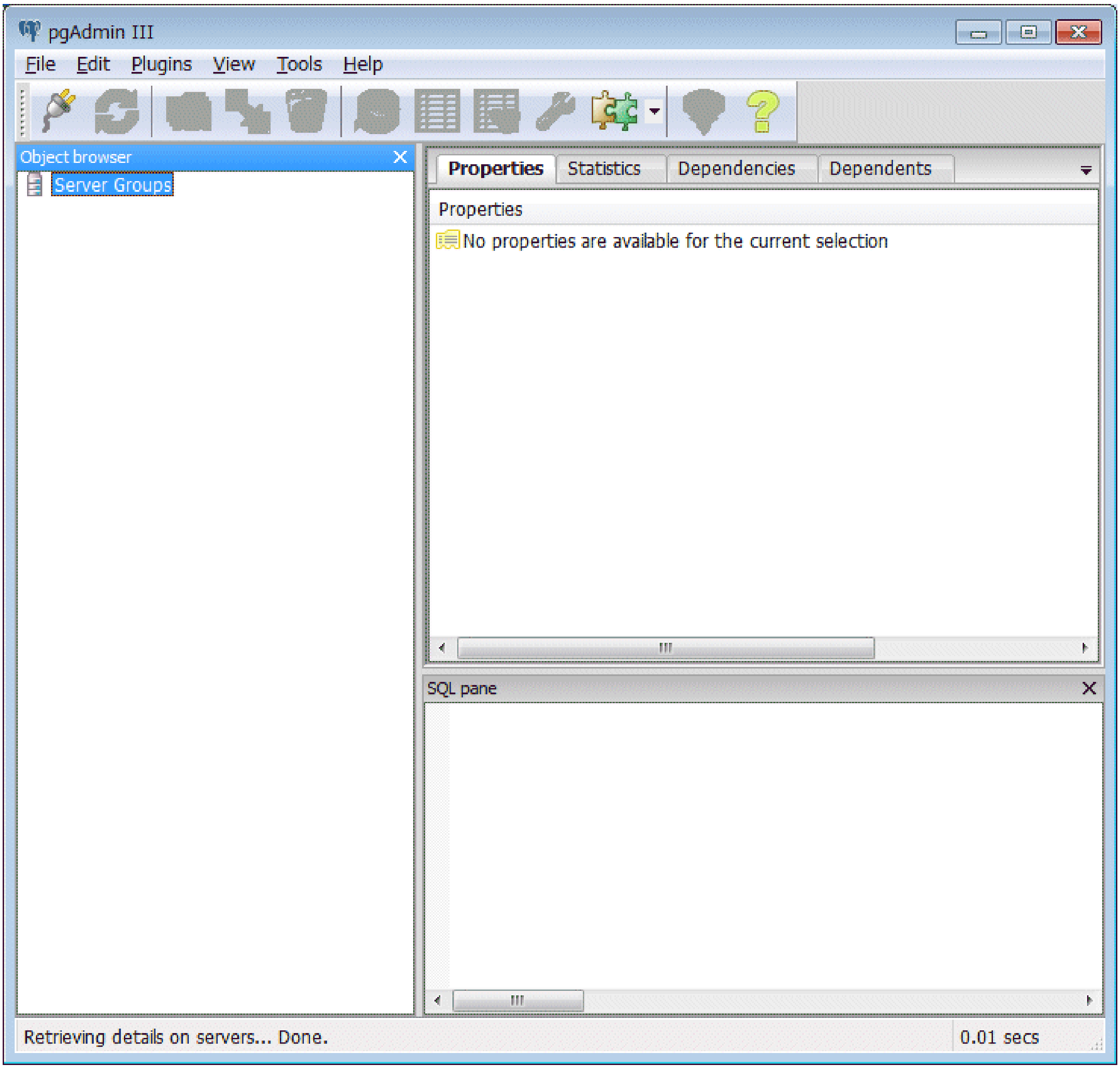
Windows(R) 10

Click [Start] >> [All apps] >> [FUJITSU Enterprise Postgres Client(AAbit)] and start [pgAdmin III (*AA*bit) (*x.y* SP*z*)].

Other operating systems

Click [Start] >> [All Programs] >> [FUJITSU Enterprise Postgres Client(*AA*bit) *x.y* SP*z*] and start [pgAdmin III (*AA*bit) (*x.y* SP*z*)].

The following window is displayed when pgAdmin starts.



pgAdmin III
File Edit Plugins View Tools Help
Object browser
Server Groups
Properties Statistics Dependencies Dependents
Properties
SQL pane
0.01 secs

No properties are available for the current selection

Retrieving details on servers... Done.

Note

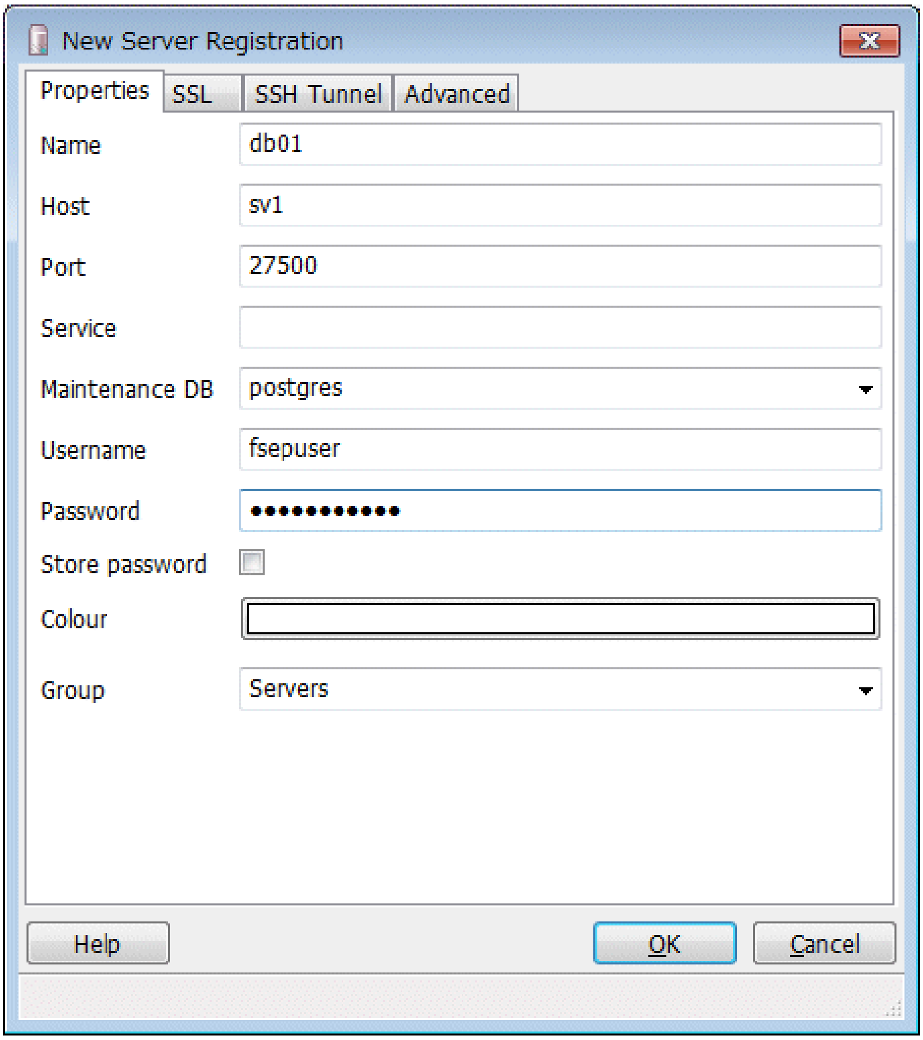
- You must start the instance to be connected to before using pgAdmin.
- Refer to "2.1 Starting and Stopping an Instance" for information on how to start an instance.
- Adobe(R) Reader(R) X is required for browsing the manual from [FUJITSU Enterprise Postgres Help] in pgAdmin.

1.3.2 Adding an Instance

This section describes how to add an instance to be connected to.

1. From the [File] menu in pgAdmin, click [Add Server].

2. In the [New Server Registration] window, specify a value for each item.



Name	db01
Host	sv1
Port	27500
Service	
Maintenance DB	postgres
Username	fsepuser
Password	●●●●●●●●●●●●
Store password	□
Colour	
Group	Servers

([Properties] tab)

- [Name]: Name of the instance to be managed
- [Host]: Host name or IP address of the server where FUJITSU Enterprise Postgres is installed
- [Port]: Port number of the instance
- [Username]: User name of the instance administrator
- [Password]: Password for the user name specified in [Username]

When you add an instance using pgAdmin, the instance is automatically connected to immediately after the addition is completed.

Note

If you select [Store password], a file storing the FUJITSU Enterprise Postgres connection password is created in the following location. Set the appropriate access permissions for the password file to protect it from unauthorized access.

- %APPDATA%\postgresql\pgpass.conf

1.3.3 Connecting/Disconnecting an Instance

This section describes how to connect pgAdmin to an instance, and how to disconnect it.

Note

To connect to an instance created using WebAdmin, you must first configure the settings in the [Client Authentication] window of WebAdmin to permit connection from pgAdmin.

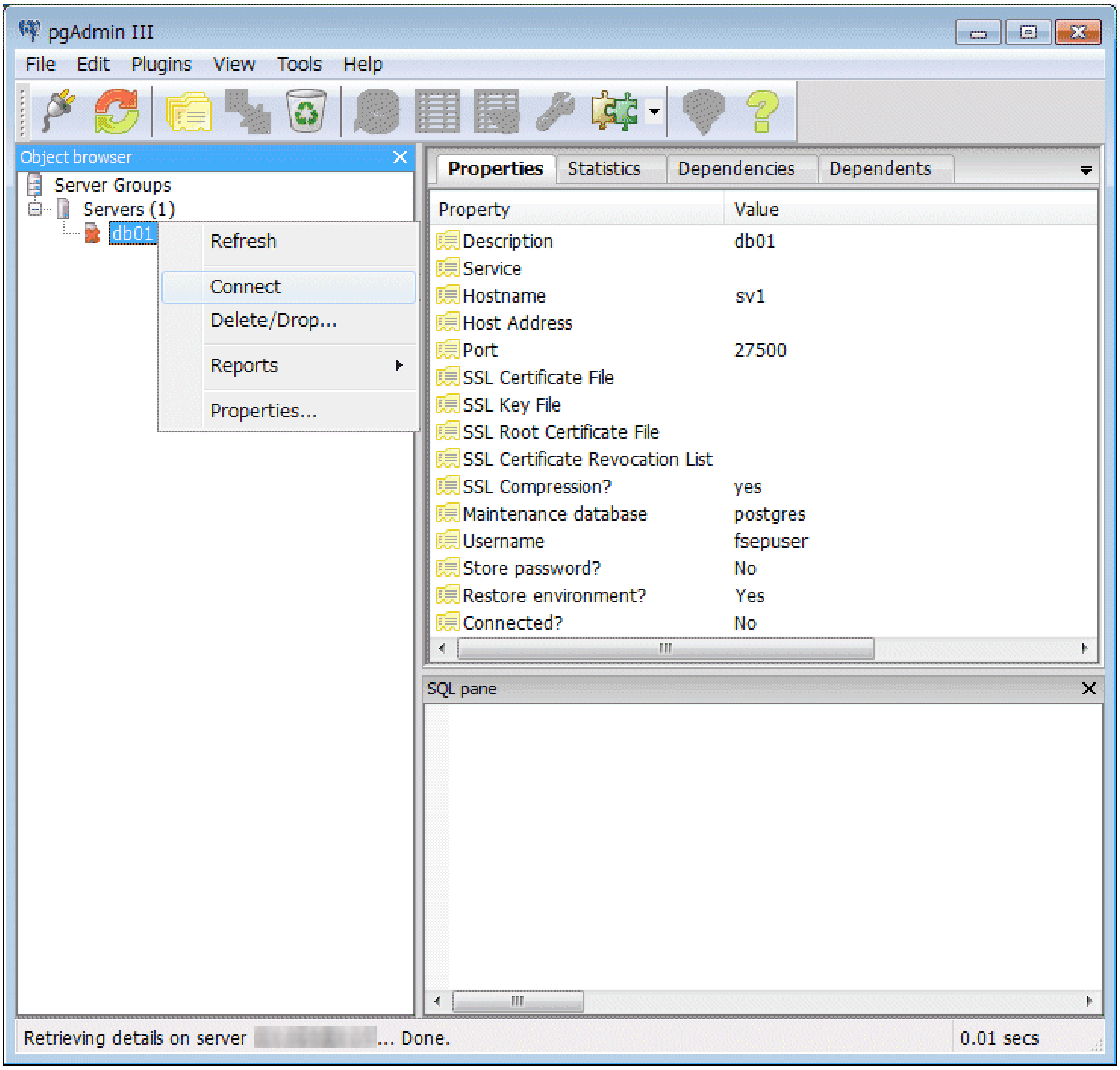
See

Refer to "Changing the settings" in the Installation and Setup Guide for Server for information on the [Client Authentication] window of WebAdmin.

Connecting to an instance

Starting pgAdmin does not connect it to any instance.

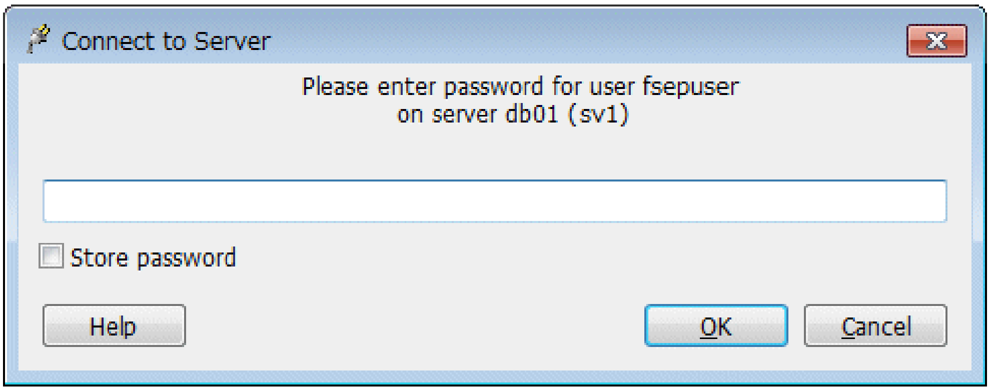
To connect to an instance, right-click the instance in [Object browser] and select [Connect].



Property	Value
Description	db01
Service	
Hostname	sv1
Host Address	
Port	27500
SSL Certificate File	
SSL Key File	
SSL Root Certificate File	
SSL Certificate Revocation List	
SSL Compression?	yes
Maintenance database	postgres
Username	fsepuser
Store password?	No
Restore environment?	Yes
Connected?	No

Retrieving details on server [redacted]... Done. 0.01 secs

If a password was not saved when the instance was added, the following password entry window is displayed.



Connect to Server

Please enter password for user fsepuser
on server db01 (sv1)

□ Store password

Disconnecting from an instance

To disconnect from an instance, right-click the server in [Object browser] in the pgAdmin window and select [Disconnect server].

1.4 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" under the procedure for creating an instance in "Using the initdb Command" in the Installation and Setup Guide for Server for information on the values to be set in 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 need to configure the environment variables.

  See

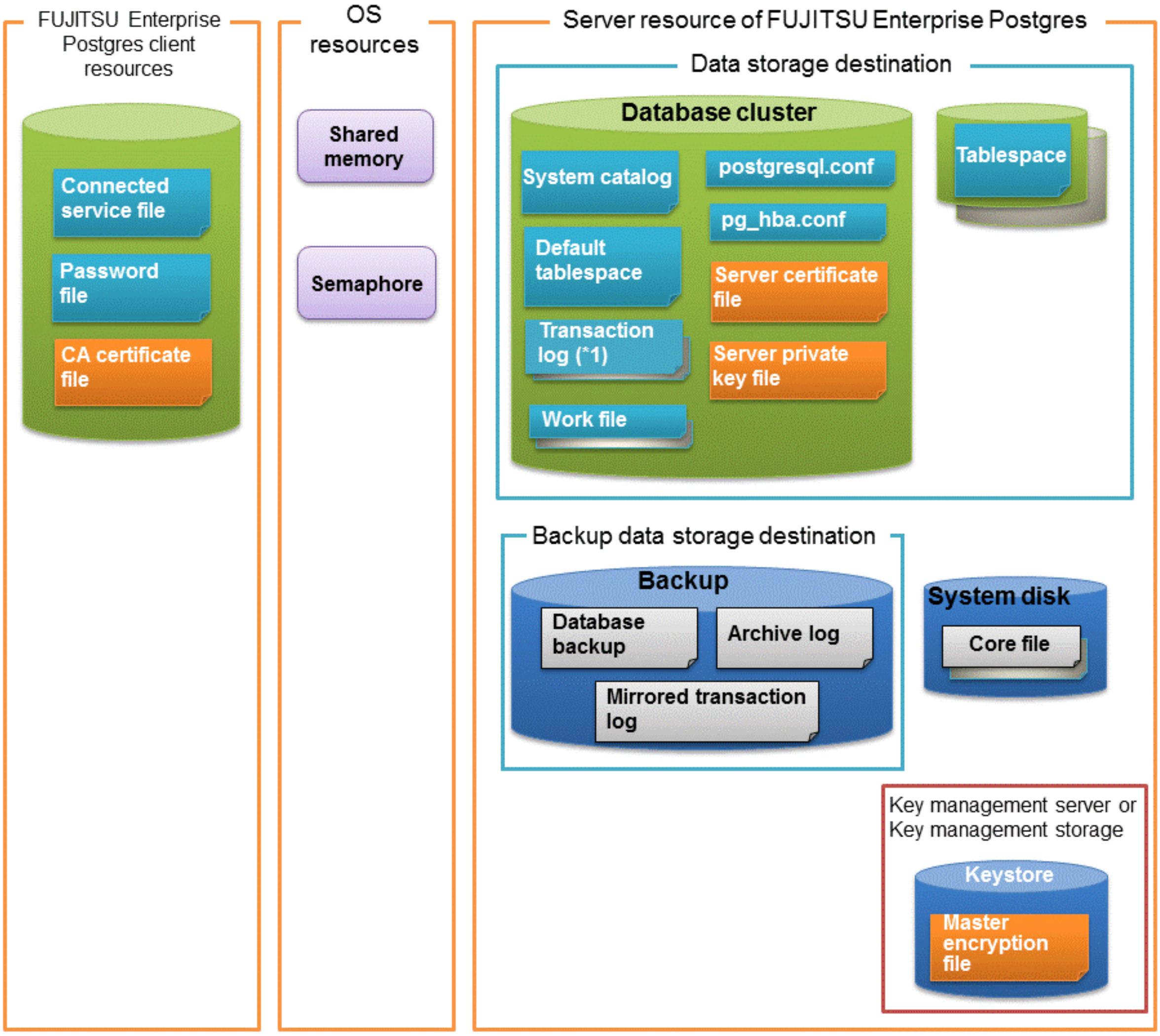
  - Refer to "PostgreSQL Client Applications" under "Reference" in the PostgreSQL Documentation, or "Reference" for information on client commands.
  - Refer to "Configuring Environment Variables" in the Installation and Setup Guide for Client for information on the values to be set in the environment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

1.5 Operating Environment of FUJITSU Enterprise Postgres

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

1.5.1 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
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.
Core file	FUJITSU Enterprise Postgres process core file that is output when an error occurs during an 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.

1.5.2 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 "<*xy*>" indicates the product version and level. **64-bit product** %Program Files%\Fujitsu\fsepv<*xy*>server64 **32-bit product (when installed on a 64-bit OS)** %Program Files(x86)%\Fujitsu\fsepv<*xy*>server32 **32-bit product (when installed on a 32-bit OS)** %Program Files%\Fujitsu\fsepv<*xy*>server32
Database cluster	Y	1	Specify using WebAdmin or server commands.
Tablespace	Y	Multiple	Specify using pgAdmin or the DDL statement.
Backup	Y	Multiple	Specify using WebAdmin or server commands.

File type	Required	Quantity	How to specify the location
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

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.

1.6 Notes on Compatibility of Applications Used for Operations

When you upgrade FUJITSU Enterprise Postgres to a newer version, there may be some affect 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 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.

Note

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.	

*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.

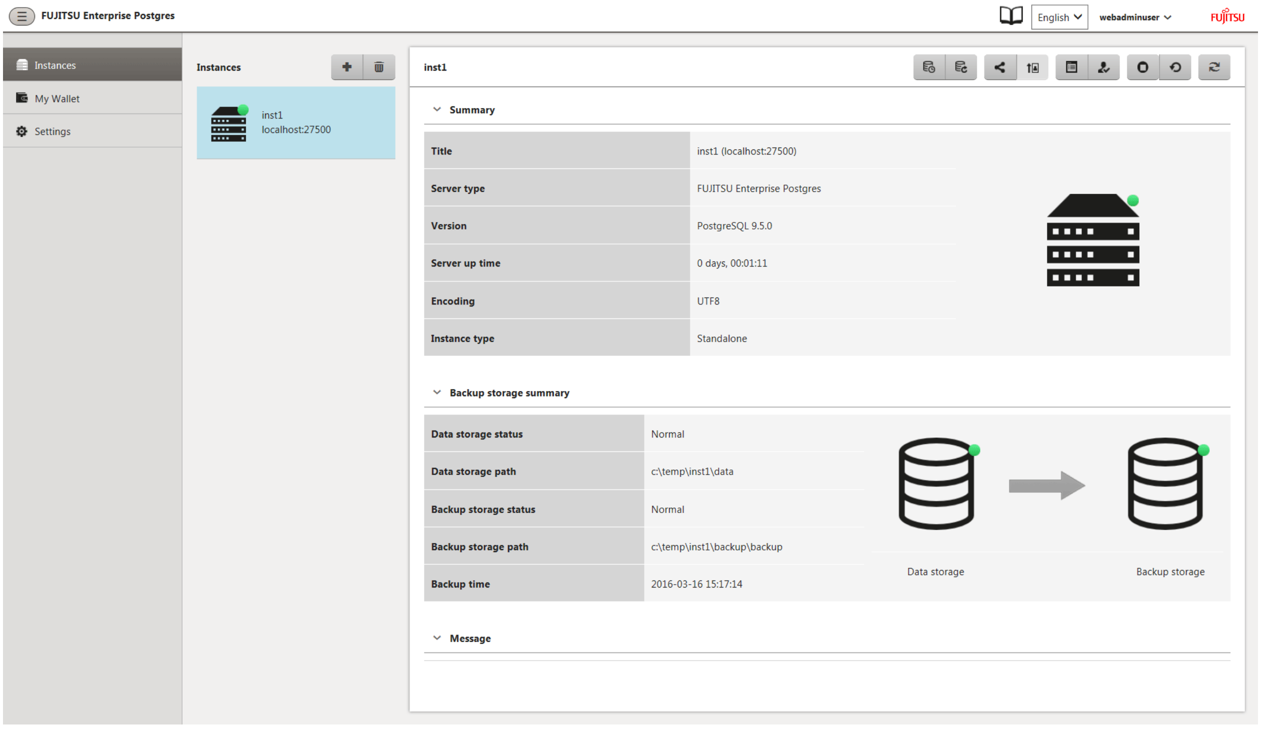
Checking the operating status of an instance

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.

Figure 2.1 Status when an instance is active

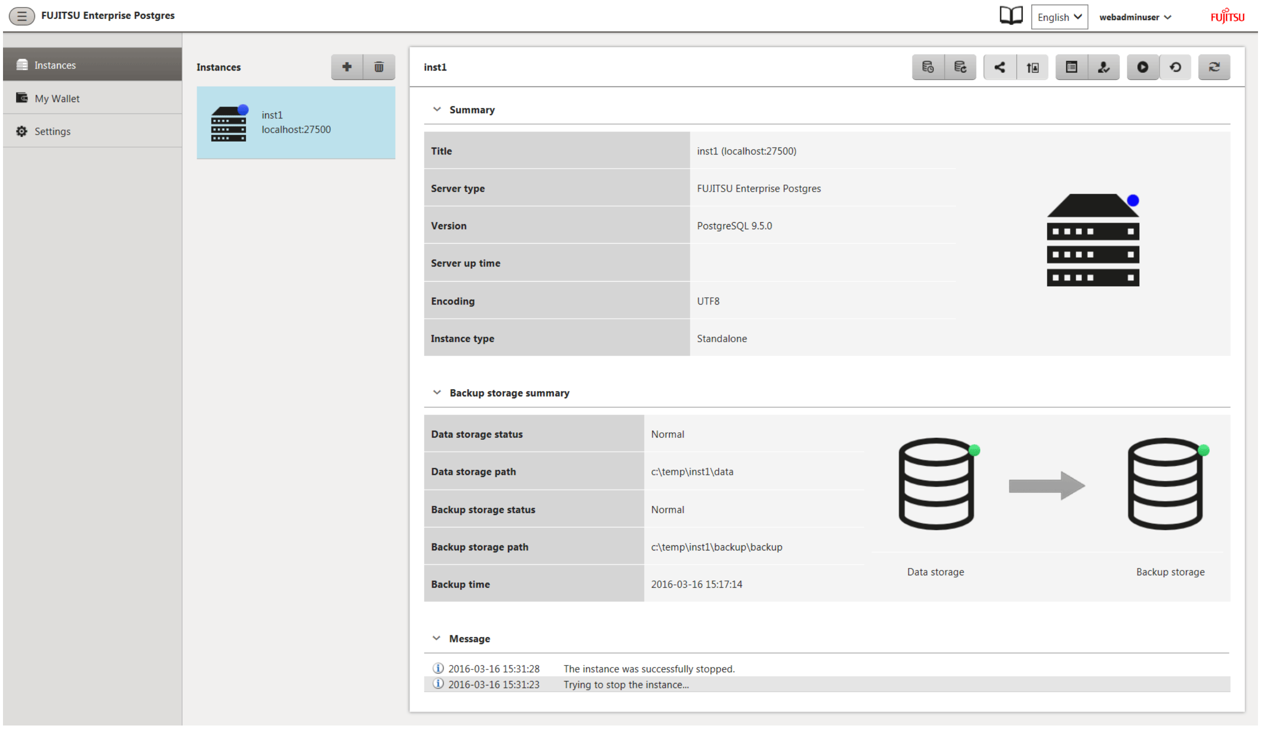


Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	0 days, 00:01:11
Encoding	UTF8
Instance type	Standalone

Backup storage summary

Data storage status	Normal
Data storage path	c:\temp\inst1\data
Backup storage status	Normal
Backup storage path	c:\temp\inst1\backup\backup
Backup time	2016-03-16 15:17:14

Figure 2.2 Status when an instance is stopped



FUJITSU Enterprise Postgres

English | webadminuser | FUJITSU

- Instances
- My Wallet
- Settings

Instances

inst1
localhost:27500

inst1

Summary

Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	
Encoding	UTF8
Instance type	Standalone

Backup storage summary

Data storage status	Normal
Data storage path	c:\temp\inst1\data
Backup storage status	Normal
Backup storage path	c:\temp\inst1\backup\backup
Backup time	2016-03-16 15:17:14

Message

```
ⓘ 2016-03-16 15:31:28   The instance was successfully stopped.
ⓘ 2016-03-16 15:31:23   Trying to stop the instance...
```

Note

- 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 event 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 Commands

The Windows service-related commands enable you to start or stop an instance and to check its operating state.

If you are to use Windows services, you should register instances in Windows services.

See

Refer to "When an instance was created with WebAdmin" in "Configuring Automatic Start and Stop of an Instance" in the Installation Guide for Server for information on registering instances in Windows services.

Note

While it is also possible for you to execute the pg_ctl command to start and stop instances without having to register instances in Windows services, it is recommended that you use Windows services to start and stop instances for the following reason:

- If you use the pg_ctl command to start an instance, the instance will be started as a user process. Therefore, when you close the [Command Prompt] window in which you executed the command, Windows forces the postgres process to stop.

Starting an instance

You can start an instance by specifying the service name in the net start command or sc start command.

Also, you can use the following procedure to start an instance in the Windows services window:

1. Display the [Services] window

   - Windows Server(R) 2012 and Windows Server(R) 2012 R2:

     In the [Start] screen, select [Administrative Tools], and then click [Services].

   - All other operating systems:

     In the [Start] menu, select [Administrative Tools], and then click [Services].

2. Start a service

   Select the instance name that you wish to start from the services list, and click [Start Service].

Stopping an instance

You can stop an instance by specifying the service name in the net stop command or sc stop command.

Also, you can use the following procedure to stop an instance in the Windows services window:

1. Display the [Services] window

   - Windows Server(R) 2012 and Windows Server(R) 2012 R2:

     In the [Start] screen, select [Administrative Tools], and then click [Services].

   - All other operating systems:

     In the [Start] menu, select [Administrative Tools], and then click [Services].

2. Stop the service

   Select the instance name that you wish to stop from the services list, and click [Stop Service]. If you stop a service while applications and commands are running, FUJITSU Enterprise Postgres will force those applications and commands to close and will stop normally.

Checking the operating state of an instance

Use the following procedure to check if an instance is operating correctly immediately after performing the operation to start an instance:

1. Display the [Services] window

   In the [Start] menu, select [Administrative Tools], and then click [Services].

2. Check the state of the service

   In the services list, check the state of the services for the applicable FUJITSU Enterprise Postgres.

To check the operating state of an instance during operation, use the pg_ctl command.

Specify the following in the pg_ctl command:

- Specify "status" 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.

Example

When the instance is active:

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

When the instance is inactive:

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

Information

You can also use the net start command or sc query command to check the operating state of an instance.

See

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

2.2 Creating a Database

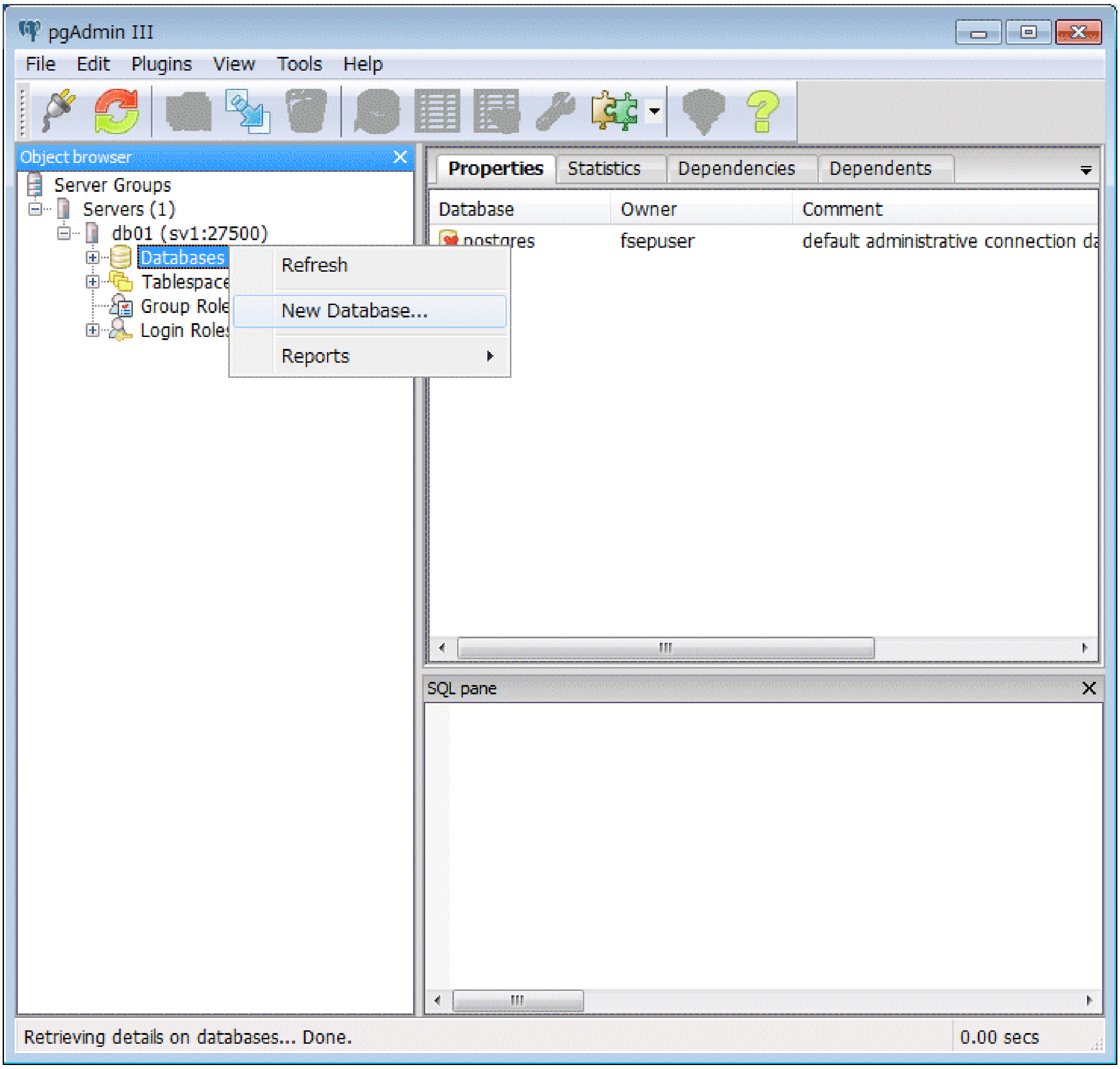
This section explains how to create a database.

- 2.2.1 Using pgAdmin
- 2.2.2 Using Client Commands

2.2.1 Using pgAdmin

Follow the procedure below to define a database using pgAdmin.

1. In the pgAdmin window, right-click [Database] in [Object browser], and then click [New Database] to display a new database window.



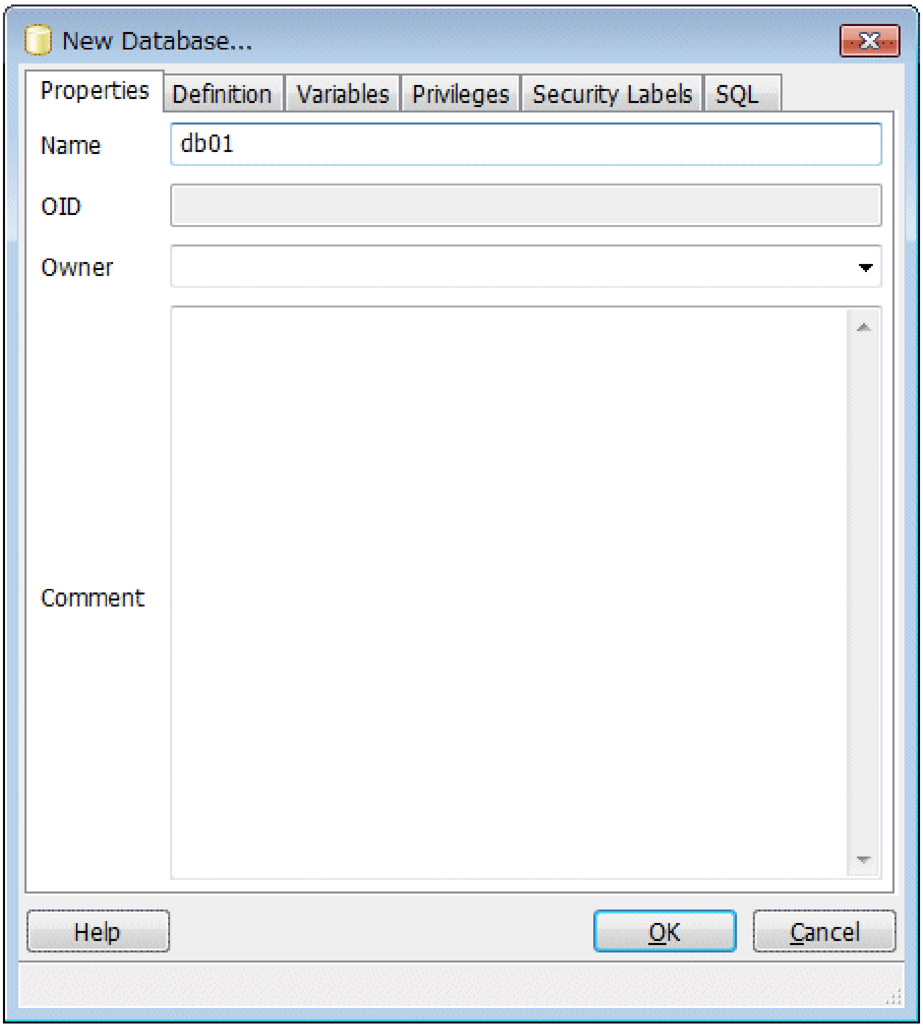
Database	Owner	Comment
postgres	fsepuser	default administrative connection da

Retrieving details on databases... Done. 0.00 secs

2. Specify appropriate values for the following items in the new database window.

   - [Properties] tab

     The following example illustrates creation of the database "db01".



Name: db01

     - [Name]: Name of the database to be managed

3. Click [OK] to create the database.

2.2.2 Using Client Commands

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 (9.5.2)
Type "help" for help.
```

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 the \l+ command, 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.

  The pgx_rcvall command is used for recovery.

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)

Note

- Backup operation cannot be performed on an instance that is part of a streaming replication cluster in standby mode.
- 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.

- 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

- 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 event log, create the tablespace, and then perform the recovery process again.
- If performing backups with WebAdmin, the following password file is temporarily created during backup for WebAdmin to connect to the database:

  - *userProfileFolder*\localSettingsFolder\Fujitsu\fsep_*version\instanceName*\pgpass.conf

  Therefore, when you are backing up corefiles created in the core_directory parameter of postgresql.conf, or log files created in the log_directory parameter of postgresql.conf, ensure not to back up the password files located in the same directories at the same time.

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.

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.

Note

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.

Note

WebAdmin uses the labels "Data storage path", "Backup storage path" and "Transaction log path" to indicate "data storage destination", "backup data storage destination" and "transaction log storage destination" respectively. In this manual these terms are used interchangeably.

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 .

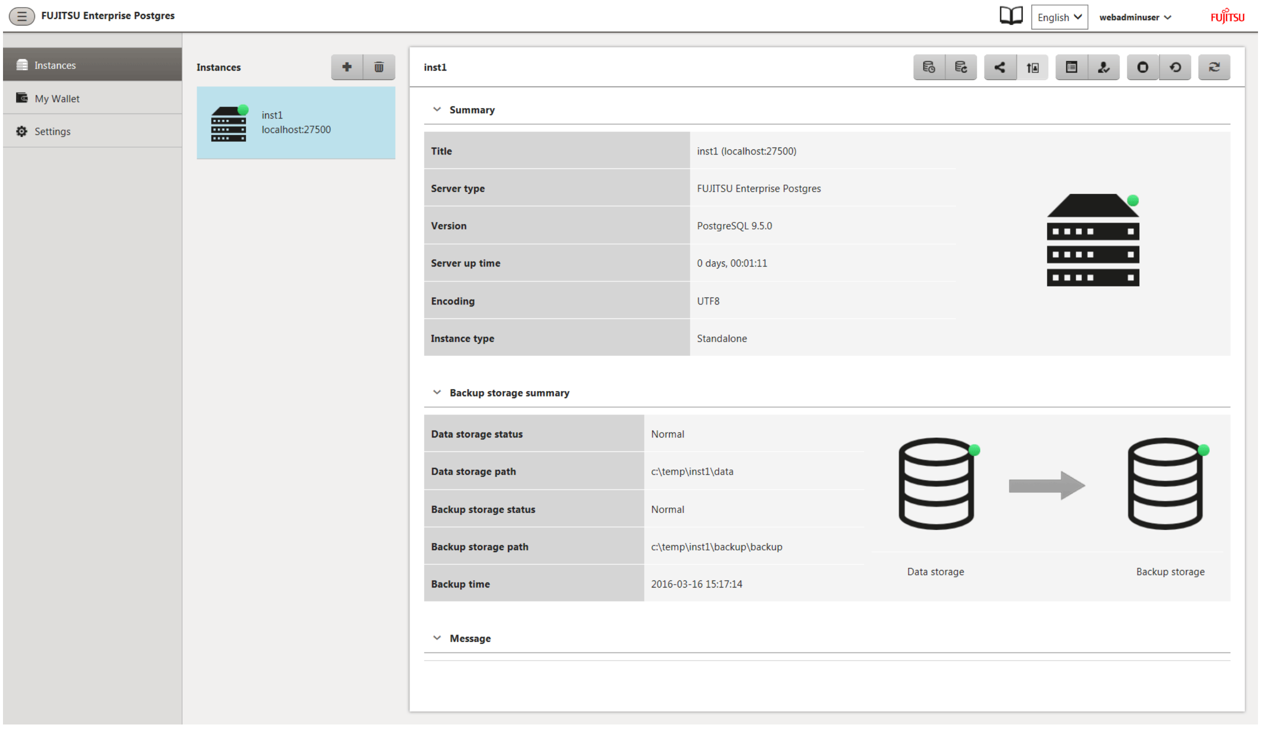
2. Back up the database

   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 destination] or [Backup data storage destination] 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.



Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	0 days, 00:01:11
Encoding	UTF8
Instance type	Standalone

Data storage status	Normal
Data storage path	c:\temp\inst1\data
Backup storage status	Normal
Backup storage path	c:\temp\inst1\backup\backup
Backup time	2016-03-16 15:17:14

Note

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.

   In [Properties] in Windows(R) Explorer, set appropriate permissions so that only the instance administrator can access the directory.

   ### See

   Refer to [Help and Support] in Windows(R) for information on [Properties].

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.
wal_level	archive or hot_standby(*1)	Specify the output level for the transaction log. *1: hot_standby is a setting for streaming replication.
archive_mode	on	Specify the archive log mode. Specify [on] (execute).
archive_command	'cmd /c ""*installationDirectory*\\bin\\pgx_xlogcopy.cmd" "%p" "*backupDataStorageDestinationDirectory*\\archived_xlog\\%f""'	Specify the path name of the command that will save the transaction log and the storage destination. Note the following when specifying the path: - Specify \\ as the path delimiter. - Enclose the path in double quotes ("") if it contains spaces.

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

Backup operation

Use the pgx_dmpall command to perform 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 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 D:\database\inst1
Date                    Status          Dir
2015-05-01 13:30:40     COMPLETE        E:\backup\inst1\2015-05-01_13-30-40
```

If an error occurs and backup fails, a message is output to the event 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_20150503_1');
LOG:  restore point "batch_20150503_1" created at 0/20000E8
STATEMENT:  select pg_create_restore_point('batch_20150503_1');
 pg_create_restore_point
-------------------------
 0/20000E8
(1 row)
```

Refer to "10.3.2 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

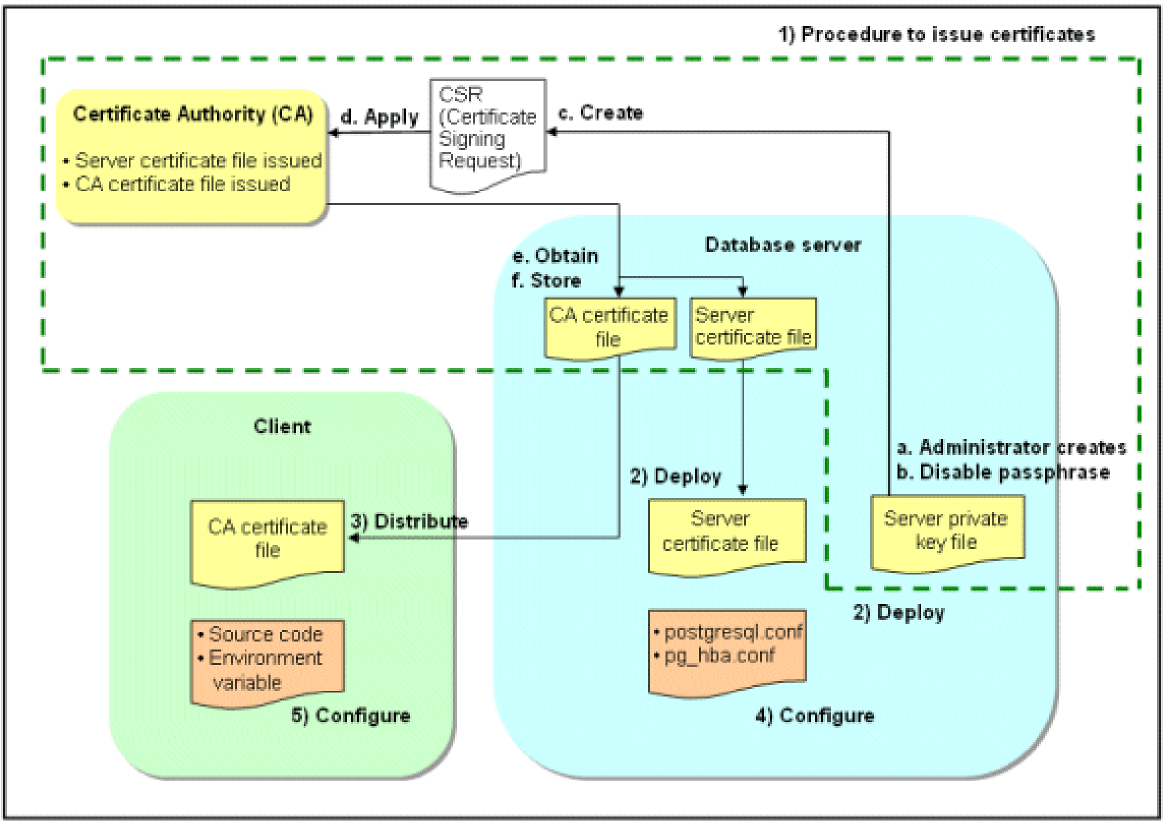
To encrypt communication data transferred between a client and a server, configure communication data encryption as described below. Communication data encryption not only protects the communication content, but it also guards against man-in-the-middle (MITM) attacks (for example, data and password theft 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, certificates with the same "Common Name" must be used. To ensure this, take one of the following actions:

- Create one server certificate, replicate it, and place a copy on each server used for database multiplexing.
- Create a server certificate with the same "Common Name" 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.

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.

Faster encryption and decryption based on hardware

TDE minimizes the overhead of encryption and decryption by using the AES-NI (Advanced Encryption Standard New Instructions) built into Intel(R) Xeon(R) processors since the 5600 series. This means that even in situations where previously the minimum encryption target was selected as a tradeoff between performance and security, it is now possible to encrypt all the data of an application.

You can reference a list of processors equipped with AES-NI on the following page at Intel Corporation's website:

http://ark.intel.com/search/advanced/?s=t&AESTech=true

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.

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.

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

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.

   ```
   keystore_location = 'C:\\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 commands

     Refer to "2.1.2 Using Commands", and restart the instance.

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, select one of the following methods:

- Use an automatically opening keystore

  Select this method if ease of operation has priority over enhanced security. When using an automatically opening keystore, the content of the keystore file is decrypted and a copy of the keystore file is generated. Although the content of this file is obfuscated, the level of security becomes slightly weaker.

  Select this method if performing operations using WebAdmin.

- Enter a passphrase when starting an instance

  Select this method if enhanced security has priority over ease of operation.

  Specify the --keystore-passphrase in the pg_ctl command and start the instance. This displays the prompt that asks for the passphrase to be entered.

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

  After performing the above operation, use the pg_ctl command to stop the instance.

  Then start the instance in Windows services. Refer to "2.1.2 Using Commands" for information on how to start an instance in Windows services.

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 'C:\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' tablespace_encryption_algorithm =
'AES256';
```

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.

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

Note

An encrypted tablespace cannot be created from the window used for creating the pgAdmin tablespace, or from the query tool. To create an encrypted tablespace, click [PSQL Console] from the [Plugins] menu and create an encrypted tablespace in the psql console window.

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 "C.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 C:\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.

Set the permission mode in [Properties] in Windows(R) Explorer.

See

Refer to [Help and Support] in Windows(R) for information on [Properties].

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

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

Note

- To use WebAdmin for recovery, you must enable automatic opening of the keystore.
- Refer to "5.7 Backing Up and Restoring/Recovering the Database" after enabling or reconfiguring encryption to back up the database.
- 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

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 May 1, 2015.

```
> pgx_dmpall -D D:\database\inst1
> copy C:\key\store\location\keystore.ks C:\keybackup\keystore_20150501.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 May 5, 2015.

```
> psql -c "SELECT pgx_set_master_key('passphrase')" postgres
> copy C:\key\store\location\keystore.ks C:\keybackup\keystore_20150505.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 May 5, 2015.

```
> copy C:\keybackup\keystore_20150505.ks C:\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 May 2015.

```
> copy C:\keybackup\keystore_20150501.ks C:\key\store\location\keystore.ks
> pgx_rcvall -B E:\backup\inst1 -D 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 May 1, 2015.

```
> pgx_dmpall -D D:\database\inst1
> copy C:\key\store\location\keystore.ks C:\keybackup\keystore_20150501.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 May 1, 2015.

```
> copy C:\keybackup\keystore_20150501.ks C:\key\store\location\keystore.ks
> pgx_keystore --enable-auto-open C:\key\store\location\keystore.ks (Execute only when enabling
automatic opening)
> pgx_rcvall -B E:\backup\inst1 -D 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 For this reason, the pg_restore 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, refer to "5.3 Opening the Keystore" for information on starting an instance by specifying pg_ctl --keystore-passphrase start.

See

- Refer to "Reference" in the PostgreSQL Documentation for information on the following commands:
  - psql
  - pg_dump
  - pg_restore
  - 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 use file transfer software that performs encrypted communication for Windows, 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:

- fsutil command

Example

```
# Export the contents of the table my_table to a CSV file.
> psql -c "COPY my_table TO 'C:\WINDOWS\Temp\my_table.csv' (FORMAT CSV)" postgres
# Encrypt the exported file.
> C:\OpenSSL-Win32\bin\openssl enc -e -aes256 -in C:\WINDOWS\Temp\my_table.csv -out my_table.csv.enc
(The user is prompted to enter the passphrase to be used for encryption)

# Check the size of plain text files, and delete them after zero padding
> dir C:\WINDOWS\Temp\my_table.csv
> fsutil file setzerodata offset=0 length=7 C:\WINDOWS\Temp\my_table.csv
> del C:\WINDOWS\Temp\my_table.csv

# Decrypt the encrypted files on other systems.
> C:\OpenSSL-Win32\bin\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.

See

Refer to the Cluster Operation Guide for information on building a cluster system environment using failover operation.

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.

If specifying --keystore-passphrase in the pg_ct command, refer to "5.3 Opening the Keystore" for details.

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_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 minidump, which is the process memory dump. You should, therefore, safely delete the memory dump. You can safely delete files by using the following command:
  - fsutil 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.

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 'C:\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 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.

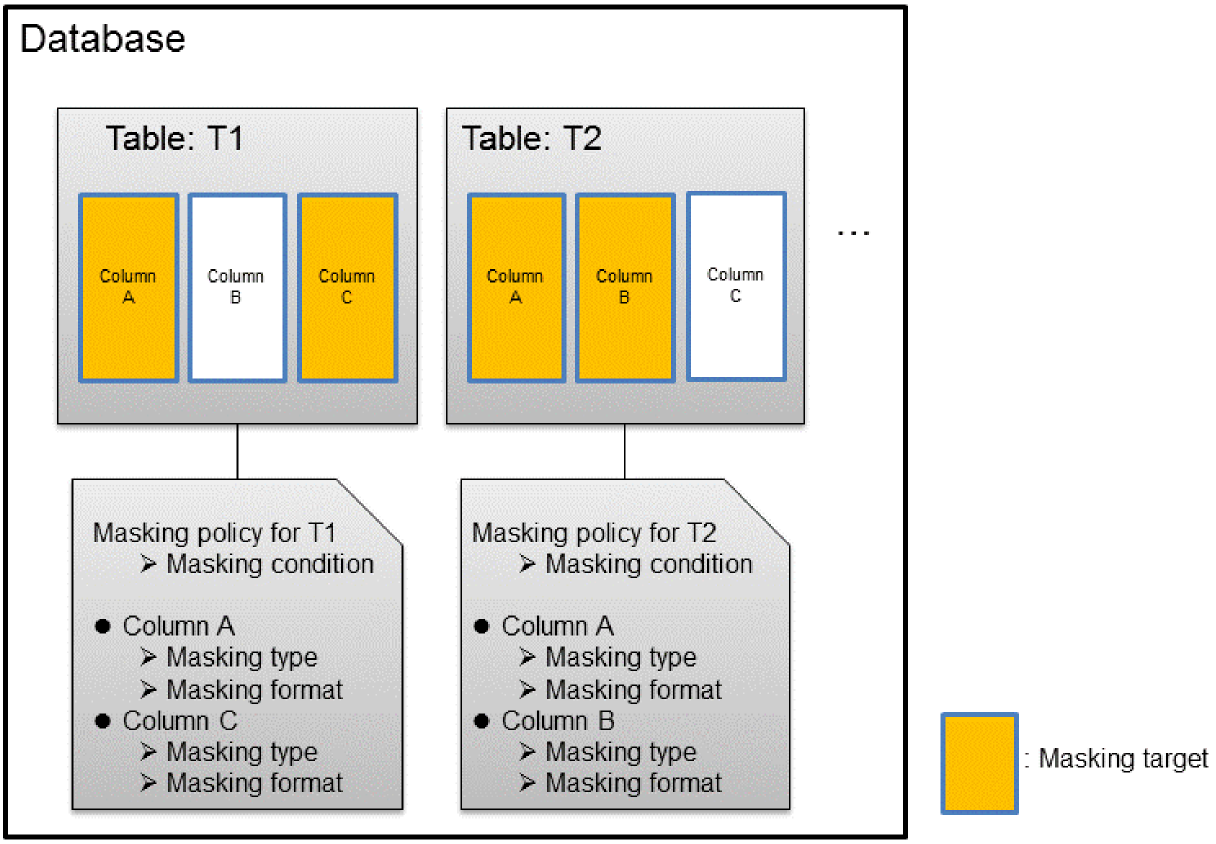
Note

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.

6.1 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 6.1 Masking policy



Note

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

6.1.1 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 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.

6.1.2 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.

6.1.3 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.

6.1.4 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 "6.3 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. - *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*.

Category	Method of specifying function_parameters
	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 "2010-10-10 10:10:10", it will be changed to "2010-10-10 00:00:00".

See

- Refer to "B.3.2 pgx_create_confidential_policy" for information on function_parameters.
- Refer to "6.3 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 "6.3 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.

6.2 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. Run 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.

6.2.1 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.3.2 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.

6.2.2 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.3.1 pgx_alter_confidential_policy" for information on the pgx_alter_confidential_policy system management function.

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

6.2.4 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.3.4 pgx_enable_confidential_policy" for information on the pgx_enable_confidential_policy system management function.

6.2.5 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.3.3 pgx_drop_confidential_policy" for information on the gx_drop_confidential_policy function.

6.3 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
Character type	character varying(*n*)	Y	Y	Y
	varchar(*n*)	Y	Y	Y

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

Chapter 7 Periodic Operations

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

7.1 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 event 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.

7.2 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

7.2.1 Monitoring Disk Usage

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

- fsutil volume diskfree 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%

7.2.2 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.

- del 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 "7.5 Reorganizing Indexes" for details.

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

  Execute backup using WebAdmin or the pgx_dmpall command.

7.3 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.

Note

The maximum number of connections allowed is 125, unless the desktop heap setting is changed.

See

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

7.4 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 two methods are supported for monitoring connections that have been in the waiting status for an extended period:

- 7.4.1 Using the View (pg_stat_activity)
- 7.4.2 Using pgAdmin

7.4.1 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 state='idle in transaction' and current_timestamp >
cast(query_start + interval '60 minutes' as timestamp);
-[ RECORD 1 ]----+-----------------------------
datid            | 13003
datname          | db01
pid              | 4638
usesysid         | 10
usename          | fsep
application_name | apl01
client_addr      | 192.33.44.15
client_hostname  |
client_port      | 27500
backend_start    | 2015-04-24 09:09:21.730641+09
xact_start       | 2015-04-24 09:09:23.858727+09
query_start      | 2015-04-24 09:09:23.858727+09
state_change     | 2015-04-24 09:09:23.858834+09
waiting          | f
state            | idle in transaction
backend_xid      |
backend_xmin     |
query            | begin;
```

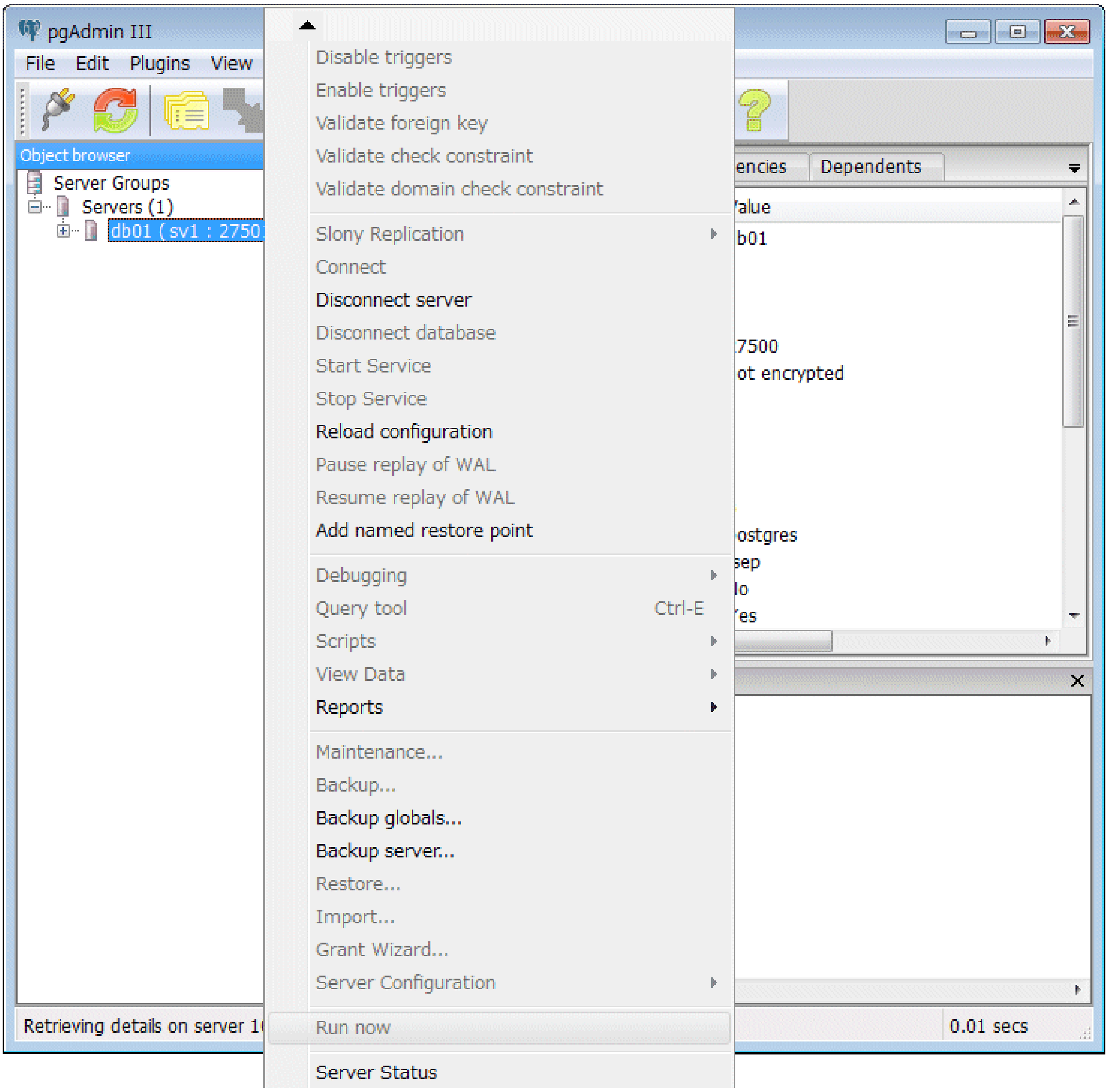
See

- 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.

7.4.2 Using pgAdmin

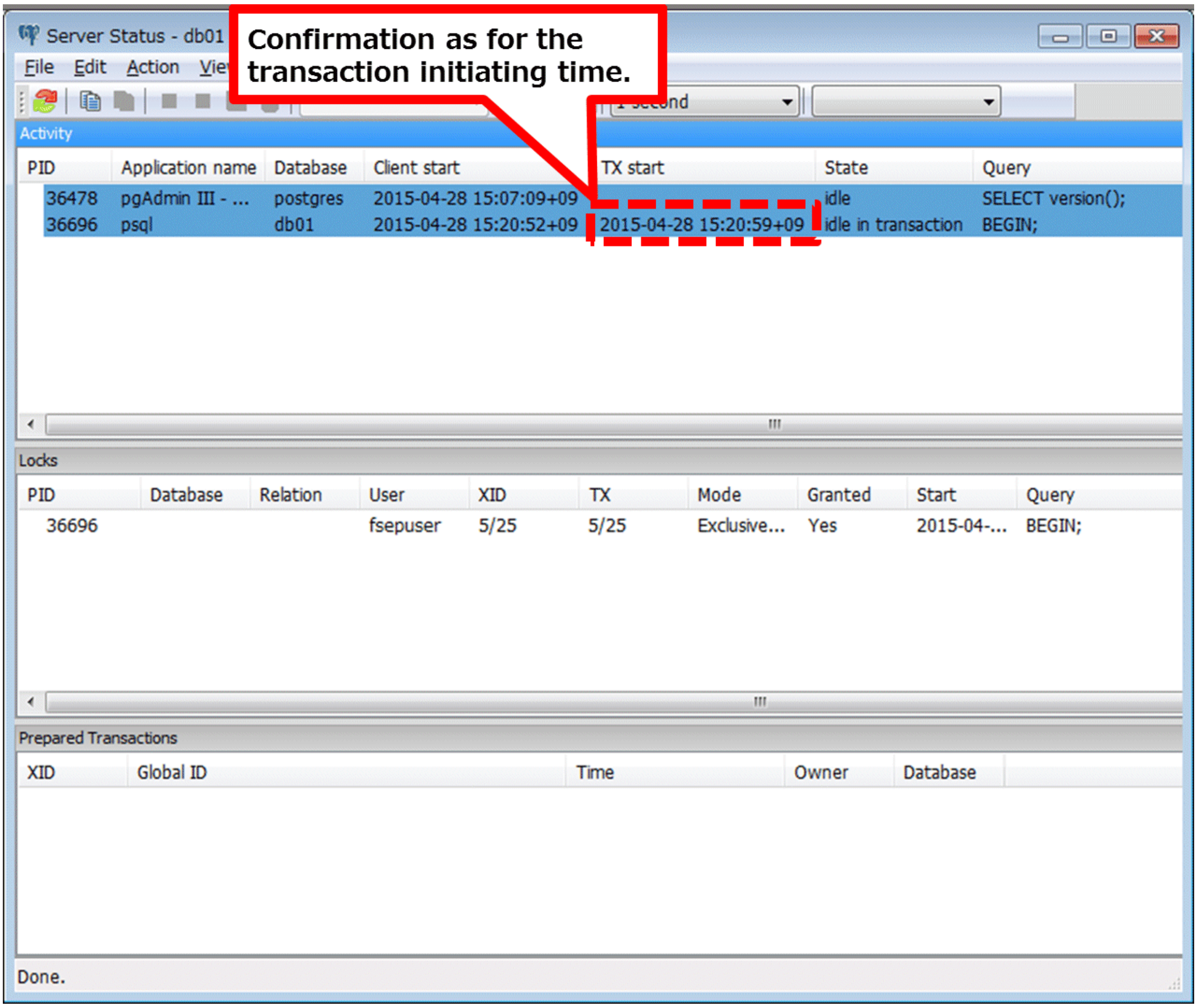
This section describes the procedure for monitoring connections using [Server Status] in pgAdmin.

1. From the [Tools] menu in pgAdmin, click [Server Status].



2. Identify client connections that have been in the waiting state for an extended period.

   From the transaction start time displayed under [TX Start], identify connections that have been in the waiting state for an extended period.



Confirmation as for the transaction initiating time.

PID	Application name	Database	Client start	TX start	State	Query
36478	pgAdmin III - ...	postgres	2015-04-28 15:07:09+09		idle	SELECT version();
36696	psql	db01	2015-04-28 15:20:52+09	2015-04-28 15:20:59+09	idle in transaction	BEGIN;

PID	Database	Relation	User	XID	TX	Mode	Granted	Start	Query
36696			fsepuser	5/25	5/25	Exclusive...	Yes	2015-04-...	BEGIN;

XID	Global ID	Time	Owner	Database

7.5 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 "7.2 Monitoring Disk Usage and Securing Free Space".

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

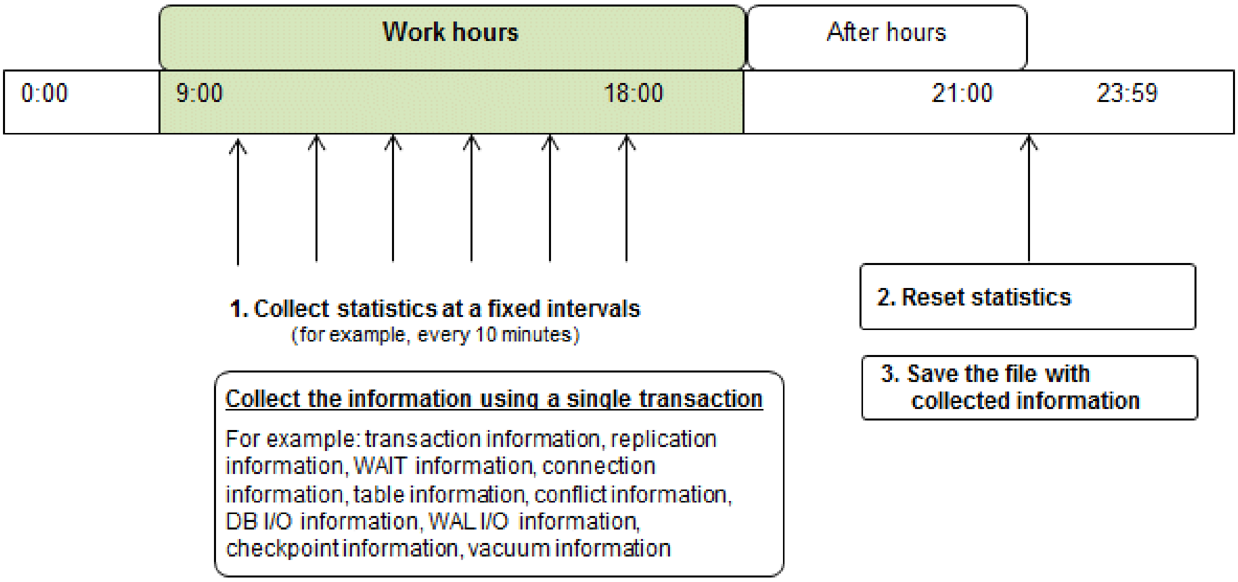
Refer to "Notes on Application Compatibility" in the Application Development Guide for information on maintaining application compatibility.

7.6 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 7.1 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)
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
2. Reset statistics
3. Save the file with collected 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 "7.6.1 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 "7.6.3 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

7.6.1 Information that can be Collected

Information that can be collected is categorized into the following two 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 7.1 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 "C.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 "C.3 pgx_stat_latch" for details.
pgx_stat_walwriter	Displays statistics related to WAL writing, in a single line. Refer to "C.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 "C.5 pgx_stat_sql" for details.

7.6.2 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
pgx_stat_sql	track_sql

Remarks: You cannot change the collection status for pgx_stat_walwriter.

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

7.6.3 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

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.

Chapter 8 Streaming Replication Using WebAdmin

This chapter describes how to create a streaming replication cluster using WebAdmin.

8.1 Streaming Replication

Streaming replication allows the creation of one or more standby instances, which connect to the master instances and replicate the data using WAL records. The standby instance can be used for read-only operations.

WebAdmin can be used to create a streaming replication cluster. WebAdmin allows the creation of a cluster in the following configurations:

- Master-Standby Configuration: This configuration creates a master and standby instance together.
- Standby Only Configuration: This configuration creates a standby instance from an already existing instance.

Point

- A standby instance can be created from a standalone instance, a master instance, or even from another standby instance.
- 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].

See

When a standby instance is created from an existing instance, it is necessary to set the values for the replication-related variables before a standby instance can be created. Refer to "Configuring Parameters" in the Installation and Setup Guide for Server for details.

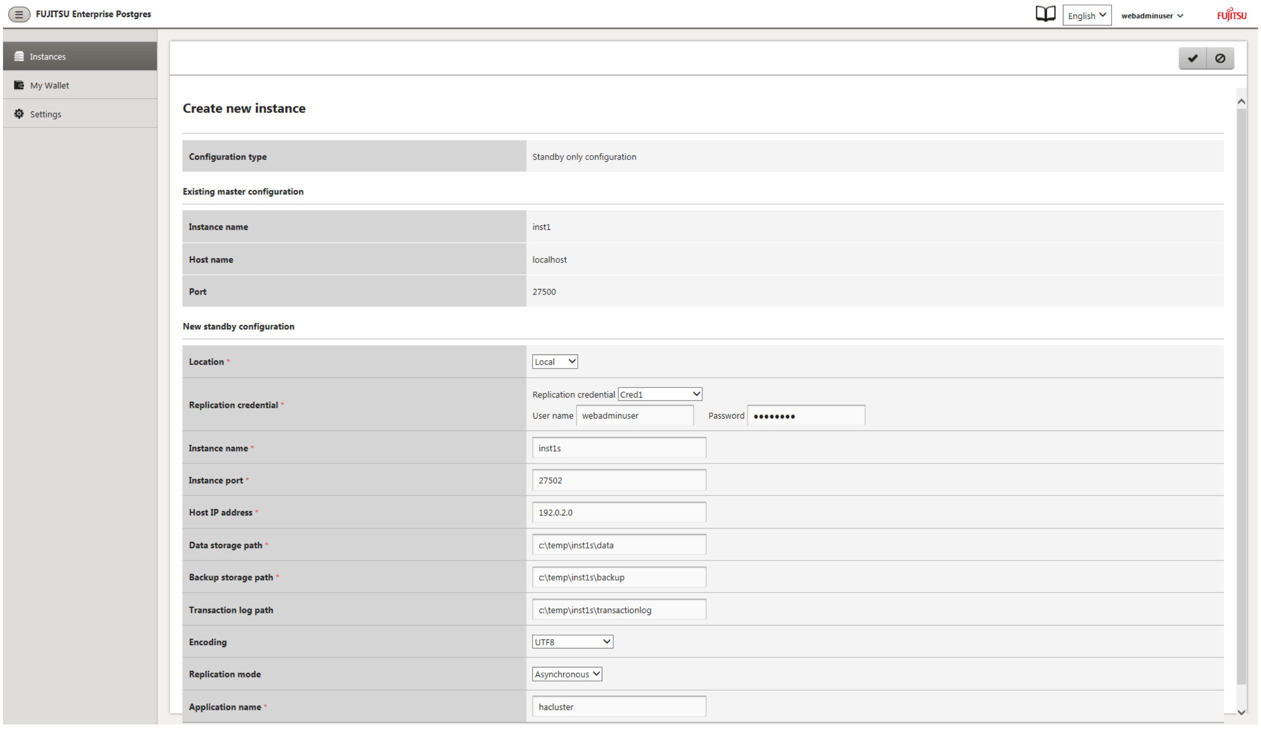
8.1.1 Creating a Standby Instance

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. In the example below, a standby instance is created from instance "inst1".

   The instance name, host address and port of the selected instance are already displayed for easy reference.



Create new instance

Configuration type	Standby only configuration

Existing master configuration

Instance name	inst1
Host name	localhost
Port	27500

New standby configuration

Location *	Local
Replication credential *	Replication credential Cred1 User name webadminuser Password ••••••••
Instance name *	inst1s
Instance port *	27502
Host IP address *	192.0.2.0
Data storage path *	c:\temp\inst1s\data
Backup storage path *	c:\temp\inst1s\backup
Transaction log path	c:\temp\inst1s\transactionlog
Encoding	UTF8
Replication mode	Asynchronous
Application name *	hacluster

   Enter the following items:

   - [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 F 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

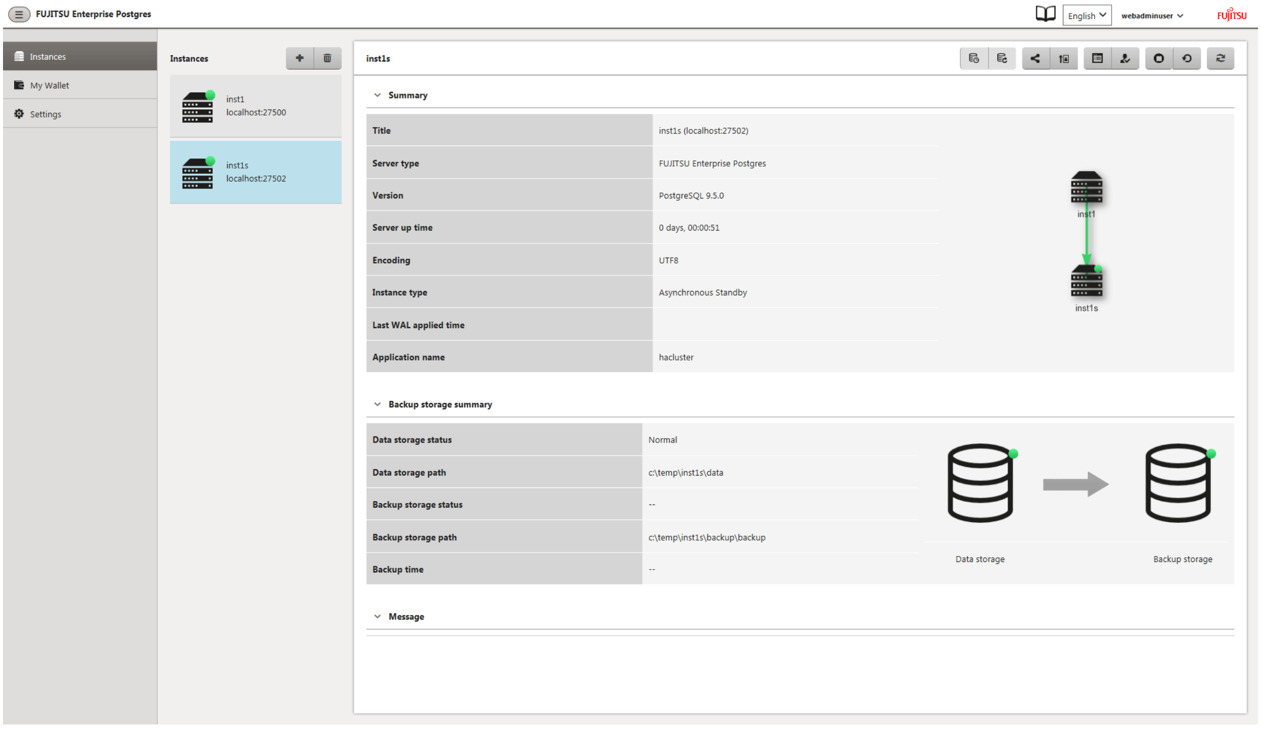
   - [Encoding]: Database encoding system

   - [Replication mode]: Whether the standby instance created should be in Asynchronous or Synchronous mode.

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

4. Click ✓ to create the standby instance.

5. Once the standby instance is created successfully, select "inst1s" in the [Instances] tab. The following page will be displayed:



Title	inst1s (localhost:27502)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	0 days, 00:00:51
Encoding	UTF8
Instance type	Asynchronous Standby
Last WAL applied time	
Application name	hacluster

Data storage status	Normal
Data storage path	c:\temp\inst1s\data
Backup storage status	--
Backup storage path	c:\temp\inst1s\backup\backup
Backup time	--

Note

- Backups are not possible for standby instances in WebAdmin. As a result, and are disabled and no value is shown for [Backup storage status] and [Backup time].

- 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 (16491)
ERROR:  pgx_rcvall: backup of the database has not yet been performed, or an incorrect backup
storage directory was specified
```

8.1.2 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 .

8.1.3 Converting an Asynchronous Replication to Synchronous

Streaming replication between a master and standby instance can be configured to be in Asynchronous or Synchronous mode. This mode can be changed even after the standby instance was successfully created.

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.

Note

- 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.
- 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.

8.1.4 Converting a Synchronous Replication to Asynchronous

Streaming replication between a master and standby instance can be configured to be in Asynchronous or Synchronous Mode. This mode can be changed even after the standby instance was successfully created.

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.

Note

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.

Chapter 9 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.

9.1 Installing Vertical Clustered Index (VCI)

This section describes the installation of VCI.

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

9.1.1 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 aggregation of sum and average

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 "9.1.4 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.

9.1.2 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.

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.

9.1.3 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

9.1.3.1 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	Literall '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
vci.smc_directory	Directory name in which a temporary file is created as the dynamic shared memory	Directory name in which a temporary file is created as the	N

Parameter name	Setting value	Description	Required
		dynamic shared memory during a scan using a VCI.	

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
vci.smc_directory = 'E:\\vci\\work'
```

Note

If performing operations that use VCI, do not delete the shared library names specified to "shared_preload_libraries" and "session_preload_libraries". If operations that access VCI are performed after the VCI was defined, unexpected behavior may occur.

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.

9.1.3.2 Installing the Extensions

Execute the CREATE EXTENSION statement 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 "9.2.1 Commands that cannot be Used for VCI" for details.

9.1.3.3 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 "9.1.4.1 Relation Types" for details.
- Some data types cannot be specified on the column specification of CREATE INDEX. Refer to "9.1.4.2 Data Types" for details.
- Some operations cannot be performed for VCI. Refer to "9.2.1 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.

9.1.3.4 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)
```

9.1.4 Data that can Use VCI

This section describes on which relation types and for which data types VCIs can be created.

9.1.4.1 Relation Types

VCIs cannot be created on some relation types.

The ON clause of CREATE INDEX described in "9.1.3.3 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

9.1.4.2 Data Types

VCIs cannot be created for some data types.

The column specification of CREATE INDEX described in "9.1.3.3 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	N
	nvarchar(*n*)	N
	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
Bit string	bit(*n*)	Y
	bit varying(*n*)	Y
Text search	tsvector	N
	tsquery	N

Category	Data type	Supported by VCI?
UUID	uuid	Y
XML	xml	N
JSON	json	N
	jbson	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

9.2 Operating VCI

This section describes how to operate VCI.

9.2.1 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	

Command	Subcommand	Description
CREATE EXTENSION	SCHEMA	The subcommands on the left cannot be performed if the VCI extension is specified. This operation is not required for VCI.
DROP EXTENSION	-	The VCI extension cannot be specified. (Restriction in this edition.)

- 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}	
	SET WITH OIDS	If WITHOUT OIDS is specified for a table that contains a VCI, the SET WITH OIDS clause cannot be used. If the operation is required, delete the VCI using DROP INDEX, and re-create it using CREATE INDEX after completing the operation.
	SET WITHOUT OIDS	If WITH OIDS is specified for a table that contains a VCI, the SET WITHOUT OIDS clause cannot be used.

Command	Subcommand	Description
		If the operation is required, delete the VCI using DROP INDEX, and re-create it using CREATE INDEX after completing the operation.
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. This operation is not supported in VCI.
	CONCURRENTLY	
	[ASC | DESC]	
	[NULLS { FIRST | LAST }]	
	WITH	
	WHERE	
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.

9.2.2 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.4 VCI Data Load Control Function" for information on pgx_prewarm_vci.

Chapter 10 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 "10.1 Recovering from Disk Failure (Hardware)", and take actions accordingly.

- If the disk is not defective

  Refer to "10.14 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 event 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.

- 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

10.1 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:

- 10.1.1 Using WebAdmin

- 10.1.2 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.

10.1.1 Using WebAdmin

Recover the database cluster by following the appropriate recovery procedure below for the disk where the failure occurred.

Note

Recovery operation cannot be performed on an instance that is part of a streaming replication cluster in standby mode.

If disk failure occurs on a standby instance, it may be necessary to delete and re-create the instance.

Recovery operation can be performed on an instance that is part of a streaming replication cluster in "Master" mode. If a recovery operation is performed on a master instance, it will break the replication cluster and streaming replication will stop between the master instance and all its standby instances. In such an event, the standby instances can be promoted to standalone instances or can be deleted and re-created.

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

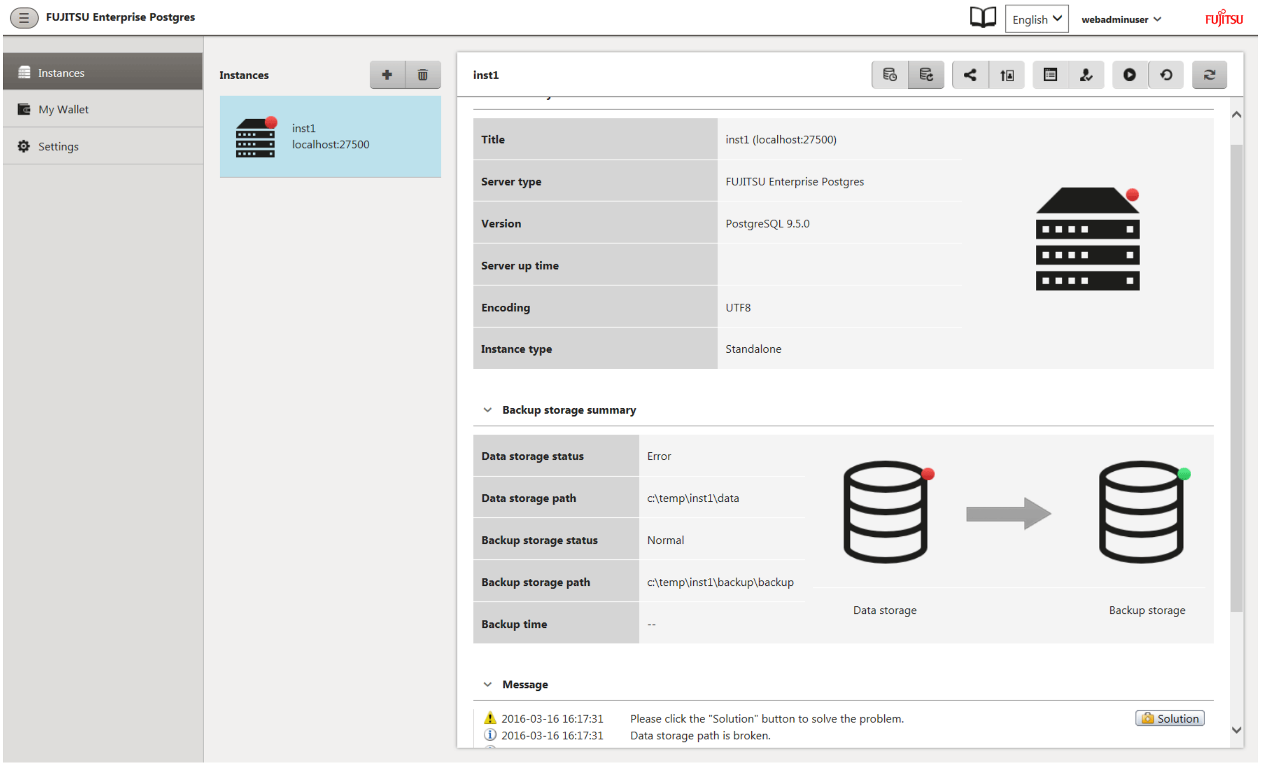
   Do the following if the data in the database has been encrypted:

   - 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.



Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	
Encoding	UTF8
Instance type	Standalone

Backup storage summary

Data storage status	Error
Data storage path	c:\temp\inst1\data
Backup storage status	Normal
Backup storage path	c:\temp\inst1\backup\backup
Backup time	--

Message

⚠ 2016-03-16 16:17:31 Please click the "Solution" button to solve the problem. Solution
ⓘ 2016-03-16 16:17:31 Data storage path is broken.

7. Run recovery

   In the [Restore Instance] dialog box, click [Yes].

   Instance restore is performed. An instance is automatically started when recovery is successful.

   Note

   WebAdmin does not support recovery of hash index. If you are using a hash index, then after recovery, execute the REINDEX command to rebuild it. Use of hash indexes is not recommended.

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 "10.10.3 Other Errors" to perform recovery.

If failure occurred on the backup data storage disk

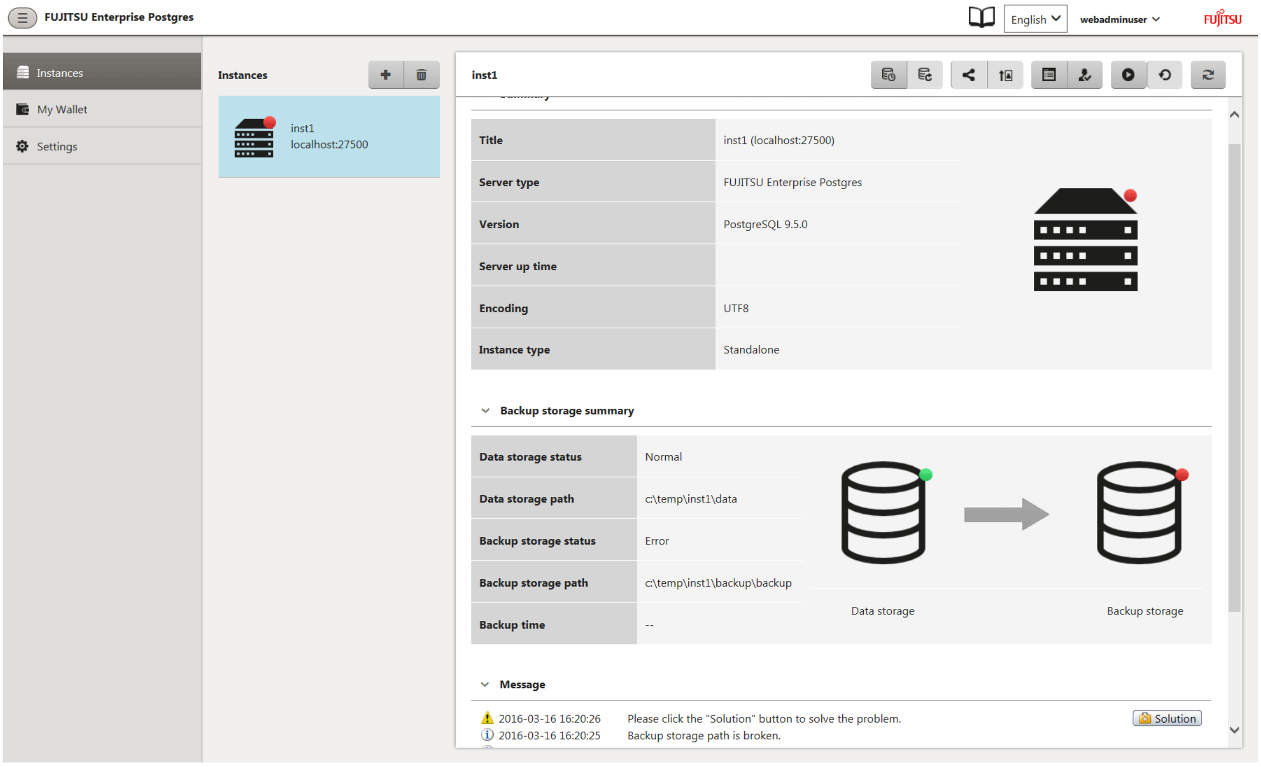
Follow the procedure below to recover the backup data storage disk.

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 [Instance] tab, click [Solution] for the error message.



Title	inst1 (localhost:27500)
Server type	FUJITSU Enterprise Postgres
Version	PostgreSQL 9.5.0
Server up time	
Encoding	UTF8
Instance type	Standalone

Data storage status	Normal
Data storage path	c:\temp\inst1\data
Backup storage status	Error
Backup storage path	c:\temp\inst1\backup\backup
Backup time	--

```
⚠ 2016-03-16 16:20:26   Please click the "Solution" button to solve the problem.
ⓘ 2016-03-16 16:20:25   Backup storage path is broken.
```

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.

10.1.2 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 Commands" for details.

   If the instance fails to stop, refer to "10.11 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.

   In [Properties] in Windows(R) Explorer, set appropriate permissions so that only the instance administrator can access the storage destination directory. (Refer to [Help and Support] in Windows(R) for information on [Properties].)

   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 D:\database\inst1 -B E:\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
   ```

7. Start the instance

   Refer to "2.1.2 Using 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, 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.

   In [Properties] in Windows(R) Explorer, set appropriate permissions so that only the instance administrator can access the backup data storage destination directory. (Refer to [Help and Support] in Windows(R) for information on [Properties].)

   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 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 Commands" for details.

   If the instance fails to stop, refer to "10.11 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.

   In [Properties] in Windows(R) Explorer, set appropriate permissions so that only the instance administrator can access the backup data storage destination directory. (Refer to [Help and Support] in Windows(R) for information on [Properties].)

   Refer to "3.2.2 Using Server Commands" for details.

5. Start the instance

   Start the instance. Refer to "2.1.2 Using 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 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.

10.2 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:

- 10.2.1 Using WebAdmin
- 10.2.2 Using the pgx_rcvall Command

Note

- 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.

10.2.1 Using WebAdmin

If using WebAdmin, recover the data to the point immediately prior to data corruption by using the backup data.

Refer to "10.1.1 Using WebAdmin" for details.

10.2.2 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 Commands" for information on how to stop an instance.

   If the instance fails to stop, refer to "10.11 Actions in Response to Failure to Stop an Instance".

3. Confirm the backup date and time

   Pinpoint a date and time prior to the data corruption based on the content of the job log or event log.

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 pg_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, "May 20, 2015 10:00:00" is specified as the recovery time.

   ```
   > pgx_rcvall -D D:\database\inst1 -B E:\backup\inst1 -e "2015-05-20 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
   ```

6. Start the instance

   Start the instance. Refer to "2.1.2 Using 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.

Note

The pgx_rcvall command cannot accurately recover a hash index. If you are using a hash index, wait for the instance to start and then execute the REINDEX command for the appropriate index.

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.

10.3 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:

- 10.3.1 Using WebAdmin
- 10.3.2 Using the pgx_rcvall Command

Note

- 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.

10.3.1 Using WebAdmin

You can use WebAdmin to recover data to a backup point.

Note

Recovery operation cannot be performed on an instance that is part of a streaming replication cluster in standby mode.

If disk failure occurs on a standby instance, it may be necessary to delete and re-create the instance.

Recovery operation can be performed on an instance that is part of a streaming replication cluster in "Master" mode. If a recovery operation is performed on a master instance, it will break the replication cluster and streaming replication will stop between the master instance and all its 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.

   Note

   WebAdmin cannot accurately recover a hash index. If you are using a hash index, then after recovery, execute the REINDEX command for the appropriate index.

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.

10.3.2 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 Commands" for information on how to stop an instance.

   If the instance fails to stop, refer to "10.11 Actions in Response to Failure to Stop an Instance".

3. Confirm the restore point

   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.

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 pg_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_20150503_1".

```
> pgx_rcvall -D D:\database\inst1 -B E:\backup\inst1 -n batch_20150503_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
```

6. Start the instance

   Start the instance.

   Refer to "2.1.2 Using Commands" for information on how to start an instance.

   Note

   The pgx_rcvall command cannot accurately recover a hash index. If you are using a hash index, wait for the instance to start and then execute the REINDEX command for the appropriate index.

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.

10.4 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 "10.4.1 When using the view (pg_stat_activity)")
- pgAdmin (refer to "10.4.2 Using pgAdmin")

Use the system management function (pg_terminate_backend) to disconnect connections.

10.4.1 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 (9.5.2)
Type "help" for help.
```

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_hostname,pg_terminate_backend(pid) from
pg_stat_activity where 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
```

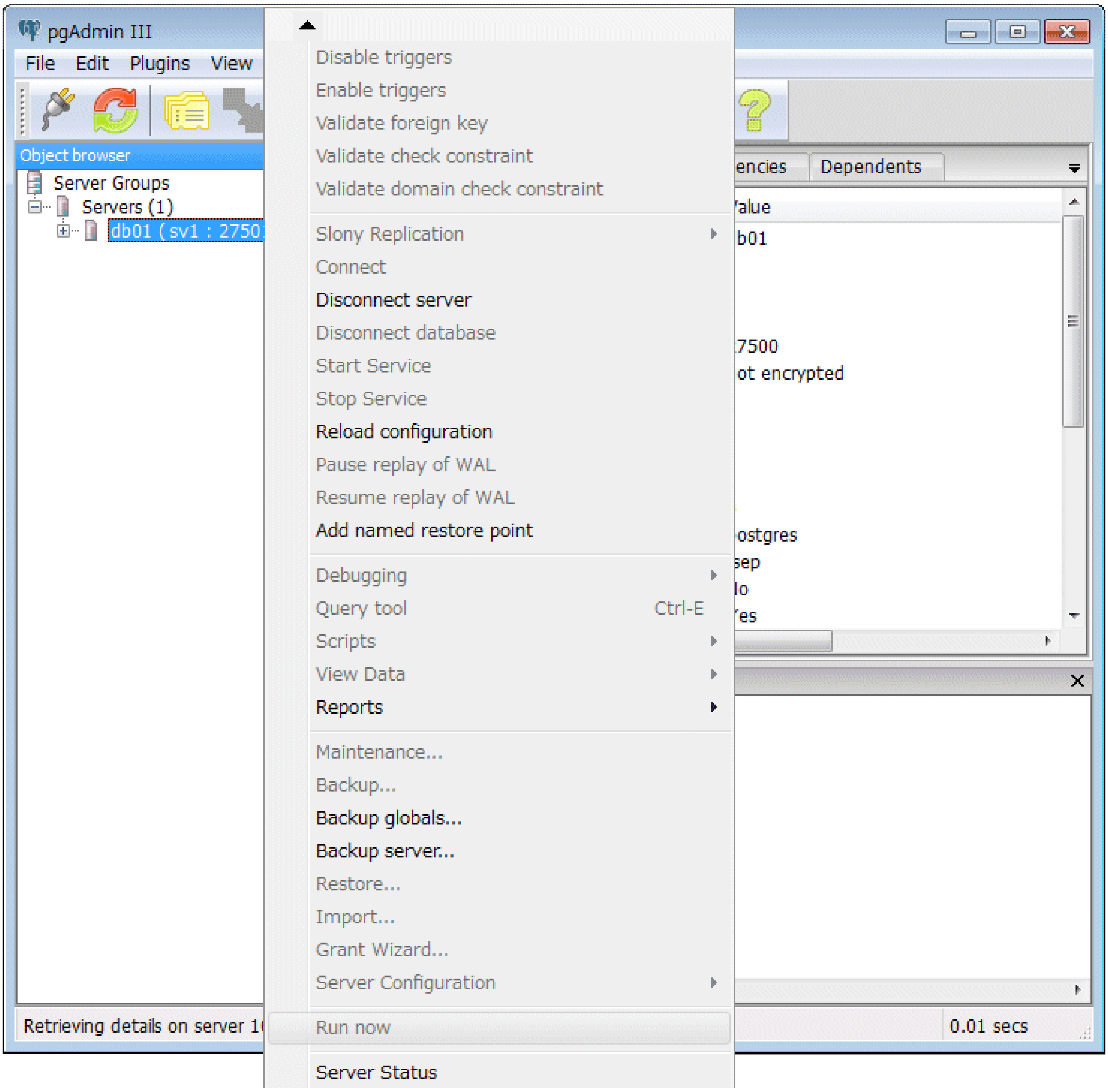
See

- 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.

10.4.2 Using pgAdmin

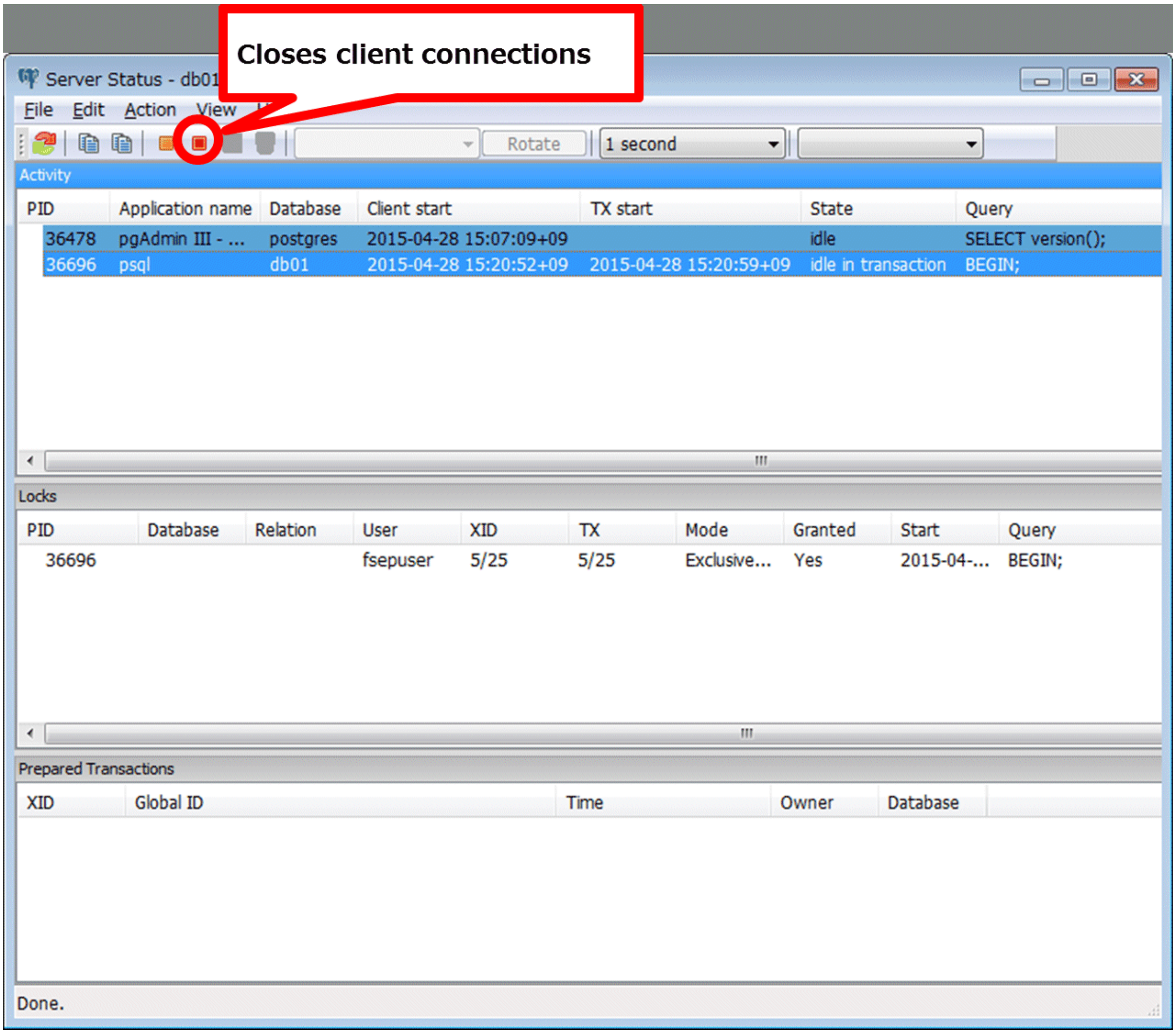
If using pgAdmin, follow the procedure below to close connections.

1. From the [Tools] menu in pgAdmin, click [Server Status].



2. Close client connections that have been in the waiting state for an extended period.

   From the transaction start time displayed under [TX Start], select connections that have been in the waiting state for an extended period. Then click the red square button to close the connections.



Closes client connections

Activity

PID	Application name	Database	Client start	TX start	State	Query
36478	pgAdmin III - ...	postgres	2015-04-28 15:07:09+09		idle	SELECT version();
36696	psql	db01	2015-04-28 15:20:52+09	2015-04-28 15:20:59+09	idle in transaction	BEGIN;

Locks

PID	Database	Relation	User	XID	TX	Mode	Granted	Start	Query
36696			fsepuser	5/25	5/25	Exclusive...	Yes	2015-04-...	BEGIN;

Prepared Transactions

XID	Global ID	Time	Owner	Database

10.5 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 operations.

- 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.

10.6 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:

- 10.6.1 Using a Tablespace
- 10.6.2 Replacing the Disk with a Larger Capacity Disk

10.6.1 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.

10.6.2 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:

- 10.6.2.1 Using WebAdmin
- 10.6.2.2 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.

Note

- Before replacing the disk, stop applications and instances that are using the database.
- 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.

10.6.2.1 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 "10.1.1 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.

10.6.2.2 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 Commands" for information on how to stop an instance.

   If the instance fails to stop, refer to "10.11 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.

   In [Properties] in Windows(R) Explorer, set appropriate permissions so that only the instance administrator can access the data storage destination directory. (Refer to [Help and Support] in Windows(R) for information on [Properties].)

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 D:\database\inst1 -B E:\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
```

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 Commands" for information on how to start an instance.

   Note

   The pgx_rcvall command cannot accurately recover a hash index. If you are using a hash index, wait for the pgx_rcvall command to end and then execute the REINDEX command for the appropriate index.

10. Resume applications

    Resume applications that are using the database.

11. Restore files

    Restore the files backed up in step 1.

10.7 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:

- 10.7.1 Temporarily Saving Backup Data
- 10.7.2 Replacing the Disk with a Larger Capacity Disk

10.7.1 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:

- 10.7.1.1 Using WebAdmin
- 10.7.1.2 Using Server Commands

10.7.1.1 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 (E:\backup\inst1) under F:\mnt\usb\backup.

   Example

   ```
   > mkdir F:\mnt\usb\backup
   > move E:\backup\inst1\* F:\mnt\usb\backup
   ```

   Note: Place the temporary backup destination directory in a location where it will not impact on operating system resources or FUJITSU Enterprise Postgres resources.

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 F:\mnt\usb.

   Example

   ```
   > rmdir /S /Q F:\mnt\usb\backup
   ```

10.7.1.2 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, 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 (E:\backup\inst1) under F:\mnt\usb\backup.

Example

```
> mkdir F:\mnt\usb\backup
> move E:\backup\inst1\* F:\mnt\usb\backup
```

Note: Place the temporary backup destination directory in a location where it will not impact on operating system resources or FUJITSU Enterprise Postgres resources.

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 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 F:\mnt\usb.

   Example

   ```
   > rmdir /S /Q F:\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 Commands" for details.

   If the instance fails to stop, refer to "10.11 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 (E:\backup\inst1) under F:\mnt\usb\backup.

Example

```
> mkdir F:\mnt\usb\backup
> move E:\backup\inst1\* F:\mnt\usb\backup
```

Note: Place the temporary backup destination directory in a location where it will not impact on operating system resources or FUJITSU Enterprise Postgres resources.

4. Start the instance

   Start the instance. Refer to "2.1.2 Using 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 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 F:\mnt\usb.

   Example

   ```
   > rmdir /S /Q F:\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.

10.7.2 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:

- 10.7.2.1 Using WebAdmin
- 10.7.2.2 Using Server Commands

Note

Before replacing the disk, stop applications that are using the database.

10.7.2.1 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 (E:\backup\inst1) under F:\mnt\usb\backup.

   Example

   ```
   > mkdir F:\mnt\usb\backup
   > move E:\backup\inst1\* F:\mnt\usb\backup
   ```

   Note: Place the temporary backup destination directory in a location where it will not impact on operating system resources or FUJITSU Enterprise Postgres resources.

3. Replace with a larger capacity disk

   Replace the disk. Then, recover the volume configuration information.

4. 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 "10.1.1 Using WebAdmin".

5. Restore files

   Restore the files backed up in step 1.

6. 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 F:\mnt\usb.

   Example

   ```
   > rmdir /S /Q F:\mnt\usb\backup
   ```

10.7.2.2 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

No	Step	Instance stopped	
		No	Yes
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 (E:\backup\inst1) under F:\mnt \usb\backup.

   Example

   ```
   > mkdir F:\mnt\usb\backup
   > move E:\backup\inst1\* F:\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, 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.

   In [Properties] in Windows(R) Explorer, set appropriate permissions so that only the instance administrator can access the backup data storage destination directory. (Refer to [Help and Support] in Windows(R) for information on [Properties].)

   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 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 F:\mnt\usb.

Example

```
> rmdir /S /Q F:\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 (E:\backup\inst1) under F:\mnt\usb\backup.

   Example

   ```
   > mkdir F:\mnt\usb\backup
   > move E:\backup\inst1\* F:\mnt\usb\backup
   ```

   Note: Place the temporary backup destination directory in a location where it will not impact on operating system resources or FUJITSU Enterprise Postgres resources.

3. Stop applications

   Stop applications that are using the database.

4. Stop the instance

   Stop the instance. Refer to "2.1.2 Using Commands" for information on how to stop an instance.

   If the instance fails to stop, refer to "10.11 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.

   In [Properties] in Windows(R) Explorer, set appropriate permissions so that only the instance administrator can access the backup data storage destination directory. (Refer to [Help and Support] in Windows(R) for information on [Properties].)

   Refer to "3.2.2 Using Server Commands" for details.

7. Start the instance

   Start the instance. Refer to "2.1.2 Using 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 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 F:\mnt\usb.

    Example

    ```
    > rmdir /S /Q F:\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.

10.8 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.

10.8.1 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:

- 10.8.1.1 Using WebAdmin
- 10.8.1.2 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.

Note

- Before replacing the disk, stop applications that are using the database.
- 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.

10.8.1.1 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 "10.1.1 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.

10.8.1.2 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 Commands" for information on how to stop an instance.

If the instance fails to stop, refer to "10.11 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.

   In [Properties] in Windows(R) Explorer, set appropriate permissions so that only the instance administrator can access the transaction log destination directory. (Refer to [Help and Support] in Windows(R) for information on [Properties].)

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 D:\database\inst1 -B E:\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
```

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 Commands" for information on how to start an instance.

Note

The pgx_rcvall command cannot accurately recover a hash index. If you are using a hash index, wait for the instance to start and then execute the REINDEX command for the appropriate index.

10. Resume applications

    Resume applications that are using the database.

11. Restore files

    Restore the files backed up in step 1.

10.9 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 event 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.

10.10 Actions in Response to Instance Startup Failure

If an instance fails to start, refer to the event 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.

10.10.1 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 event 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

10.10.2 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 "10.14.2 Errors Caused by Power Failure or Mounting Issues", and take actions accordingly.

10.10.3 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 event log and the server log.

There are two methods of recovery:

- 10.10.3.1 Using WebAdmin
- 10.10.3.2 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.

10.10.3.1 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 "10.2.1 Using WebAdmin" for details.

10.10.3.2 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 "10.2.2 Using the pgx_rcvall Command" for details.

10.11 Actions in Response to Failure to Stop an Instance

If an instance fails to stop, refer to the event 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:

- 10.11.1 Using WebAdmin
- 10.11.2 Using Server Commands

10.11.1 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.

10.11.2 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.

10.11.2.1 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 "10.11.2.2 Stopping the Instance Using the Immediate Mode" or "10.11.2.3 Forcibly Stopping the Server Process".

Example

```
> pg_ctl stop -D D:\database\inst1 -m fast
```

10.11.2.2 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 "10.11.2.3 Forcibly Stopping the Server Process".

Example

```
> pg_ctl stop -D D:\database\inst1 -m immediate
```

10.11.2.3 Forcibly Stopping the Server Process

If both the Fast mode and the Immediate mode fail to stop the instance, use the kill parameter of the pg_ctl command to forcibly stop the server process.

The procedure is as follows:

1. Execute the wmic command to identify the process ID of the server process.

   ```
   c:\>wmic
   wmic:root\cli>process where "name = \"postgres.exe\"" get CommandLine,Name,ProcessId
   CommandLine                                                                   Name
   ProcessId
   :
   "C:\Program Files\Fujitsu\fsepv<xy>server64\bin\postgres.exe" -D "D:\database\inst1"
   postgres.exe  896
   :
   ```

   The postgres.exe process ID(896) that indicates the data storage destination directory of the applicable instance in the -D option becomes the server process.

2. Forcibly stop the server process

   As instance manager, forcibly stop the server process using the pg_ctl command.

```
c:\>pg_ctl kill QUIT 896
```

10.12 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 event 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.

Refer to "Deleting Instances" in the Installation and Setup Guide for details on how to delete a instance.

10.13 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 (such as .NET TransactionScope), 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 | 2015-05-06 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
```

10.14 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.

- 10.14.1 Network Error with an External Disk
- 10.14.2 Errors Caused by Power Failure or Mounting Issues

10.14.1 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 event 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.

10.14.2 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 event 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 the existing setting (automatic mounting at the time of starting the operating system) has been changed so that mounting is not performed automatically. In this case, set the mounting to be performed automatically.

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.

- 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.

- 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 parallel scan

- enable_parallelagg (boolean)

  This parameter enables or disables the query planner's use of parallel aggregation plan. If set to "off", parallel aggregation plan will not be used. Normally, use the default value when using the parallel scan feature.

  - Valid values: on or off, 1 or 0

  - Default value: on

- enable_parallelscan (boolean)

  This parameter enables or disables the query planner's use of parallel scan plan. If set to "off", parallel scan plan will not be used. Normally, use the default value when using the parallel scan feature.

  - Valid values: on or off, 1 or 0

  - Default value: on

- max_parallel_degree (4-byte signed integer)

  This parameter specifies the maximum number of parallel processes (background processes) to be used per SQL statement. If set to "0", no parallel processes will be generated, and therefore execution plans will not be able to use parallel scan. When using the parallel scan feature, specify this parameter.

  - Minimum value: 0

  - Maximum value: 8388607

  - Default value: 0

- parallel_scan_pages_threshold (4-byte signed integer)

  If the target table size is equal to the specified threshold (number of pages) or more, the query planner considers the parallel scan plan as an option. Normally, use the default value when using the parallel scan feature.

  - Minimum value: 1

  - Maximum value: 4-byte signed integer

  - Default value: 1000

- parallel_setup_cost (double precision floating point)

  This parameter sets the estimated cost of the planner for starting parallel processes. This value is used to calculate the cost for the parallel scan plan.

  - Minimum value: 0

  - Maximum value: Maximum value of double precision floating point

  - Default value: 1000.0

- parallel_tuple_cost (double precision floating point)

  This parameter sets the estimated cost of the planner for transferring rows from parallel processes to backend processes. This value is used to calculate the cost for the parallel scan plan.

  - Minimum value: 0
  - Maximum value: Maximum value of double precision floating point
  - Default value: 0.1

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.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 / 1024

  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.

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.

- 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.

- vci.smc_directory (string)

  This parameter specifies a directory name in which a temporary file is created as the dynamic shared memory during a scan using a VCI.

  If this parameter is omitted, a directory (*dataStorageDir*\\base\\pgsql_tmp) under the data storage directory will be used.

Note

Note the following when specifying the path:

- Specify \\ as the path delimiter.

- Enclose the path in double quotes (") if it contains spaces.

See

Refer to "Server Configuration" under "Server Administration" in the PostgreSQL Documentation for information on other postgresql.conf parameters.

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*)	void	Opens the keystore
pgx_set_master_key(*passphrase*)	void	Sets the master encryption key
pgx_set_keystore_passphrase(*oldPassphrase*, *newPassphrase*)	void	Changes the keystore passphrase

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.

Only superusers can execute this function. Also, this function cannot be executed within a transaction block.

The pgx_set_master_key function generates a master encryption key and stores it in the 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.

The pgx_set_keystore_passphrase function changes the 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 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.3.1 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    := 'columnName'
[, function_type    := 'FULL'] |
[, function_type    := 'PARTIAL', partialOpt] |
[, function_type    := 'REGEXP', regexpOpt]
)


partialOpt:
function_parameters    := 'maskingFormat'


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    := 'columnName'
)
```

- Change the masking condition

```
pgx_alter_confidential_policy(
commonArg,
action        := 'MODIFY_EXPRESSION',
expression    := 'expression'
)
```

- Change the content of a masking policy set for a masking target

```
pgx_alter_confidential_policy(
commonArg,
action        := 'MODIFY_COLUMN',
column_name    := 'columnName'
[, function_type    := 'FULL'] |
[, function_type    := 'PARTIAL', partialOpt] |
[, function_type    := 'REGEXP', regexpOpt]
)


partialOpt:
function_parameters    := 'maskingFormat'


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    := 'columnName',
column_description    := 'columnDesc'
)
```

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

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		
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.
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.3.2 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        := 'expression'
[, enable            := 'policyStatus']
[, policy_description    := 'policyDesc']
[, column_name        := 'columnName'
     [, function_type    := 'FULL'] |
     [, function_type    := 'PARTIAL', partialOpt] |
     [, function_type    := 'REGEXP', regexpOpt]
     [, column_description    := 'columnDesc']
])


partialOpt:
function_parameters    := 'maskingFormat'


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

Masking type for which an argument can be specified	Argument	Data type	Description	Default value
	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
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.3.3 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.3.4 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	Mandatory

Argument	Data type	Description	Default value
		- 'f': Disabled	

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.3.5 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.4 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.

Appendix C 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.

C.1 pgx_tablespaces

The pgx_tablespaces catalog provides information related to the encryption of tablespaces.

Name	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

C.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 C.1 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

C.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 C.2 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

C.4 pgx_stat_walwriter

The pgx_stat_walwriter view display statistics related to WAL writing, in a single line.

Table C.3 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

C.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 C.4 pgx_stat_sql view

Column	Type	Description
selects	bigint	Number of SELECT statements executed
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

Appendix D Tables Used by Data Masking

This appendix explains tables used by the data masking feature.

D.1 pgx_confidential_columns

This table provides information on masking target for which masking policies are set.

Name	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)
```

D.2 pgx_confidential_policies

This table provides information on masking policies.

Name	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
expression	varchar(1024)	Masking condition
enable	boolean	Masking policy status

Name	Type	Description
		- '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)
```

D.3 pgx_confidential_values

This table provides information on replacement characters when full masking is specified for masking type.

Name	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 E Activating 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 activate the Web server feature of WebAdmin.

This appendix describes how to activate and stop the Web server feature of WebAdmin.

Note that "<*xy*>" in paths indicates the product version and level.

E.1 Activating the Web Server Feature of WebAdmin

Follow the procedure below to activate the Web server feature of WebAdmin:

1. Display the [Services] window

   - Windows Server(R) 2012 or Windows Server(R) 2012 R2:

     In the [Start] screen, select [Administrative Tools], and then click [Services].

   - All other operating systems:

     In the [Start] menu, select [Administrative Tools], and then click [Services].

2. Start a service

   Select the displayed name "FUJITSU Enterprise Postgres WebAdmin *version*", and then click [Start Service].

You can also start a service by specifying the service name of the Web server feature of WebAdmin in the net start command or sc start command.

E.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. Display the [Services] window

   - Windows Server(R) 2012 or Windows Server(R) 2012 R2:

     In the [Start] screen, select [Administrative Tools], and then click [Services].

   - All other operating systems:

     In the [Start] menu, select [Administrative Tools], and then click [Services].

2. Stop a service

   Select the displayed name "FUJITSU Enterprise Postgres WebAdmin *version*", and then click [Stop Service].

You can also stop a service by specifying the service name of the Web server feature of WebAdmin in the net stop command or sc stop command.

Appendix F 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.

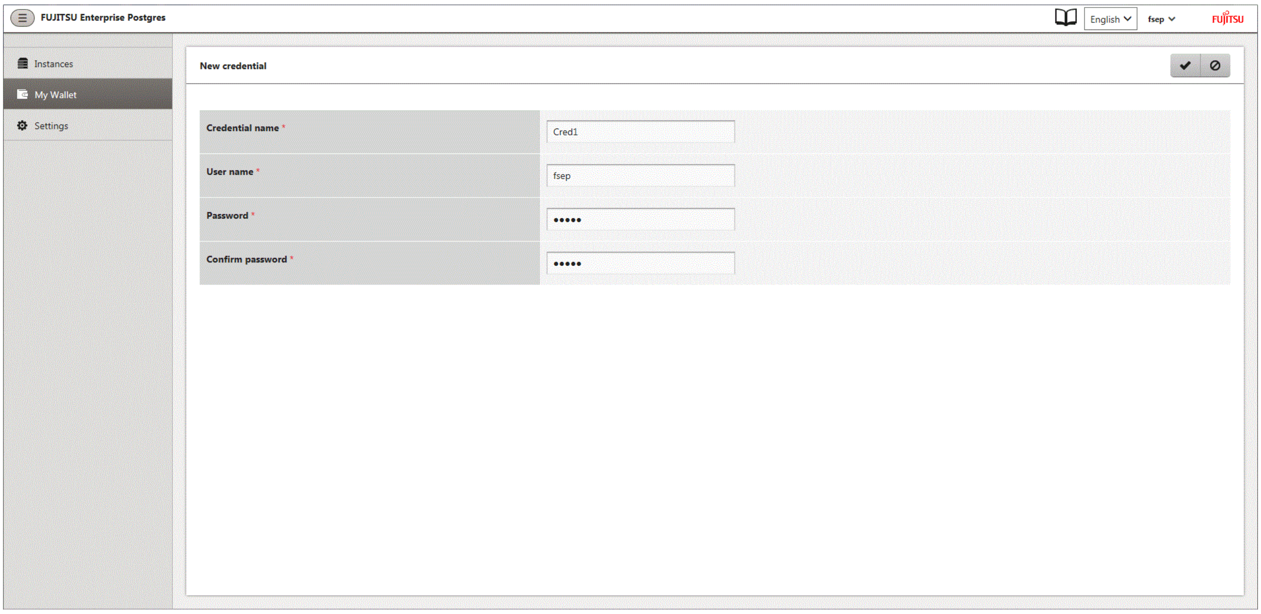
Note

It is not mandatory to create a credential in the Wallet. It is possible to create a remote instance or a standby instance without creating any credential in the Wallet.

If no credential is created beforehand, a user name and password can be entered in the instance creation page. When creating a "Remote" instance, if operating system credentials are entered without using a credential stored in the Wallet, WebAdmin automatically creates a credential with the given user name and password, and stores it in the user's wallet for future use.

F.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 *	Cred1
User name *	fsep
Password *	•••••
Confirm password *	•••••

Enter the following items:

- [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

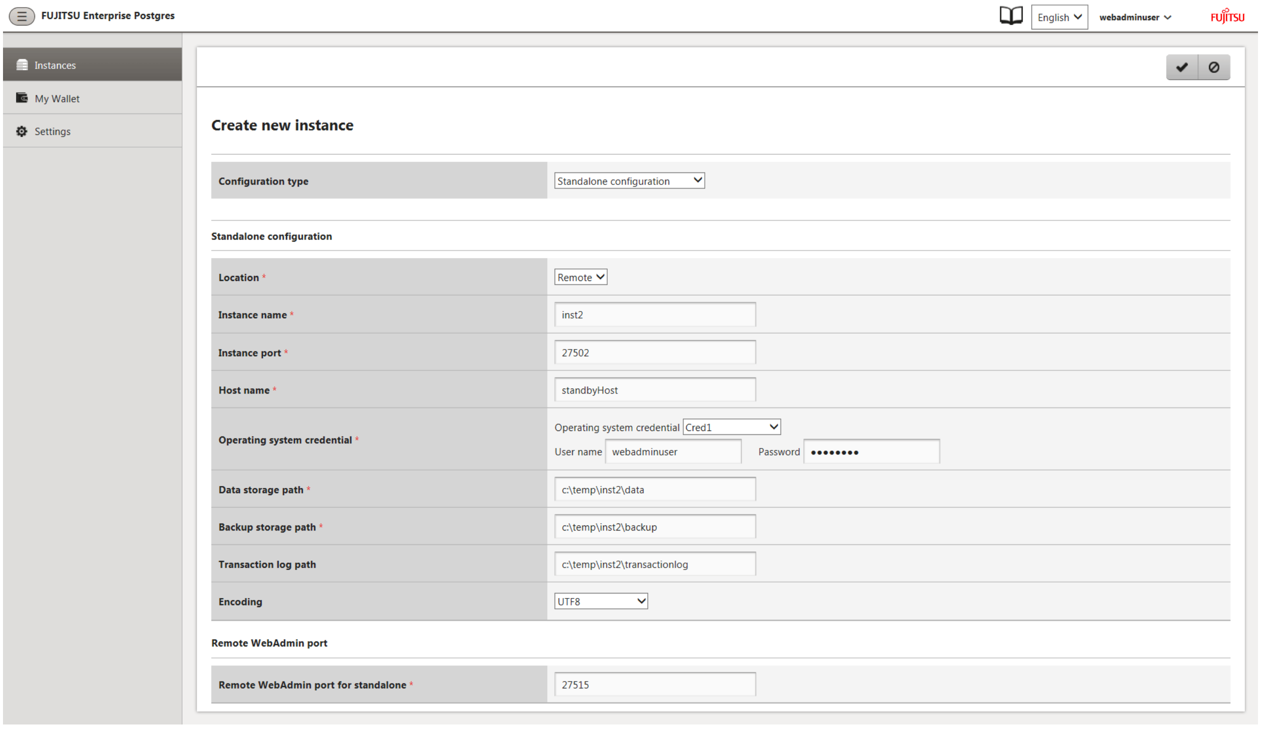
- [Confirm password]: Reenter the password.

3. Click ✓ to store the credential.

F.2 Using a Credential

Once a credential is created in the Wallet, it can be used during remote instance creation or standby instance creation.

The following page uses the credential that was created in the previous section.



Create new instance

Configuration type	Standalone configuration

Standalone configuration

Location *	Remote
Instance name *	inst2
Instance port *	27502
Host name *	standbyHost
Operating system credential *	Operating system credential Cred1 User name webadminuser Password ••••••••
Data storage path *	c:\temp\inst2\data
Backup storage path *	c:\temp\inst2\backup
Transaction log path	c:\temp\inst2\transactionlog
Encoding	UTF8

Remote WebAdmin port

Remote WebAdmin port for standalone *	27515

When "Cred1" is selected in [Operating system credential], the user name and password are automatically populated from the credential.

Appendix G 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 FJQSS (Information Collection Tool) to collect data for initial investigation.

See

Refer to the following manual for information on how to use FJQSS:

- Windows Server(R) 2012 and Windows Server(R) 2012 R2

  In the [Apps] menu, select [FJQSS (Information Collection Tool)], and then click [FJQSS User's Guide].

- Windows Server(R) 2008 R2 or earlier

  In the [Start] menu, select [FJQSS (Information Collection Tool)], and then click [FJQSS User's Guide].

Note

When using FJQSS to collect data for initial investigation, a window will be displayed for you to set the following environment variables:

- PGDATA

  Set the data storage destination.

- PGPORT

  Set the instance port number. This does not need to be set if the default port number (27500) has not been changed.

- PGUSER

  Set the database superuser.
  Set the database superuser so that client authentication is possible.
  FJQSS establishes a TCP/IP connection with the template1 database and collects data from the database.

In addition, when using database multiplexing, set the following environment variables:

- MCCONTROLDIR

  Refer to "Mirroring Controller Resources" in the Cluster Operation Guide for information on the Mirroring Controller management directory.

- The instance administrator user must perform FJQSS operations if using database multiplexing mode.

Index