FUJITSU Enterprise Postgres 14

Security Operation Guide

Windows/Linux

FUJITSU

J2UL-2749-01ENZ0(00)
January-2022

Preface

Purpose of this document

This document describes security when building and operating a FUJITSU Software Enterprise Postgres (hereinafter referred to as "FUJITSU Enterprise Postgres") database system.

Intended readers

This document is intended for those who are:

- Considering installing FUJITSU Enterprise Postgres
- Designing, building, and operating the security operating environment in FUJITSU Enterprise Postgres
- Accessing FUJITSU Enterprise Postgres database systems

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

- Business operations
- FUJITSU Enterprise Postgres
- Linux

Structure of this document

This document is structured as follows:

Chapter 1 Overview of Security

Provides an overview of the security system, and explains the security features provided by FUJITSU Enterprise Postgres.

Chapter 2 Overview of Security Operation

Provides an overview of security operation.

Chapter 3 Tasks of the Manager

Explains the tasks for security measures to be implemented by the manager.

Chapter 4 Tasks of Administrators

Explains the tasks for security measures to be implemented by administrators.

Chapter 5 Tasks of Users

Explains the tasks for security measures to be implemented by users.

Chapter 6 Audit Log Feature

Explains the audit log feature provided by FUJITSU Enterprise Postgres.

References

This document contains abstracts from the following document:

- Database Security Guideline Version 2.0
  (Database Security Consortium (DBSC))

Export restrictions

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

Issue date and version

```
Edition 1.0: January 2022
```

Copyright

Contents

Chapter 1 Overview of Security....1
1.1 What is Security?....1
1.2 Security Requirements....1
1.3 Security Threats....2
1.4 Security Scope....5
1.5 Security Provided by FUJITSU Enterprise Postgres....6
1.5.1 Roles Targeted For Security....6
1.5.2 Security Features....7

Chapter 2 Overview of Security Operation....9
2.1 Security Operation Flow....9

Chapter 3 Tasks of the Manager....11
3.1 Defining Important Information and Risk Analysis....11
3.2 Formulating Account Management Policies....11
3.3 Formulating Log Retrieval Policies....11
3.4 Formulating Rules....12
3.5 Implementing Training....13
3.6 Checking the Database Management Operations....13
3.7 Periodic Diagnosis of the Status of Security Measures....13

Chapter 4 Tasks of Administrators....14
4.1 Receiving Training....14
4.2 Initial Setup....14
4.3 Authentication....15
4.3.1 Managing Accounts....15
4.3.2 Managing Passwords....16
4.3.3 Configuring Connections and Authentication....17
4.4 Access Control....17
4.5 Encryption....18
4.6 Controlling Use of External Media....18
4.7 Security Measures for Servers/Applications....18
4.8 Log Management....19
4.8.1 Retrieving Logs....19
4.8.2 Maintaining Logs....19
4.9 Detecting Unauthorized Access....20
4.10 Analyzing Logs....20

Chapter 5 Tasks of Users....21
5.1 Receiving Training....21
5.2 Managing Accounts/Passwords....21

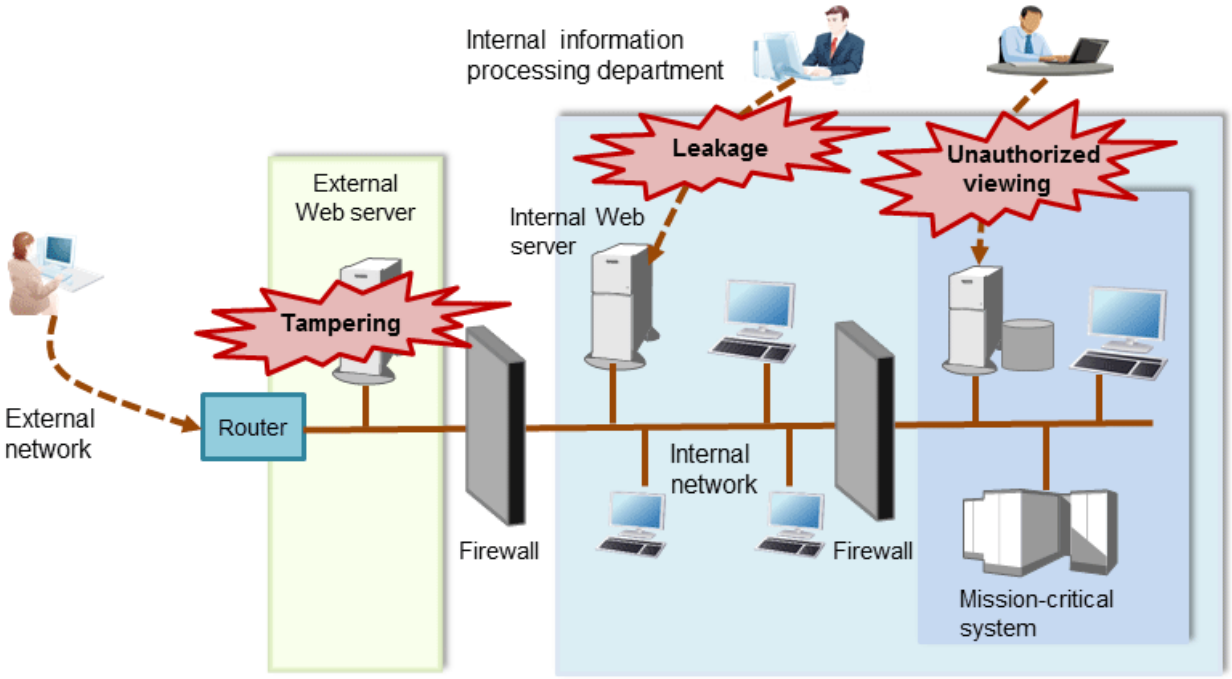
Chapter 6 Audit Log Feature....22
6.1 Audit Log Output Modes....22
6.2 Setup....22
6.3 pgaudit Configuration File....25
6.4 Session Audit Logging....29
6.5 Object Audit Logging....34
6.6 Database Multiplexing....36
6.6.1 Setup....36
6.6.2 Configuring Audit Log Retrieval....37
6.7 Failover....38
6.7.1 Configuring Audit Log Retrieval....38
6.8 View Audit Logs Using SQL....38
6.9 Removing Setup....39

Chapter 1 Overview of Security

1.1 What is Security?

Computer security is the protection of information systems and data from risks such as leakage or tampering of information, attacks, intrusions, eavesdropping from external sources, and interference with information services. Security measures are essential for the advance prevention of security threats in order for information systems to gain trust as social infrastructure.

Figure 1.1 Security threats



The security measures in information systems can be classified as follows:

- Network
- Web
- Application
- Database
- PC

This document focuses on database security measures when using FUJITSU Enterprise Postgres.

1.2 Security Requirements

Below are the necessary security requirements for information systems.

Maintenance of security policies

A security policy clarifies the approach the company should take in relation to information assets, and the actions employees should take.

It is necessary to undertake security of information systems while maintaining security policies.

Integrated security management

Security has the aspects below. It is necessary to manage information in an integrated manner based on these aspects.

Confidentiality

Access to the information is restricted to prevent leakage of information outside of the company

Example measures: Prevention of information leakage or setup of access privileges

Integrity

Integrity is guaranteed, ensuring information does not become corrupted or tampered with

Example measures: Prevention or detection of tampering

Availability

Failure is prevented and normal operation is maintained so that information can be used when needed

Example measures: Power supply measures, system mirroring

1.3 Security Threats

A security threat is defined as something that threatens the confidentiality, integrity, and availability indicated in "1.2 Security Requirements" in respect to information assets. This includes technical threats such as accessing a database, but does not include physical destruction.

Threats are considered to be a combination of type of user who is the source of the threat, information assets that need to be protected, techniques, and unauthorized actions. For example, a threat might be a general user exploiting a database vulnerability to obtain database management information, and then tampering with that information.

When considering security measures, it is firstly necessary to clarify what kind of threats there are. A list of possible threats is shown in the table below. Refer to "Types of user" and "Information assets" for details on the definition of each type of user and information assets that should be protected.

Possible threats

Type of user	Information asset	Technique	Unauthorized action
General user Internal user System manager System developer System administrator System operator	Database management information	Eavesdropping of packets	Unauthorized acquisition (viewing) of information Unauthorized tampering or destruction (updating) of information
		Dictionary attack of passwords	
		Unauthorized acquisition of IDs/passwords through social engineering	
		Unauthorized acquisition of information through misuse of settings	
		Unauthorized acquisition of information through exploiting a database vulnerability	
		Acquisition by an unauthorized route	
General user Internal user	General database information	Acquisition by a normal route	Misuse of information that can be acquired normally (taking data outside of the company)
		SQL issued with the aim of obstructing a job	Obstructing a job (resource depletion)

Type of user	Information asset	Technique	Unauthorized action
General user Internal user	General database information	Eavesdropping of packets	Unauthorized tampering or destruction (updating) of information
		Dictionary attack of passwords	
		Unauthorized acquisition of IDs/passwords through social engineering	
		Unauthorized acquisition of information through exploiting configuration errors	
		Unauthorized acquisition of information through exploiting a database vulnerability	
		Acquisition by an unauthorized route	
System manager System developer System administrator System operator	General database information	Eavesdropping of packets	Unauthorized acquisition (viewing) of information Unauthorized tampering or destruction (updating) of information
		Dictionary attack of passwords	
		Unauthorized acquisition of IDs/passwords through social engineering	
		Unauthorized acquisition of information through exploiting configuration errors	
		Unauthorized acquisition of information through exploiting a database vulnerability	
		Acquisition by an unauthorized route	
System developer	Database management information	Creation of a backdoor	Unauthorized acquisition (viewing) of information Unauthorized tampering or destruction (updating) of information
	General database information		
System manager System administrator	Database management information	Unauthorized acquisition of information by creating an unauthorized database administrator account	Unauthorized acquisition (viewing) of information Unauthorized tampering or destruction (updating) of information
	General database information		
System manager System operator	Database management information	Unauthorized acquisition of information by tampering with database-related files	Unauthorized acquisition (viewing) of information

Type of user	Information asset	Technique	Unauthorized action
	General database information	(definition file, physical file, and so on)	Unauthorized tampering or destruction (updating) of information
Database administrator	Database management information	Misuse of information (taking information outside of the company) after obtaining it through the normal route	Misuse of information that can be acquired normally (taking information outside of the company)
		Unauthorized use of IDs/ passwords from the management information	Tampering with or destroying information that can be acquired
		Unauthorized acquisition of information by tampering with management information	
		SQL issued with the aim of obstructing a job	Obstructing a job (resource depletion)
	General database information	Eavesdropping of packets	Unauthorized acquisition (viewing) of information Unauthorized tampering or destruction (updating) of information
		Misuse of information (taking information outside of the company) after obtaining it through an unauthorized route	
Database operator	Database management information	Eavesdropping of packets	Unauthorized acquisition (viewing) of information Unauthorized tampering or destruction (updating) of information
		Dictionary attack of passwords	
		Unauthorized acquisition of IDs/passwords through social engineering	
		Unauthorized acquisition of information by exploiting configuration errors	
		Unauthorized acquisition of information through exploiting a database vulnerability	
		Acquisition by an unauthorized route	
	General database information	Acquisition by a normal route	Misuse of information that can be acquired normally (taking data outside of the company)
		SQL issued with the aim of obstructing a job	Obstructing a job (resource depletion)

Types of user

In database security, the persons involved with databases and their roles are defined below.

Type of user	Role
System manager	Manages developers, administrators, and operators
System developer	Builds the network around the database server Builds the database server
System administrator	Operates devices of the surrounding database network Operates the database server
System operator	Operates the surrounding database network
Database administrator	Builds the database system Operates the database system
Database operator	Performs business operations
Internal user	End user inside the company
General user	End user outside the company

Information assets

In database security, it is necessary to protect the information assets to be stored on the database server.

Such assets are defined below.

Database management information

- Database configuration information (system catalog, user ID/password, and so on)
- Database logs (such as access logs)

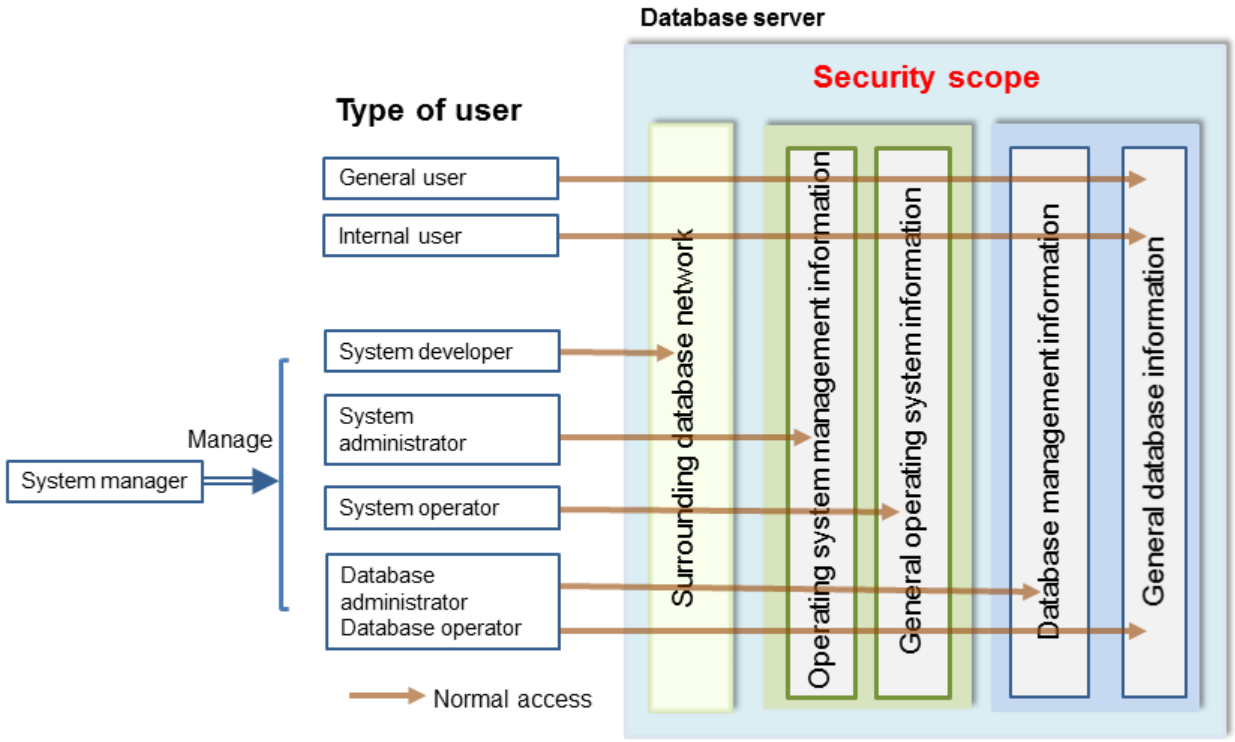
General database information

- Job data
- Applications

1.4 Security Scope

In database systems, both the database server and the surrounding database network are part of the security scope. It is necessary to clarify the extent of the security scope that each type of user is involved with, and consider security measures for the same.

The relationship of the security scope and the types of user is shown below.



1.5 Security Provided by FUJITSU Enterprise Postgres

FUJITSU Enterprise Postgres provides security features that satisfy the security requirements indicated in "1.2 Security Requirements".

This section describes security provided by FUJITSU Enterprise Postgres.

1.5.1 Roles Targeted For Security

In FUJITSU Enterprise Postgres database systems, the roles targeted in relation to security are "Manager", "Administrator", and "User". In order to build a robust security system, it is necessary to put security measures in place for each role.

The roles targeted for security and the mapping of Types of user indicated in "1.3 Security Threats" are shown in the table below.

Role targeted for security	Type of user
Manager	System manager
Administrator	System developer
	System administrator
	System operator
	Database administrator
	Database operator
User	General user
	Internal user

Manager

The manager establishes a security policy and decides on an operations policy for the organization as a whole.

Refer to "Chapter 3 Tasks of the Manager" for details.

Administrator

Administrators design, build and operate a system. While doing this, the administrators must implement the security measures in accordance with the security policy established by the manager.

Refer to "Chapter 4 Tasks of Administrators" for details.

User

A user is a person other than the manager or an administrator who accesses a database. There may be any number of users. It is necessary for users to be registered in the database system, and that access to the database is restricted according to the access privileges.

Refer to "Chapter 5 Tasks of Users" for details.

1.5.2 Security Features

FUJITSU Enterprise Postgres provides the following security features:

- Authentication
- Access control
- Encryption
- Audit log
- Data masking

This section describes each of these features.

Authentication

The databases that can be accessed can be restricted by authenticating the database users who access the database. Additionally, authentication of the server can be performed to prevent spoofing of the database server.

Refer to "Client Authentication" in "Server Administration" in the PostgreSQL Documentation for details on authentication.

Refer to "Secure TCP/IP Connections with SSL" in "Server Setup and Operation" in the PostgreSQL Documentation for details on server authentication.

Access control

Database objects can only be used by the object creator or database user who was specified as the owner when the object was created (both persons are hereinafter referred to as "owner"), or instance administrator, when objects are in their initial state. By having the object owner or instance administrator control access privileges for database users, it is possible to control what kind of tables the database users who connect to the database can access, and what kind of operations they can perform.

Refer to "Privileges" in "The SQL Language" in the PostgreSQL Documentation for details on object access control.

Encryption

FUJITSU Enterprise Postgres provides a transparent data encryption feature that satisfies the requirements below.

- Confidential information can be changed into an unidentifiable state.
- The encryption key and data are managed separately.
- The encryption key is replaced at regular intervals.

PostgreSQL provides an encryption feature called "pgcrypto" that can also be used in FUJITSU Enterprise Postgres, however, it is recommended to use the transparent data encryption features because it will otherwise be necessary to modify the applications that consider encryption. Refer to "Protecting Storage Data Using Transparent Data Encryption" in the Operation Guide for details.

Additionally, if communication data transferred between a client and a server contains confidential information, it is necessary to encrypt the communication data to protect it against threats, such as eavesdropping on the network.

Refer to "Configuring Secure Communication Using Secure Sockets Layer" in the Operation Guide for details on encryption of communication data.

Audit log

A feature that addresses threats such as misuse of administrator privileges, unauthorized access to a database by a user, and other such threats. Information for tracing the processing of administrators and users is retrieved and stored as an audit log.

By periodically viewing and monitoring audit logs, the administrators can detect events that are impacting on the system in some way, or are depleting system resources as a result of incorrect operations by users, and can take appropriate measures to prevent information leakages or system failures in advance.

Refer to "Chapter 6 Audit Log Feature" for details.

Data masking

A feature that changes part of the data to make it available for reference in response to queries issued by an application.

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 without exposing the actual data.

Specifically, the data changed by the data masking feature can be transferred to a test database so that users who perform testing or development can reference the data. During testing, it is desirable to use the data that will be used on a production environment database. However, actual production data should not be used as is for testing because of the risk of leakage of confidential data. This feature enables data that is similar to actual production data to be safely used in test and development environments.

Refer to "Data Masking" in the Operation Guide for details on data masking.

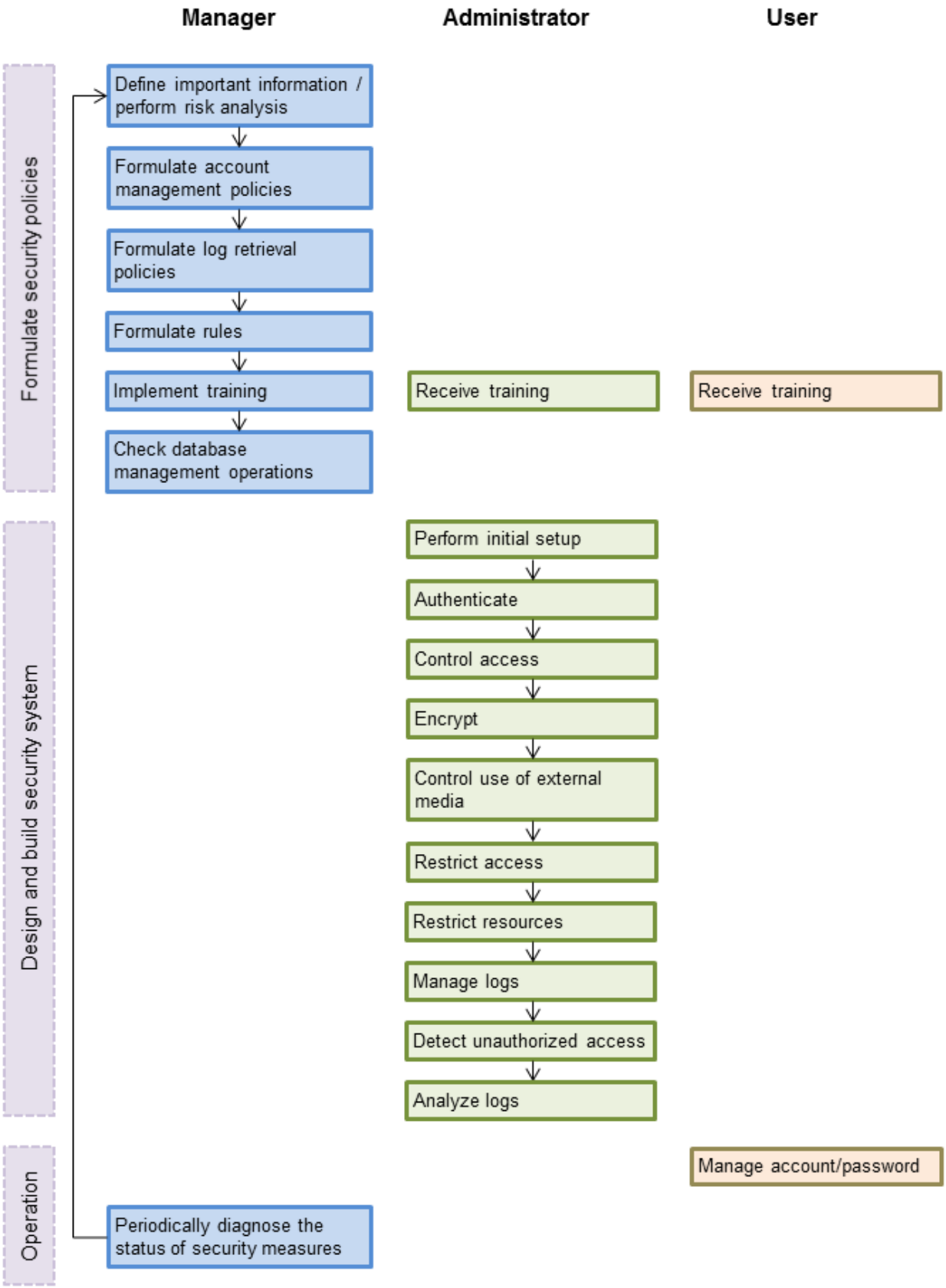
Chapter 2 Overview of Security Operation

2.1 Security Operation Flow

This section shows the flow of work when building a security environment and performing security operation in FUJITSU Enterprise Postgres.

When performing security operation, there are technical measures to be implemented to address security threats by equipping the system with security features, and manual work, such as the implementation of security guidelines, a training system, and the establishment of usage rules.

Figure 2.1 Security operation flow



Chapter 3 Tasks of the Manager

The manager formulates security policies, which become guidelines for security measures.

3.1 Defining Important Information and Risk Analysis

Before formulating security policies, define important information and perform risk analysis. Based on the importance of the information and the result of risk analysis, decide what kind of security measures to put in place.

In defining the important information, identify what should be protected and classify it by importance in order to effectively implement the security measures. Information that should be protected includes "database management information" and "general database information", as indicated in "Information assets". Examples of information classifications are "personal information" and "confidential information".

In the risk analysis, refer to "Possible threats" to identify threats that may arise, and analyze the risks in respect to such threats.

Additionally, by performing a risk analysis once annually as a guide, it is possible to identify threats that may adversely impact the business and related vulnerabilities.

3.2 Formulating Account Management Policies

In formulating an account management policy, implement the following and document the formulated policy.

Organize system users and roles

Identify the necessary roles of the relevant system based on "Types of user". Additionally, organize personnel for each role.

Organize accounts

Organize accounts with the appropriate privileges for each role, and decide on account policies.

- Database administrator account
  - Organize separate accounts for database administrators and database operators
  - Ensure that the database administrator account can only be used by specific persons
  - Perform tasks that do not require database administrator privileges using a separate account without database administrator privileges
- General account

  Create an account for general users by application usage.

Review account management policy

Review the accounts in order to effectively implement security measures.

- Regularly check the accounts mentioned above and their privileges, and determine if they are still appropriate
- If there have been system or operational changes, review the accounts and privileges
- If unsuitable accounts and privileges are discovered, modify them as required

3.3 Formulating Log Retrieval Policies

In formulating a log retrieval policy, implement the following and document the formulated policy.

Organize the purpose of log retrieval

To clarify what logs will be retrieved for, define their reason for retrieval.

Examples of the purpose might include, "To use for investigation in the event of unauthorized access", and "To submit to investigating authorities as evidence if any issues arise".

Decide on the types of logs to be retrieved

In order to retrieve appropriate logs, organize the types of logs that can be retrieved in the target system, and decide on the logs to be retrieved.

Examples of log types are "operating system logs", "application run logs", and "database audit logs".

Organize log retrieval target access

In order to decide on access for log retrieval targets, organize what kind of access will take place.

For example, the following access is possible:

- Access related to important information
  - Access to personal information, confidential information, and database management information
  - Access outside of business hours
  - Login
  - Specific SQL
- Access suspected to be unauthorized
  - Large amount of search access
  - Access from different locations
  - Access outside of business hours

Decide on the log retrieval content

In order to effectively use retrieved logs, organize the required content as a log, and decide on the retrieval content.

For example, the following output content is possible:

- When (time)
- Who (database account, application user)
- What (object ID, table name)
- Where from (machine name, IP address)
- How (SQL type, SQL statement)
- Execution result (success/fail)

Formulate log maintenance policy

In order to use the logs as purposed, formulate the log maintenance policy.

For each log, define its location, storage medium, retention period, access control, and so on.

3.4 Formulating Rules

Formulate the rules that will become the standard for security measures of the target system. Additionally, prescribe penalties for security violations. For example, formulate rules and penalties as below:

- Rules
  - Applying security patches and update programs
  - Prohibiting unauthorized acquisition of information from the database
  - Prohibiting the saving of acquired information to media that is not permitted for use
- Penalties
  - Prescribe penalties in the company's employment policies and procedures

- Set fines

3.5 Implementing Training

In order to have administrators and users recognize the importance and necessity of information security, and to prevent unauthorized access due to operational omissions and mistakes, implement and promote security-related training for administrators and users.

For example, implement promotion of security policies, formulation of training schedules, and formulation of training materials.

3.6 Checking the Database Management Operations

In order to prevent operational errors and unauthorized actions by administrators, implement the measures below:

- Always collect the latest information on security incidents and vulnerabilities related to databases
- Implement management operations only after providing advance notice
- Retain records of management operations

3.7 Periodic Diagnosis of the Status of Security Measures

In order to check if the security measures are effective, periodically diagnose if the security measures have been put in place appropriately based on the security threats.

Additionally, evaluate if the current security measures and policies are effective for the threats and vulnerabilities, and if there are any issues, review the security policies and security measures.

Chapter 4 Tasks of Administrators

Administrators perform the actions below as security measures when designing, building, and operating the system in accordance with the security policies formulated by the manager.

Preparation

- Implement training

Measures to protect against unauthorized behavior

- Perform initial setup
- Authenticate
- Control access
- Encrypt
- Control use of external media
- Restrict access
- Restrict resources

Measures to detect and trace unauthorized behavior

- Manage logs
- Detect unauthorized access
- Analyze logs

4.1 Receiving Training

Administrators receive security-related training in accordance with the training schedule formulated by the manager. Additionally, administrators instruct users to receive training.

4.2 Initial Setup

To minimize database vulnerabilities and the possibility of unauthorized access, implement the security measures below in the initial stage of system building. Additionally, configure the database server so that it primarily operates the database system only.

Making the server more robust

Configure the operating system and network to prevent intrusion into or destruction of a database server, so that the system operates on a secure server.

- Remove unnecessary features or services on the operating system
- Enable only the necessary protocols
- Implement the security features for services, protocols, and daemons considered to have a relatively low security level, such as file sharing and FTP

Installing the latest version

Always download and apply the latest patches in order to reflect the latest security measures.

Installing the minimum necessary features

Install only the necessary features in order to prevent unauthorized use of the system.

Additionally, delete or disable features and services that will not be used.

Changing the port

To prevent unauthorized use of the system, change the default port that is set during installation.

Point

Specify the port during setup of FUJITSU Enterprise Postgres. Refer to the Installation and Setup Guide for Server for details.

Access restrictions for communication features

To prevent unauthorized use of the system using the communication features, implement access restrictions for communication features.

Settings for prohibiting the access path to database configuration files

To prevent database destruction, implement the measures below:

- Restrict users who are permitted to access database configuration files, and periodically review the permissions
- Allow only administrators to access table or definition scripts

Restrictions on the access path to the database

To prevent unauthorized use or operating errors for the database, restrict the distribution range of applications used to access the database only to devices used by users who are permitted access.

Dealing with unauthorized programs

To prevent unauthorized intrusions into a system through a backdoor, such as by tampering with the program source code of an application, document the author of the program to be run and perform checking and testing so that the program will not be tampered with. Additionally, employ safe coding techniques so that issues with general coding vulnerabilities can be addressed.

System security settings

In cases where it is clear that the system security settings will impact security, set reliable security settings in the initial setup stage, such as setting appropriate security parameters.

4.3 Authentication

When accessing a database, authentication must always be performed in order to prevent tampering or information leakage from spoofing by a malicious user.

Password authentication is used when logging on to a database, and the account and password used for authentication are to be strictly managed by administrators.

Additionally, authentication must also be implemented reliably for connections to a database from clients, so that only permitted users can access the database.

4.3.1 Managing Accounts

For account management, perform the actions below.

Create the required accounts

To prevent unauthorized use of accounts, such as spoofing, implement the measures below when creating an account:

- Select the required account
- Specify the user privileges
- Create database administrator accounts and general user accounts separately according to the privileges

Point

Accounts are created using the CREATE ROLE statement. Refer to "CREATE ROLE" in the PostgreSQL Documentation for details.

Delete unnecessary accounts

Remove accounts not used on a daily basis, such as unused accounts and accounts not needed for operations that are created by default during product installation.

Point

Accounts are deleted using the DROP ROLE statement. Refer to "DROP ROLE" in the PostgreSQL Documentation for details.

Set up account lockout

The usage frequency of accounts is to be checked periodically, and if there are any accounts that have not been used for a long period, lock those accounts. Set a limit for failed login attempts, and if this limit is exceeded, lock the account. Additionally, set the period until a locked account is reenabled.

Point

Account locking can be performed by using LDAP authentication. Refer to "LDAP Authentication" in the PostgreSQL Documentation for details.

Manage database administrator accounts

Manage database administrator accounts in accordance with the account management policy formulated by the manager.

Manage development environment and production environment accounts

To prevent unauthorized use of accounts used in a development environment, delete accounts used in the development environment before operation starts in the production environment. In cases where it is unavoidable to use an account used in the development environment in the production environment, use different passwords in each environment.

Set up a temporary use account

If a temporary user will use the system, either provide a shared account with a temporary password for each use, or create a temporary account.

4.3.2 Managing Passwords

Manage passwords as below.

Make strong passwords

The use of account passwords that can easily be guessed by others, such as a password that matches the ID, or the default password provided during installation, is prohibited. Set a complex and strong password.

Change passwords regularly

Change passwords regularly to prevent others from accessing the account in case the password is obtained by unauthorized means. Additionally, configure the settings to force a password change when prompted after the first use.

Set the password expiry period

To encourage regular changing of passwords, set a password expiry period.

Point

Password setting and changing is specified using the CERATE ROLE statement or ALTER ROLE statement. Refer to "CREATE ROLE" and "ALTER ROLE" in the PostgreSQL Documentation for details.

Additionally, by using passwordcheck and LDAP authentication, the actions below can be performed:

- The default password set during installation can be changed
- The password expiry period can be set

- The number and types of characters used for the password can be checked

Refer to "passwordcheck" and "LDAP Authentication" in the PostgreSQL Documentation for details.

4.3.3 Configuring Connections and Authentication

Configure connections and authentication so that the database can only be accessed by permitted users.

Point

Client authentication is configured in pg_hba.conf. Refer to "Client Authentication" in the PostgreSQL Documentation for details.

4.4 Access Control

If appropriate access privileges are not set for administrators and users, security incidents may occur, such as information leakage resulting from access to information by an unauthorized person. To minimize such incidents, it is necessary to implement the security measures below for the access privileges and perform rule-based access control.

Point

Notes when setting access privileges

- The creation of a special account that allows granting of privileges to all users is prohibited
- The creation of a general account that allows access to general information such as operations data is prohibited

Identifying the database access requirements

To set the appropriate access privileges for each usage purpose for the database, follow the procedure below to identify the access requirements:

1. Classify the usage purpose of the account, such as "For database management", "For object management", and "For data access".
2. Classify the required privileges for each usage purpose, such as "By feature" and "By object".
3. Categorize the accounts based on each privilege.
4. Identify the minimum necessary range of data and minimum necessary access content (view, update, create, delete) to be accessed for each categorized account, and decide on the database access requirements.

Setting the access privileges

Assign the minimum necessary privileges based on the database access requirements for each categorized account. Additionally, restrict accounts when assigning administrator privileges.

Reviewing access privileges

To reflect changes in access requirements in the system, periodically review the access privileges and check if there are any access privileges that are no longer needed. If any unnecessary access privileges have been set, promptly modify the access privileges.

Point

Access privileges are set using the GRANT statement or REVOKE statement. Refer to "GRANT" and "REVOKE" in the PostgreSQL Documentation for details.

4.5 Encryption

To prevent unauthorized usage of data in the event information leakage occurs due to data theft, eavesdropping of communication, and other such activities, implement the encryption measures below.

Encrypt communication

To protect data from eavesdropping over the network between a database server and clients, use the encryption feature to encrypt communications.

Refer to "Configuring Secure Communication Using Secure Sockets Layer" in the Operation Guide for details.

Encrypt data

To protect data from theft, use the encryption feature to encrypt the data. The data below is targeted for encryption:

- Data to be stored on the database
- Backup data
- Data files

Refer to "Protecting Storage Data Using Transparent Data Encryption" in the Operation Guide for details.

Manage encryption keys

Restrict the persons who can access the encryption key to a minimum number of database administrators.

Additionally, to ensure the encrypted information will not be easily decrypted, create a mechanism for appropriately managing the encryption key for the entire life cycle (generation, distribution, saving, and disposal), and strictly manage the encryption key.

Refer to "Configuring Secure Communication Using Secure Sockets Layer" and "Protecting Storage Data Using Transparent Data Encryption" in the Operation Guide for details.

4.6 Controlling Use of External Media

Information leakage can be prevented by controlling use of external media (such as CD/DVD, USB drive, and external hard disk) and PCs that are connected to the database, and restricting the removal of data from the database.

Restricting connection of external media

Remove external media and printers that will not be used in operations, and restrict connection of external media to which information may be written.

Restricting use of external media

Restrict connections for external media and printers to control the writing of information to these devices.

Controlling use of connected PCs

Prevent leakage of information from PCs connected to the database:

- Limit connections of external media to PCs
- Implement security measures to make the PC robust
- Implement individual authentication for access from the PC
- Manage the installed software and monitor the software usage status
- Limit connections to printers

4.7 Security Measures for Servers/Applications

An even more robust security system can be achieved by strengthening security for servers and applications in addition to the security measures for databases. Implement the security measures below for servers and applications:

Restrict access

Implement the measures below and restrict access to the database server:

- Install the database server inside the firewall to prevent direct access to the database server from many unspecified PCs.
- In the local network, implement measures such as using the router to restrict IP addresses, and restrict PCs and segments that can directly access the database server.

Restrict resources

Restrict excessive use of CPU resources by general users to prevent the disruption of service and extraction of large amounts of data.

4.8 Log Management

Logs are a feature that addresses threats such as misuse of administrator privileges, and unauthorized access to a database by a user. Information for investigating/tracing processes and operations performed for the database is retrieved and managed as logs for identifying the cause in the event information leakage or unauthorized access occurs.

FUJITSU Enterprise Postgres provides the audit log feature for retrieving and managing logs. Refer to "Chapter 6 Audit Log Feature" for details.

This section describes the information that should be obtained as logs and how to maintain logs, as a measure for managing information leakage and unauthorized access.

4.8.1 Retrieving Logs

The audit logs below are retrieved in accordance with the log retrieval policy formulated by the manager.

Login information

Retrieves logs during login and logout.

Database access information (view/update)

Retrieves all access relating to the information below:

- General database information (such as personal information and confidential information used in the business)
- Database management information (system catalog, user ID/password, and so on)

Changed information of database objects

Retrieves logs related to creating, changing, and deleting database objects such as database accounts and tables.

Operation logs for audit logs

To prevent suppression of retrieved audit logs, operations such as initialization of audit logs, and stoppage of the audit log feature are retrieved as logs.

4.8.2 Maintaining Logs

Logs are maintained in accordance with the log retrieval policy formulated by the manager.

Storing logs

Perform the actions below and store logs securely so that the retrieved logs will not be updated by others:

- Save logs to external media, and store the external media in a secure location, such as lockable storage
- Restrict the viewing of logs to administrators only, and set access restrictions for logs, such as not assigning update rights
- Decide on the log retention period, with consideration to cases where investigation tracing back to the time of discovery of an issue is required

Preventing tampering of logs

Implement measures to prevent tampering of logs, such as retaining multiple copies of logs and using storage that cannot be rewritten.

Encrypting logs

Encrypt logs so that logs are not easily viewed.

4.9 Detecting Unauthorized Access

To address unauthorized access, it is necessary to establish a mechanism for detecting unauthorized access to databases and monitor access.

Communicating unauthorized access

Create a mechanism that notifies of detected unauthorized access, such as notifying the manager and the administrator, if an account lock occurs due to the limit for failed login attempts being exceeded.

Checking access times

Create a mechanism that can check for suspicious access to the information below outside of normal access hours, together with implementing measures to address such access.

Detecting access to database management information

- Monitor logs and detect access during timeframes that have not been applied for
- In the event a request for access permission outside of normal access hours is made, the log is checked for any discrepancies in the requested content and work result

Detecting access to general database information

- Decide on the timeframes during which access to the database is permitted for each general account
- Detect access outside of normal access hours from session information logs

Checking the connection source where access is not permitted

To detect access from connection sources that are not permitted, define the sources from where access is permitted, and detect access from connection sources that are not permitted.

Define the access patterns (connection source, operating system user and account) of database administrator accounts and general accounts, and check for access outside of these patterns.

4.10 Analyzing Logs

Create a mechanism that analyzes logs to detect unauthorized behavior in cases where information leakage, unauthorized access, or other such activity, is suspected. Analyses should include those shown below.

Periodic analysis of session information

Analyze session information of logs from the perspectives below to detect unauthorized logins:

- Trend of sessions with a large number of failed login attempts
- Trend of sessions with accounts that are logged in for long periods of time
- Trend of sessions in which a large amount of resources are used

Periodic analysis of database access information

Analyze SQL statements from the perspectives below to detect unauthorized access to the database:

- Trend of SQL being executed over a long period of time
- Trend of SQL using a large amount of resources

Chapter 5 Tasks of Users

The user performs the actions below as security measures when using the system.

5.1 Receiving Training

The user must receive security-related training as instructed by the manager or the administrator to learn about security. By having users with a common awareness relating to security, an even more stable security system can be established.

5.2 Managing Accounts/Passwords

Users can use the database system by using the account and password provided by the administrator. At such times, the user is to implement the measures below so that the account and password are not misused by others:

- Be responsible for managing the ID and password in a manner that ensures the account does not become locked during login
- Change the password regularly
- Comply with the expiry period set for the password by promptly changing the password when it is about to expire

Chapter 6 Audit Log Feature

In PostgreSQL, logs output as server logs can be used as audit logs by using the log output feature. There are, however, logs that cannot be analyzed properly, such as SQL runtime logs, which do not output the schema name. Additionally, because the output conditions cannot be specified in detail, log volumes can be large, which may lead to deterioration in performance.

The audit log feature of FUJITSU Enterprise Postgres enables retrieval of details relating to database access as an audit log by extending the feature to pgaudit. Additionally, audit logs can be output to a dedicated log file or server log. This enables efficient and accurate log monitoring.

Note

The audit log feature cannot be used if PostgreSQL is running in single-user mode.

6.1 Audit Log Output Modes

In pgaudit, the two types of audit log below can be output.

Session Audit Logging

Session Audit Logging outputs information related to SQL executed in backend processes (processes generated when connection requests are received from clients), information related to starting and connecting databases, and information related to errors, as a log. In Session Audit Logging, by specifying the log output conditions and filtering the logs to be output, performance degradation due to outputting large volumes of logs can be prevented.

Refer to "6.4 Session Audit Logging" for details.

Object Audit Logging

When SELECT, INSERT, UPDATE, and DELETE are executed for specific objects (tables, columns), Object Audit Logging outputs these as a log. TRUNCATE is not supported. Object Audit Logging outputs object operations for which privileges have been assigned to specified roles, as a log. Object Audit Logging can control log output at an even finer level of granularity than Session Audit Logging.

Refer to "6.5 Object Audit Logging" for details.

Information

Depending on the application or command, FUJITSU Enterprise Postgres may execute SQL internally and the audit logs may be retrieved.

Also, the audit logs of multiple SQLs with the same statement ID may be retrieved. This is because before the user executes the SQL, another SQL is executed internally by FUJITSU Enterprise Postgres.

6.2 Setup

This section describes the setup method of pgaudit.

1. Copy the pgaudit files

L

As superuser, run the following command. Note that "<*x*>" in paths indicates the product version.

```
$ su -
Password:******
# cp -r /opt/fsepv<x>server64/OSS/pgaudit/* /opt/fsepv<x>server64
```

W

Open a command prompt as administrator privileges, run the following command. Note that "<*x*>" in paths indicates the product version.

```
> xcopy /E "c:\Program Files\Fujitsu\fsepv<x>server64\OSS\pgaudit\*"  "c:\Program
Files\Fujitsu\fsepv<x>server64"
```

2. Create the pgaudit configuration file

   Create the pgaudit configuration file, which describes the information required for pgaudit actions. Create the file using the same encoding as used for the database.

   In addition, set write permissions for the database administrator only in the pgaudit configuration file so that policies related to the audit log are not viewed by unintended users.

   Refer to "6.3 pgaudit Configuration File" for details.

   Note

   Do not define the rule section in the pgaudit configuration file at this point.

   Example of a pgaudit configuration file

```
[output]
logger = 'auditlog'
```

3. Configure postgresql.conf

   Configure the parameters below in postgresql.conf to use audit logs:

   shared_preload_libraries

   Specify "pgaudit".

   pgaudit.config_file

   Specify the deployment destination path of the pgaudit configuration file.

   If a relative path is specified, the path will be relative to the data storage directory.

   log_replication_commands

   Specify "on".

   log_min_messages

   Check if "ERROR" or higher has been specified.

   If outputting an audit log to a server log ("serverlog" is specified in the logger parameter of the pgaudit configuration file), check the parameters below relating to server logs.

   logging_collector

   Check if "on" has been specified.

   log_destination

   Check if "stderr" has been specified.

L

   log_file_mode

   Check if the server log permissions are appropriate, so that only the permitted persons can access it.

   Information

   The default for the log_file_mode parameter is 0600, which only allows the database administrator to have access.

   For example, to permit other members of the group to which the database administrator belongs to view the audit logs, specify 0640 for log_file_mode.

Example

```
log_file_mode = 0640
```

The database administrator can also be prevented from viewing audit logs by specifying 0000. However, write privileges are assigned for outputting logs.

If outputting an audit log to a dedicated log file ("auditlog" is specified in the logger parameter of the pgaudit configuration file), check the parameter below.

max_worker_processes

If the max_worker_processes parameter has been set, add 1 to the specified value.

See

Refer to "Error Reporting and Logging" in the PostgreSQL Documentation for details on server logs.

If using database multiplexing, refer to "6.6 Database Multiplexing" for details.

Example of postgresql.conf

In the example below, only the parameters that need to be configured when using the audit log feature are described.

```
shared_preload_libraries = 'pgaudit'
pgaudit.config_file = 'pgaudit.conf'
log_replication_commands = on
log_min_messages = WARNING
```

4. Start the instance

   Start the instance and check if the message below is output.

```
LOG: pgaudit extension initialized
```

5. Create the pgaudit extension

   Execute CREATE EXTENSION to create the pgaudit extension.

```
$ psql
=# CREATE EXTENSION pgaudit;
=# \dx
                  List of installed extensions
Name    | Version |   Schema   |     Description
--------+---------+------------+---------------------------------
pgaudit | 1.0     | public     | provides auditing functionality
plpgsql | 1.0     | pg_catalog | PL/pgSQL procedural language
(2 rows)
```

6. Configure the parameters in the pgaudit configuration file

   Add or change the parameters in the pgaudit configuration file as required.

   Refer to "6.3 pgaudit Configuration File" for details.

7. Restart the instance

   Restart the instance to apply the changes to the pgaudit configuration file. After restarting, check if the changes have been reflected correctly.

Linux

```
LOG:  log_catalog = 1
LOG:  log_level_string =
LOG:  log_level = 15
LOG:  log_parameter = 0
LOG:  log_statement_once = 0
LOG:  role =
LOG:  logger = auditlog
LOG:  log_directory = pgaudit_log
LOG:  log_filename = pgaudit-%Y-%m-%d_%H%M%S.log
LOG:  log_file_mode = 0600
LOG:  log_rotation_age = 1440
LOG:  log_rotation_size = 10240
LOG:  log_truncate_on_rotation = 0
LOG:  fifo_directory = /tmp
LOG:  Rule 0
LOG:  pgaudit extension initialized
```

Windows

```
LOG:  log_catalog = 1
LOG:  log_level_string =
LOG:  log_level = 15
LOG:  log_parameter = 0
LOG:  log_statement_once = 0
LOG:  role =
LOG:  logger = auditlog
LOG:  log_directory = pgaudit_log
LOG:  log_filename = pgaudit-%Y-%m-%d_%H%M%S.log
LOG:  log_file_mode = 0600
LOG:  log_rotation_age = 1440
LOG:  log_rotation_size = 10240
LOG:  log_truncate_on_rotation = 0
LOG:  Rule 0
LOG:  pgaudit extension initialized
```

6.3 pgaudit Configuration File

In the pgaudit configuration file, specify the information required for pgaudit actions. The pgaudit configuration file comprises three sections: "output section", "option section", and "rule section".

output section

The output section is specified using the format below:

- *paramName* = '*value*'

The valid parameters in the output section are shown in the table below.

Linux

Parameter name	Description	Remarks
logger	Dedicated log file (auditlog)/*serverLog* (serverlog) that will be the output destination of the audit log The default is "auditlog" (dedicated log file).	The dedicated log file is output using the same encoding as used for the database.
log_directory	Directory where the audit log is to be created Specify the full path or the relative path from the data storage directory.	Enabled only if "auditlog" is

Parameter name	Description	Remarks
	The default is "pgaudit_log".	specified for the logger parameter
log_filename	File name of the audit log Specify a file name that varies according to the time, in the same manner as for log_filename in the postgresql.conf file. The default is "pgaudit-%Y-%m-%d_%H%M%S.log".	Enabled only if "auditlog" is specified for the logger parameter
log_file_mode	Specify the permissions of the audit log so that only permitted persons can access it. The parameter value is the numeric mode specified in the format permitted in chmod and umask system calls. The default is "0600". Refer to "log_file_mode" in "6.2 Setup" for information on audit log file permissions.	Enabled only if "auditlog" is specified for the logger parameter
log_rotation_age	Maximum age of the audit log file A new audit log file is generated when the time (minute units) specified here elapses. To disable generation of new log files based on time, specify "0". The valid units are "min" (minutes), "h" (hours), and "d" (days). If the unit is omitted, "min" will be used. The default is "1d" (1 day).	Enabled only if "auditlog" is specified for the logger parameter
log_rotation_size	Maximum size of the audit log file A new log file will be generated after logs of the size specified here are output to a log file. To disable generation of new log files based on size, specify "0". The valid units are "kB" (kilobytes), "MB" (megabytes), and "GB" (gigabytes). If the unit is omitted, "kB" will be used. The default is "10MB".	Enabled only if "auditlog" is specified for the logger parameter
log_truncate_on_rotation	If rotating audit log files based on time, this parameter is used to specify whether to overwrite (on)/not overwrite (off) existing audit log files of the same name. For example, if "on" is specified, and "pgaudit-%H.log" is specified for log_filename, 24 separate log files will be generated based on time, and those files will be cyclically overwritten. The default is "off". If "off" is specified, the logs will be written to the existing audit log files.	Enabled only if "auditlog" is specified for the logger parameter
fifo_directory	FIFO (named pipe) directory to be used between the daemon process that outputs audit log files and the backend process FIFO named p.PGAUDIT.*nnnn* (*nnnn* is the postmaster PID) are created in the fifo_directories directory. The files cannot be deleted manually. The default is "/tmp".	Enabled only if "auditlog" is specified for the logger parameter

Windows

Parameter name	Description	Remarks
logger	Dedicated log file (auditlog)/*serverLog* (serverlog) that will be the output destination of the audit log The default is "auditlog" (dedicated log file).	The dedicated log file is output using the same encoding as used for the database.
log_directory	Directory where the audit log is to be created Specify the full path or the relative path from the data storage directory. The default is "pgaudit_log".	Enabled only if "auditlog" is specified for the logger parameter
log_filename	File name of the audit log Specify a file name that varies according to the time, in the same manner as for log_filename in the postgresql.conf file. The default is "pgaudit-%Y-%m-%d_%H%M%S.log".	Enabled only if "auditlog" is specified for the logger parameter
log_file_mode	This parameter is ignored in Windows.	
log_rotation_age	Maximum age of the audit log file A new audit log file is generated when the time (minute units) specified here elapses. To disable generation of new log files based on time, specify "0". The valid units are "min" (minutes), "h" (hours), and "d" (days). If the unit is omitted, "min" will be used. The default is "1d" (1 day).	Enabled only if "auditlog" is specified for the logger parameter
log_rotation_size	Maximum size of the audit log file A new log file will be generated after logs of the size specified here are output to a log file. To disable generation of new log files based on size, specify "0". The valid units are "kB" (kilobytes), "MB" (megabytes), and "GB" (gigabytes). If the unit is omitted, "kB" will be used. The default is "10MB".	Enabled only if "auditlog" is specified for the logger parameter
log_truncate_on_rotation	If rotating audit log files based on time, this parameter is used to specify whether to overwrite (on)/not overwrite (off) existing audit log files of the same name. For example, if "on" is specified, and "pgaudit-%H.log" is specified for log_filename, 24 separate log files will be generated based on time, and those files will be cyclically overwritten. The default is "off". If "off" is specified, the logs will be written to the existing audit log files.	Enabled only if "auditlog" is specified for the logger parameter

Information

If the logger parameter is set to "serverlog", audit logs will be output to the server log as log messages, therefore the status information and message severity level according to the log_line_prefix parameter in postgresql.conf will be output to the beginning of the audit log.

If the logger parameter is omitted or set to "auditlog", audit logs will be output to a dedicated log file as dedicated logs, therefore the status information and message severity level according to the log_line_prefix parameter in the postgresql.conf file will not be output.

Refer to "Output format" in "6.4 Session Audit Logging"or "Output format" in "6.5 Object Audit Logging" for information on the output format of audit logs.

L

Point

The pgaudit log_file_mode configuration parameter setting is separate from, and unaffected by, the log_file_mode GUC parameter setting and the -g/-allow-group-access initdb option.

When using a dedicated pgaudit log file, since the pgaudit log_directory location defaults to inside the data storage directory, it is possible for the pgaudit log_file_mode permissions to conflict with the intended file permissions specified by the -g/-allow-group-access initdb option. In this case, the pgaudit log_directory should be specified to be a directory located outside of the data storage directory.

W

Point

If the file permissions for the server log do not satisfy the security requirements of the system, specify a dedicated log file for the output destination of the audit log, and set the access permissions for the output destination directory so that the security requirements are satisfied. At that time, assign read and write permissions to the database administrator.

option section

The option section is specified using the format below:

- *paramName* = '*value*'

The valid parameters in the option section are shown in the table below.

Parameter name	Description	Remarks
role	Name of roles used in Object Audit Logging If specifying a name containing uppercase characters, key words, multibyte characters and commas, enclose the name in double quotation marks.	Parameter used in Object Audit Logging only
log_catalog	Whether to enable (on)/disable (off) log output for pg_catalog If pgAdmin, and so on, will not be retrieving audit logs that access pg_catalog, specify "off". The default is "on" (enabled).	
log_parameter	Whether to enable (on)/disable (off) output of values passed by parameters in SQL execution The default is "off" (disabled).	
log_statement_once	Whether to control (on)/not control (off) output for the second and subsequent SQL statements if the same SQL statement is the log output target The default is "off" (do not control).	

Parameter name	Description	Remarks
log_level	Log level of audit logs The valid values are "DEBUG5", "DEBUG4", "DEBUG3", "DEBUG2", "DEBUG1", "INFO", "NOTICE", "WARNING", and "LOG". The default is "LOG".	Enabled only if "serverlog" is specified for the logger parameter

rule section

The rule section is used in Session Audit Logging. Refer to "6.4 Session Audit Logging" for details.

Note

Do not specify the rule section if the role parameter has been specified in the option section. If you specify the rule section, the audit logs of Object Audit Logging and Session Audit Logging will be output intermingled and you will be unable to view the logs in CSV format.

6.4 Session Audit Logging

In Session Audit Logging, specify the rules for filtering logs to be output in the rule section in the pgaudit configuration file.

Rules are specified using the formats below. Multiple values can be specified, using a comma as the delimiter.

- *paramName* = '*value*'
- *paramName* != '*value*'

If [rule] is described on its own in the rule section with no parameters specified, all audit logs will be output.

Example

```
[output]
logger = 'auditlog'
[rule]
```

If the rule section is not described ([rule] is not described), the audit logs will not be output.

Example

```
[output]
logger = 'auditlog'
```

The valid parameters in the rule section are shown in the table below.

Parameter name	Description	Example
timestamp	Timestamp range Refer to "timestamp" for details on how to specify timestamps.	timestamp = '09:00:00 - 10:00:00, 18:00:00 - 18:30:00'
database	Database name To specify a blank value, use a pair of double quotation marks (""). When specifying a name containing uppercase characters, key words, multibyte	database = 'prodcut_db'

Parameter name	Description	Example
	characters and commas, enclose the name in double quotation marks.	
audit_role	Role name To specify a blank value, specify use a pair of double quotation marks (""). When specifying a name containing uppercase characters, key words, multibyte characters and commas, enclose the name in double quotation marks.	audit_role = 'appuser1'
class	Operation class Select from the values below. Multiple values can be specified. Refer to "class" for details on the meaning of each class. - BACKUP - CONNECT - DDL - ERROR - FUNCTION - MISC - READ - ROLE - WRITE - SYSTEM	class = 'READ, WRITE'
object_type	Object type This parameter is enabled when the class parameter is "READ" and "WRITE". Select from the values below. Multiple values can be specified. - TABLE - INDEX - SEQUENCE - TOAST_VALUE - VIEW - MATERIALIZED_VIEW - COMPOSITE_TYPE - FOREIGN_TABLE - FUNCTION	object_type = 'TABLE, INDEX'
object_name	Object name This parameter is enabled when the class parameter is "READ" and "WRITE".	object_name = 'myschema.tbl1' object_name = 'myschema.tbl1, "*mySchema*.TABLE"'

Parameter name	Description	Example
	For objects that can be modified by a schema, such as a table, modify such objects by schema name. When specifying a name containing uppercase characters, key words, multibyte characters and commas, enclose the name in double quotation marks. When specifying the object name "*schemaName.tableName*", enclose the entire object name modified by schema name in double quotation marks. To specify a blank value, use a pair of double quotation marks ("").	
application_name	Application name To specify a blank value, use a pair of double quotation marks ("").	application_name = 'myapp'
remote_host	Connection destination host name or IP address If "on" is specified for the log_hostname parameter in the postgresql.conf file, specify a host name. Otherwise, specify the IP address. If using a local host, specify "[local]". To specify a blank value, use a pair of double quotation marks ("").	remote_host = 'ap_server'

timestamp

Specify a timestamp range from "*startTime*" to "*endTime*" for the log output target. The timestamp format is 'hh:mm:dd-hh:mm:dd' (hh is expressed in 24-hour notation, and hh, mm, and dd are expressed in two-digit notation).

The start time must be earlier than the end time. If specifying multiple ranges, specify each start and end timestamp using a comma as the delimiter.

End timestamps consider milliseconds. For example, if '11:00:00 - 11:59:59' is specified for the timestamp, "11:00:00:000" to "11:59:59:999" will be the target range.

The timestamps used by evaluation in the rule section of pgaudit are different to the timestamps issued in the log entries. That is because log entries are output after evaluation by pgaudit, with the timestamp being generated at that time.

class

The meaning of each class specified in the class parameter is below:

- READ: SELECT, COPY FROM
- WRITE: INSERT, UPDATE, DELETE, TRUNCATE, COPY TO
- FUNCTION: Function call, DO
- ROLE: GRANT, REVOKE, CREATE ROLE, ALTER ROLE, DROP ROLE
- DDL: All DDLs (such as CREATE and ALTER) other than the DDLs of the ROLE class
- CONNECT: Events relating to connecting (request, authenticate, and disconnect)
- SYSTEM: Instance start, promotion to primary server
- BACKUP: pg_basebackup

- ERROR: Event completed by an error (PostgreSQL error codes other than 00). This class can be used if ERROR or lower level is specified for the log_min_messages parameter in postgresql.conf.
- MISC: Other commands (such as DISCARD, FETCH, CHECKPOINT, and VACUUM)

Evaluation of the rule section

- When a log event occurs, all expressions in the rule section are evaluated at once. Log entries are only output if all parameters in the rule section are evaluated as being true.

  For example, if the rule below has been set, of the operations performed by 'apserver' to 'myschema.tbl1', the operations applicable to classes other than 'WRITE" in the period from 10 a.m. to 11 a.m. will be output as audit logs.

  ```
  [rule]
  timestamp = '10:00:00-11:00:00'
  remote_host = 'apserver'
  object_name = 'myschema.tbl1'
  class != 'WRITE'
  ```

- Multiple rule sections can be defined in the pgaudit configuration file. Log events are evaluated using each rule section, and an audit log is output for each matching rule section.

  For example, if the rules below are set, duplicated audit logs will be output.

  ```
  [rule]
  object_name = 'myschema.tbl1'
  [rule]
  object_name = 'myschema.tbl1'
  ```

- If the same parameter is specified multiple times in one rule section, the last specified parameter is effective.

  For example, if the rule below has been set, "object_name = 'myschema.tbl3'" will take effect.

  ```
  [rule]
  object_name = 'myschema.tbl1'
  object_name = 'myschema.tbl2'
  object_name = 'myschema.tbl3'
  ```

Output format

In Session Audit Logging, audit logs are output in the format below:

```
AUDIT: SESSION,READ,2020-03-12 19:00:58 PDT,
   (1)         (2)     (3)
[local],19944,psql,appuser,postgres,2/8, 2, 1,SELECT,,TABLE,myschema.account, ,
  (4)    (5)   (6)   (7)      (8)   (9)(10)(11)(12)(13)(14)    (15)        (16)
SELECT * FROM myschema.account;,<not logged>
       (17)                       (18)
```

No	Content
(1)	Log header Fixed as "AUDIT: SESSION".
(2)	Class
(3)	SQL start time
(4)	Remote host name If using a local host, [local] is output.
(5)	Backend process ID

No	Content
(6)	Application name If an application name is not specified, [unknown] is output.
(7)	User name
(8)	Database name
(9)	Virtual transaction ID
(10)	Statement ID
(11)	Substatement ID
(12)	Command tag
(13)	SQLSTATE
(14)	Object type
(15)	Object name
(16)	Error message
(17)	SQL If the SQL contains a password, such as for CREATE ROLE, and so on, it will be replaced with "<REDACTED>". Additionally, if "on" is specified for the log_statement_once parameter of the option section in the pgaudit configuration file, "<previously logged>" is output for the second and subsequent statements.
(18)	Depending on the log_parameter parameter value of the option section in the pgaudit configuration file, the output content will be as below. - log_parameter=on is specified If parameters are specified in the SQL, the parameters are concatenated and output, using a space as the delimiter. If parameters are not specified in the SQL, "<none>" is output. - log_parameter=off (default) is specified "<not logged>" is output. Additionally, if "on" is specified for the log_statement_once parameter of the option section in the pgaudit configuration file, "<previously logged>" is output for the second and subsequent statements.

Information

If accessing resources that use the features below, the command tag (12) may be output as "???".

- INSTEAD OF trigger
- RULE
- VIEW
- Security policy per row
- Table inheritance

Example

Below is an example of retrieving audit logs in Session Audit Logging.

1. Settings

In the pgaudit configuration file, specify the rule section below.

```
[rule]
class = 'READ, WRITE'
object_name = 'myschema.account'
```

2. Retrieving logs

Execute the SQL below from the client.

```
CREATE TABLE myschema.account
(
    id int,
    name text,
    password text,
    description text
);
INSERT INTO myschema.account (id, name, password, description) VALUES (1, 'user1',
'HASH1', 'blah, blah');
SELECT * FROM myschema.account;
```

The audit log below can be retrieved.

'DDL' is not defined in the class parameter, so CREATE TABLE is not output as an audit log.

```
AUDIT: SESSION,WRITE,2020-03-12 19:00:49 PDT,[local],19944,psql,appuser,postgres,
2/7,1,1,INSERT,,TABLE,myschema.account,,"INSERT INTO myschema.account (id, name,
password, description) VALUES (1, 'user1', 'HASH1', 'blah, blah');",<not logged>
AUDIT: SESSION,READ,2020-03-12 19:00:58 PDT,[local],19944,psql,appuser,postgres,
2/8,2,1,SELECT,,TABLE,myschema.account,,SELECT * FROM myschema.account;,<not logged>
```

6.5 Object Audit Logging

In Object Audit Logging, retrieval of audit logs is achieved by using roles.

Roles are specified in the role parameter of the option section to retrieve audit logs. If there are privileges for commands executed by a role, or if privileges have been inherited from another role, those command operations are output as audit logs.

For example, after "auditor" is set for the role parameter of the option section, the SELECT and DELETE privileges for the account table are assigned to "auditor". In this case, when SELECT or DELETE is executed for the account table, audit logs are output.

Output format

In Object Audit Logging, audit logs are output in the format below:

```
AUDIT: OBJECT,1,1,READ,SELECT,TABLE,public.account,SELECT password FROM account;,<not
logged>
     (1)    (2)(3)(4)  (5)    (6)        (7)              (8)                    (9)
```

No	Content
(1)	Log header Fixed as "AUDIT: OBJECT".
(2)	Statement ID
(3)	Substatement ID
(4)	Class name

No	Content
(5)	Command tag
(6)	Object type
(7)	Object name
(8)	SQL If "on" is specified for the log_statement_once parameter of the option section in the pgaudit configuration file, "<previously logged>" is output for the second and subsequent statements.
(9)	Depending on the log_parameter parameter value of the option section in the pgaudit configuration file, the output content will be as below. - When log_parameter=on If parameters are specified in the SQL, the parameters are concatenated and output, using a comma as the delimiter. If parameters are not specified in the SQL, "<none>" is output. - When log_parameter=off (default) "<not logged>" is output. Additionally, if "on" is specified for the log_statement_once parameter of the option section in the pgaudit configuration file, "<previously logged>" is output for the second and subsequent statements.

Information

If accessing resources that use the features below, the command tag (5) may be output as "???".

- INSTEAD OF trigger
- RULE
- VIEW
- Security policy per row
- Table inheritance

Example

Below is an example of retrieving logs in Object Audit Logging.

By setting the target for assigning privileges to roles in detail, log output can be controlled.

In the example below, log retrieval of the account table is controlled by the privileges assigned to the columns, however, log retrieval of the account_role_map table is controlled by the privileges assigned to the table.

1. Settings

The role parameter below is specified for the option section in the pgaudit configuration file.

```
[option]
role = 'auditor'
```

2. Defining a role

A role is defined for Object Audit Logging.

```
CREATE USER auditor NOSUPERUSER LOGIN;
```

3. Retrieving logs

Execute the SQL below from the client.

```
CREATE TABLE account
(
    id int,
    name text,
    password text,
    description text
);
GRANT SELECT (password) ON public.account TO auditor;
SELECT id, name FROM account;
SELECT password FROM account;
GRANT UPDATE (name, password) ON public.account TO auditor;
UPDATE account SET description = 'yada, yada';
UPDATE account SET password = 'HASH2';
CREATE TABLE account_role_map
(
    account_id int,
    role_id int
);
GRANT SELECT ON public.account_role_map TO auditor;
SELECT account.password, account_role_map.role_id
  FROM account
  INNER JOIN account_role_map ON account.id = account_role_map.account_id;
```

The audit log below can be retrieved.

In the account table, only the operations for columns that privileges have been assigned to are output as logs.

In the account_role_map table, privileges are assigned to the table, so operations performed for the table are output as logs.

```
AUDIT: OBJECT,4,1,READ,SELECT,TABLE,public.account,SELECT password FROM account;,<not
logged>
AUDIT: OBJECT,7,1,WRITE,UPDATE,TABLE,public.account,UPDATE account SET password =
'HASH2';,<not logged>
AUDIT: OBJECT,10,1,READ,SELECT,TABLE,public.account,"SELECT account.password,
account_role_map.role_id
        FROM account
        INNER JOIN account_role_map ON account.id = account_role_map.account_id;",<not
logged>
AUDIT: OBJECT,10,1,READ,SELECT,TABLE,public.account_role_map,"SELECT account.password,
account_role_map.role_id
        FROM account
        INNER JOIN account_role_map ON account.id = account_role_map.account_id;",<not
logged>
```

6.6 Database Multiplexing

This section describes audit log retrieval while in database multiplexing mode.

6.6.1 Setup

If setting up the audit log feature in a database multiplexing environment that has already been built, follow the procedure below.

1. Copy the pgaudit files

   Copy the pgaudit files on the primary server and standby server.

   Refer to step 1 in "6.2 Setup" for details on copying the pgaudit files.

2. Create the pgaudit configuration file

   Create the pgaudit configuration file on the primary server. Copy the pgaudit configuration file you created to the standby server.

   Refer to step 2 in "6.2 Setup" for details on creating the pgaudit configuration file.

3. Configure postgresql.conf

   In the postgresql.conf file on the primary server and standby server, configure the parameters for using audit logs. Set the same values for the parameters.

   Refer to step 3 in "6.2 Setup" and "6.6.2 Configuring Audit Log Retrieval" for details on the parameters to configure.

4. Configure the *serverIdentifier*.conf file of Mirroring Controller

   In the *serverIdentifier*.conf file on the primary server and standby server, configure the parameters for using audit logs.

   Refer to "6.6.2 Configuring Audit Log Retrieval" for details on the parameters to be set.

5. Start the instance

   Start the instance of the primary server and standby server.

6. Create the pgaudit extension

   Execute CREATE EXTENSION on the primary server to create a pgaudit extension.

   Refer to step 5 in "6.2 Setup" for details on creating pgaudit extensions.

7. Configure the parameters in the pgaudit configuration file

   Add/change the parameters of the pgaudit configuration file on the primary server. Copy the pgaudit configuration file with the added/changed parameters to the standby server.

   Refer to "6.3 pgaudit Configuration File" and "6.6.2 Configuring Audit Log Retrieval" for details on the parameters to set.

8. Restart the instance

   Restart the instance of the primary server and standby server.

6.6.2 Configuring Audit Log Retrieval

In database multiplexing mode, Mirroring Controller periodically accesses the database to check the multiplexing status and detect failure. Due to this, audit logs are also periodically retrieved, so log files become used up. Therefore, set the parameters below so that audit logs are not retrieved by Mirroring Controller.

postgresql.conf

log_connections

Omit, or specify "off".

log_disconnections

Omit, or specify "off".

serverIdentifier.conf file of Mirroring Controller

target_db

Specify "template1".

Note

If creating a new database, create it after stopping Mirroring Controller, or specify a name other than "template1" for the template database.

pgaudit configuration file

rule section database

Specify database != 'template1'.

6.7 Failover

This section explains audit log retrieval performed by the failover feature integrated with the cluster software.

6.7.1 Configuring Audit Log Retrieval

The failover feature accesses the database periodically to check the instance status. Due to this, audit logs are also periodically retrieved, so log files become depleted. Therefore, set the parameters below so that audit logs are not retrieved.

postgresql.conf

log_connections

Omit, or specify "off".

log_disconnections

Omit, or specify "off".

pgaudit configuration file

rule section database

Specify database != 'template1'.

6.8 View Audit Logs Using SQL

By using file_fdw of an additional module, audit logs can be accessed using SQL. This section describes how to view audit logs, using Session Audit Logging output to a dedicated log file as an example.

1. Install file_fdw

   Execute CREATE EXTENSION to install file_fdw as an extension.

```
$ psql
=# CREATE EXTENSION file_fdw;
=# \dx
                  List of installed extensions
Name     | Version |   Schema   |     Description
---------+---------+------------+-----------------------------------------
file_fdw | 1.0     | public     | foreign-data wrapper for flat file access
pgaudit  | 1.0     | public     | provides auditing functionality
plpgsql  | 1.0     | pg_catalog | PL/pgSQL procedural language
(3 rows)
```

2. Create an external server

   Use CREATE SERVER to create an external server managed by file_fdw.

```
$ psql
=# CREATE SERVER auditlog FOREIGN DATA WRAPPER file_fdw;
```

3. Create an audit log table.

   Use CREATE FOREIGN TABLE to define the table columns of audit logs, CSV file name and format.

```
$ psql
=# CREATE FOREIGN TABLE auditlog (
header text,
class text,
sql_start_time timestamp with time zone,
remote_host_name text,
backend_process_id text,
application_name text,
session_user_name text,
database_name text,
virtual_transaction_id text,
statement_id text,
substatement_id text,
command_tag text,
sqlstate text,
object_type text,
object_name text,
error_message text,
sql text,
parameter text
) SERVER auditlog
OPTIONS ( filename '/database/inst1/pgaudit_log/pgaudit-2020-03-12.log', format
'csv' );
```

If an audit log file is rotated and multiple audit log files exist, it is necessary to create a table for each audit log file.

4. View audit logs

   Use SELECT and view the audit logs.

```
$ psql
=# SELECT * FROM auditlog;
     header     |  class  |     sql_start_time     | remote_host_name |
backend_process_id ...
----------------+---------+------------------------+------------------
+------------------ ...
AUDIT: SESSION  | WRITE   | 2020-03-12 19:00:49+09 | [local]          |
19944              ...
AUDIT: SESSION  | READ    | 2020-03-12 19:00:58+09 | [local]          |
19944              ...
```

6.9 Removing Setup

This section describes how to remove the setup of pgaudit.

1. Start the instance
2. Remove the pgaudit extension

   Execute DROP EXTENSION to remove the pgaudit extension from the database.

```
$ psql -d <database name>
=# DROP EXTENSION pgaudit;
=# \q
```

3. Change postgresql.conf

   Remove the parameters below relating to pgaudit.

   - shared_preload_libraries
   - pgaudit.config_file

4. Restart the instance
5. Remove the pgaudit files

L

As superuser, run the following command. Note that "<*x*>" in paths indicates the product version.

```
$ su -
Password:******
# rm -rf /opt/fsepv<x>server64/filesCopiedDuringSetup
```

Information

The files copied during setup can be checked below.

```
# find /opt/fsepv<x>server64/OSS/pgaudit
```

W

Open a command prompt as administrator privileges, run the following command. Note that "<*x*>" in paths indicates the product version.

```
> del "c:\Program Files\Fujitsu\fsepv<x>server64\filesCopiedDuringSetup"
```

Information

The files copied during setup can be checked below.

```
> dir /b /s "c:\Program Files\Fujitsu\fsepv<x>server64\OSS\pgaudit"
```