FUJITSU

Fujitsu
Enterprise Postgres 17

Cluster Operation Guide (Database Multiplexing)

Linux

J2UL-2987-01ZLZ0(00)
November 2024

Preface

Purpose of this document

This document describes the tasks required for using the database multiplexing feature of Fujitsu Enterprise Postgres.

Intended readers

This document is intended for those who set up and use the database multiplexing feature.

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

- PostgreSQL
- SQL
- Linux

Structure of this document

This document is structured as follows:

Chapter 1 Overview of Database Multiplexing Mode

Provides an overview of database multiplexing mode.

Chapter 2 Setting Up Database Multiplexing Mode

Describes how to set up database multiplexing mode.

Chapter 3 Operations in Database Multiplexing Mode

Explains periodic database multiplexing mode.

Chapter 4 Action Required when an Error Occurs in Database Multiplexing Mode

Explains the action required when an error occurs during a database multiplexing mode.

Chapter 5 Managing Mirroring Controller Using WebAdmin

Explains how to set up and manage Mirroring Controller in a streaming replication cluster using WebAdmin.

Appendix A Parameters

Explains the configuration files and parameters required for database multiplexing mode.

Appendix B Supplementary Information on Building the Primary Server and Standby Server on the Same Server

Explains supplementary information on building the primary server and standby server on the same server.

Appendix C User Commands

Explains the user commands.

Appendix D Notes on Performing Automatic Degradation Immediately after a Heartbeat Abnormality

Provides notes when performing automatic degradation unconditionally after a heartbeat abnormality is detected during heartbeat monitoring of an operating system or server.

Appendix E WebAdmin Disallow User Inputs Containing Hazardous Characters

Explains characters not allowed in WebAdmin.

Appendix F Collecting Failure Investigation Data

Explains how to collect data for initial investigation.

Export restrictions

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

Issue date and version

```
Edition 1.0: November 2024
```

Copyright

Contents

Chapter 1 Overview of Database Multiplexing Mode

This chapter provides an overview of database multiplexing mode.

Point

In this and subsequent chapters, the word "Mirroring Controller" may be used in the process or management directory name or explanation.

1.1 What is Database Multiplexing Mode

Database multiplexing mode is an operation mode (log shipping mode) based on PostgreSQL streaming replication. Other software such as cluster software is not required.

This mode replicates the database on all servers that comprise the cluster system. It achieves this by transferring the updated transaction logs of the database from the server that receives the updates (primary server) to another server (standby server), and then reflecting them on the standby server. The client driver automatically distinguishes between the primary and standby servers, so applications can be connected transparently regardless of the physical server.

It consists of a feature that detects faults in the elements that are essential for the continuity of the database operation (such as the database process, disk, and network), as well as simplified switchover and standby server disconnection features. Furthermore, referencing can be performed on the standby server. The database will be copied in synchronous mode.

Note

If using WebAdmin or Mirroring Controller, Fujitsu Enterprise Postgres supports cluster systems comprising one primary server and one standby server.

- Although it is possible to connect an asynchronous standby server to the cluster system as an additional server, the standby server is not targeted for monitoring by Mirroring Controller.
- A synchronous standby server cannot be connected to the cluster system as an additional server.

See

The streaming replication feature is not described in this manual.

Refer to "High Availability, Load Balancing, and Replication" in the PostgreSQL Documentation for information on the streaming replication feature.

Figure 1.1 Failover from the primary server to the standby server

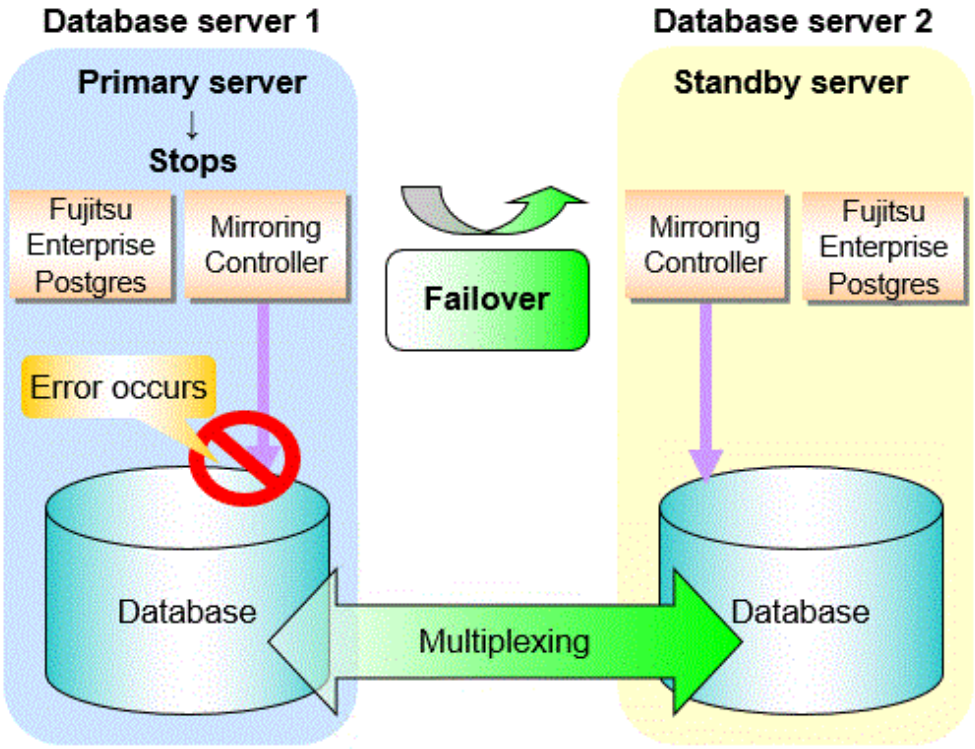
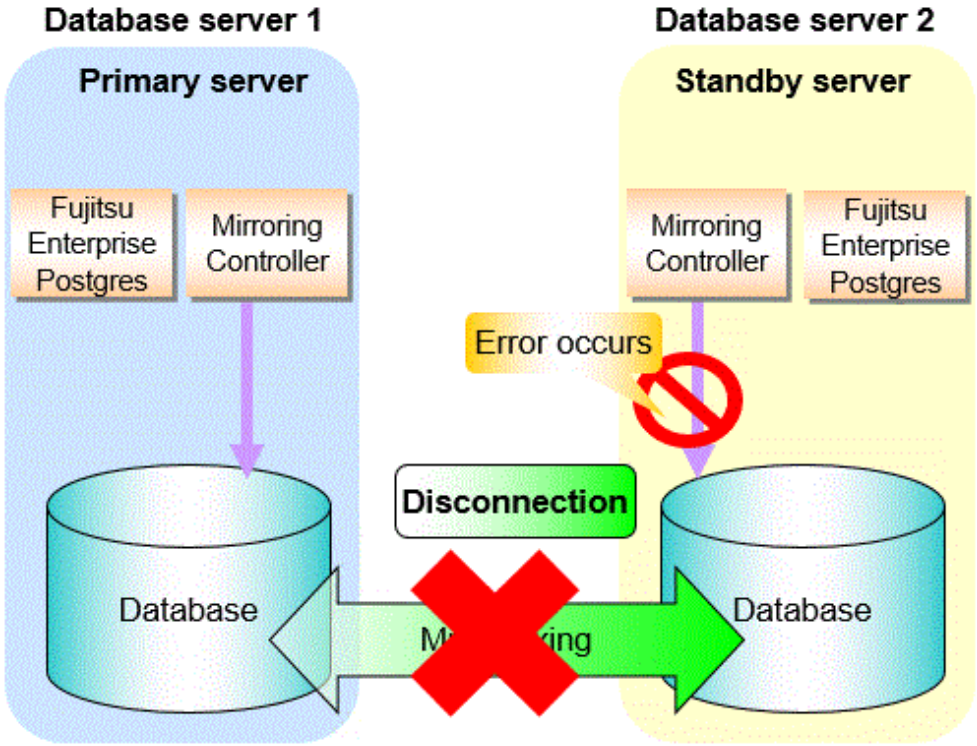


Figure 1.2 Standby server disconnection



Database degradation using the arbitration server

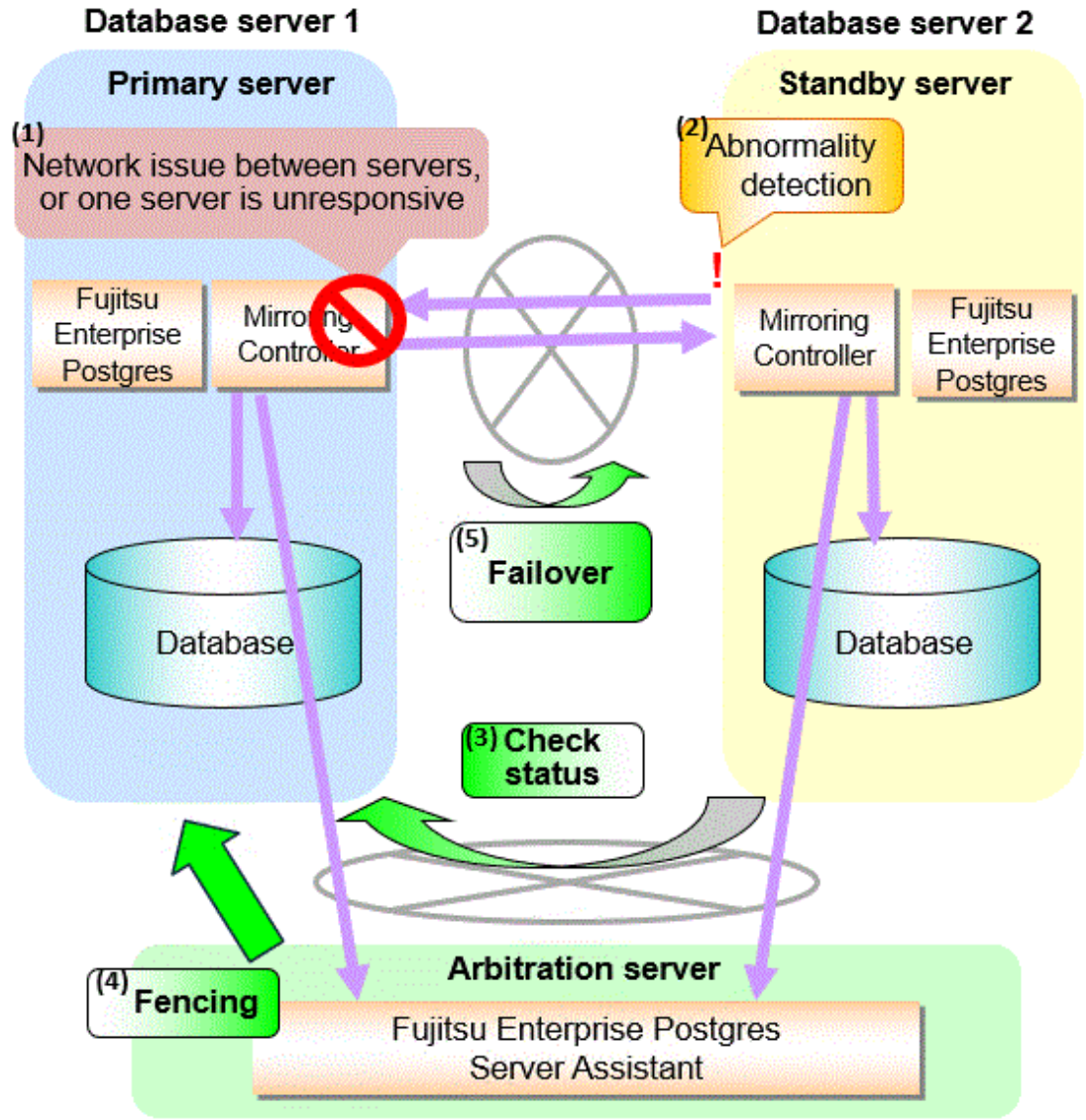
Fujitsu Enterprise Postgres provides the Server Assistant that objectively determines the status of database servers as a third party, and if necessary, isolates affected databases if the database servers are unable to accurately ascertain their mutual statuses in database multiplexing

mode, such as due to a network error between database servers, or server instability. Database degradation can be performed by using the server (arbitration server) on which the Server Assistant is installed.

For database degradation using the arbitration server, if the database servers are unable to check their mutual statuses (due to a network error between database servers or server instability), then the database server queries the arbitration server for the status of the other database server. If it is determined based on the heartbeat result that the status is unstable, the applicable database server will be isolated from the cluster system (fencing). The arbitration server periodically heartbeats the database server so that it can respond immediately to queries from the database server. The fencing process can be customized according to the environment where Mirroring Controller is used.

Additionally, the database servers are always performing their heartbeats for the arbitration server so that it can perform check requests any time.

Figure 1.3 Database degradation using the arbitration server



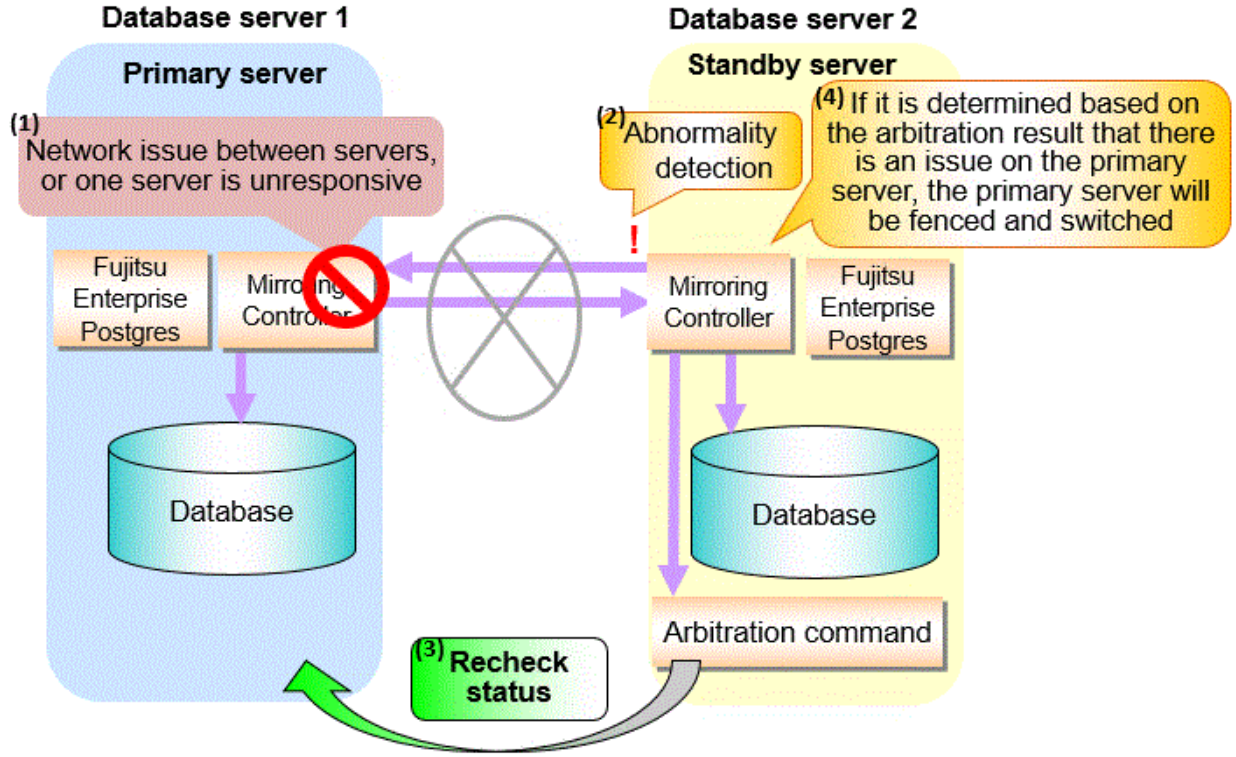
Note

Install the arbitration server on a different physical server to that of the database server. Refer to "1.2 System Configuration for Database Multiplexing Mode" for information on the system configuration when using the arbitration server.

Database degradation using the arbitration command

The arbitration command is a user command that performs arbitration processing in lieu of the arbitration server. If an arbitration server cannot be deployed, arbitration of the database server can be performed using the arbitration command.

Figure 1.4 Database degradation using the arbitration command



See

Refer to "2.6 Creating a User Command for a Database Server" or "Appendix C User Commands" for information on user commands.

1.1.1 Monitoring Using Database Multiplexing Mode

In database multiplexing mode, perform the monitoring below.

- Operating system or server failures, and no-response state

  By generating a heartbeat between Mirroring Controller on each server, operating system or server errors are detected and acknowledged between the relevant servers.

  The optimal operating method for environments where database multiplexing mode is performed can be selected from the following:

  - Use the arbitration server to perform automatic degradation (switch/disconnect)

    This is the default method.

    The arbitration server objectively determines the status of database servers, then isolates and degrades from the cluster system the ones with an unstable status.

    Refer to "Database degradation using the arbitration server" for details.

  - Call the user command that will perform the degradation decision, and perform automatic degradation

    If the arbitration server cannot be installed, select if arbitration processing can be performed by the user instead.

    Mirroring Controller queries the user command on whether to degrade. The user command determines the status of the database server, and notifies Mirroring Controller whether to perform degradation.

Refer to "Database degradation using the arbitration command" for details.

- Notification messages

  Use this method if using a two-database server configuration.

  Mirroring Controller outputs messages to the system log when an abnormality is detected. This ensures that a split brain will not occur due to a heartbeat abnormality - however, automatic switching will not be performed if the primary server operating system or server fails or becomes unresponsive.

- Perform automatic degradation unconditionally after a heartbeat abnormality

  This method is not recommended, because Mirroring Controller unconditionally will perform automatic degradation after heartbeat abnormalities.

- Database process failures, and no-response state

  Mirroring Controller periodically accesses the database processes and checks the status. A process error is detected by monitoring whether an access timeout occurs.

- Disk failure

  Mirroring Controller periodically creates files on the data storage destination disk below. A disk error is detected when an I/O error occurs.

  - Data storage destination disk
  - Transaction log storage destination disk
  - Tablespace storage destination disk

  Failures that can be detected are those that physically affect the entire system, such as disk header or device power failures.

- Streaming replication issue

  Mirroring Controller detects streaming replication issues (log transfer network and WAL send/receive processes) by periodically accessing the PostgreSQL system views.

- Mirroring Controller process failure and no response

  In order to continue the monitoring process on Mirroring Controller, Mirroring Controller process failures and no responses are also monitored.

  The Mirroring Controller monitoring process detects Mirroring Controller process failures and no responses by periodically querying the Mirroring Controller process. If an issue is detected, Mirroring Controller is automatically restarted by the Mirroring Controller monitoring process.

Point

- If output of messages is selected as the operation to be performed when a heartbeat abnormality is detected during heartbeat monitoring of the operating system or server, automatic degradation will not be performed.
  However, if an issue in the WAL send process is detected on the primary server, then the standby server will be disconnected, and as a result an automatic disconnection may be performed even if the standby server operating system or server fails or becomes unresponsive.
- You can select in the parameters if the primary server will be switched if a database process is unresponsive or if tablespace storage destination disk failure is detected. Refer to "Appendix A Parameters" for details.
- If the standby server was disconnected, Mirroring Controller will automatically comment out the synchronous_standby_names parameter and synchronized_standby_slots parameter in the postgresql.conf file of the primary server. Accordingly, you can prevent the application processing for the primary server being stopped.

Note

If the role of primary server was switched to another server and then starts degrading, the original primary server will not become the standby server automatically. Remove the cause of the error, and then change the role of the original primary server to the server currently acting as standby server. Refer to "4.1 Action Required when Server Degradation Occurs" for details.

1.1.2 Referencing on the Standby Server

1.1.2.1 If Prioritizing the Main Job on the Primary Server

If a reference job is performed on the standby server and the primary server is switched, this may impact the main job from the point of view of load and conflict. This is because, on the new primary server (that is, the original standby server), both the main job that was being executed on the original primary server and the reference job that was being continued on the original standby server will be processed.

Therefore, to degrade the reference job (so that the impact on the main job is reduced), you can select the user command below to disconnect the reference job that was performed on the original standby server.

- Post-switch command

Note

If continuing with the referencing job after switching the primary server, give careful consideration to the server resource estimates, and the likely impact on performance.

1.1.2.2 If Performing the Referencing Job on the Synchronous Standby Server

If an issue such as a log transfer network failure obstructs the continuation of a job on the primary server, the standby server may be automatically disconnected from the cluster system.

Therefore, if operating the reference job on the assumption that the connection destination is the synchronous standby server, you can select to temporarily stop the job by using the user command or the feature below, so that unexpected referencing of past data does not occur as a result of the disconnection.

- Pre-detach command
- Forced stoppage of the standby server instance on disconnection (specify in the parameter of the server configuration file)

Additionally, if the standby server is incorporated into the cluster system, reference jobs can be started or resumed by using the user command below.

- Post-attach command

See

- Refer to "2.6 Creating a User Command for a Database Server" or "Appendix C User Commands" for information on each user command.
- Refer to "A.4.1 Server Configuration File for the Database Servers" for information on the server configuration file of the database server.

Point

Mirroring Controller will continue processing regardless of the processing result of the above user commands and features.

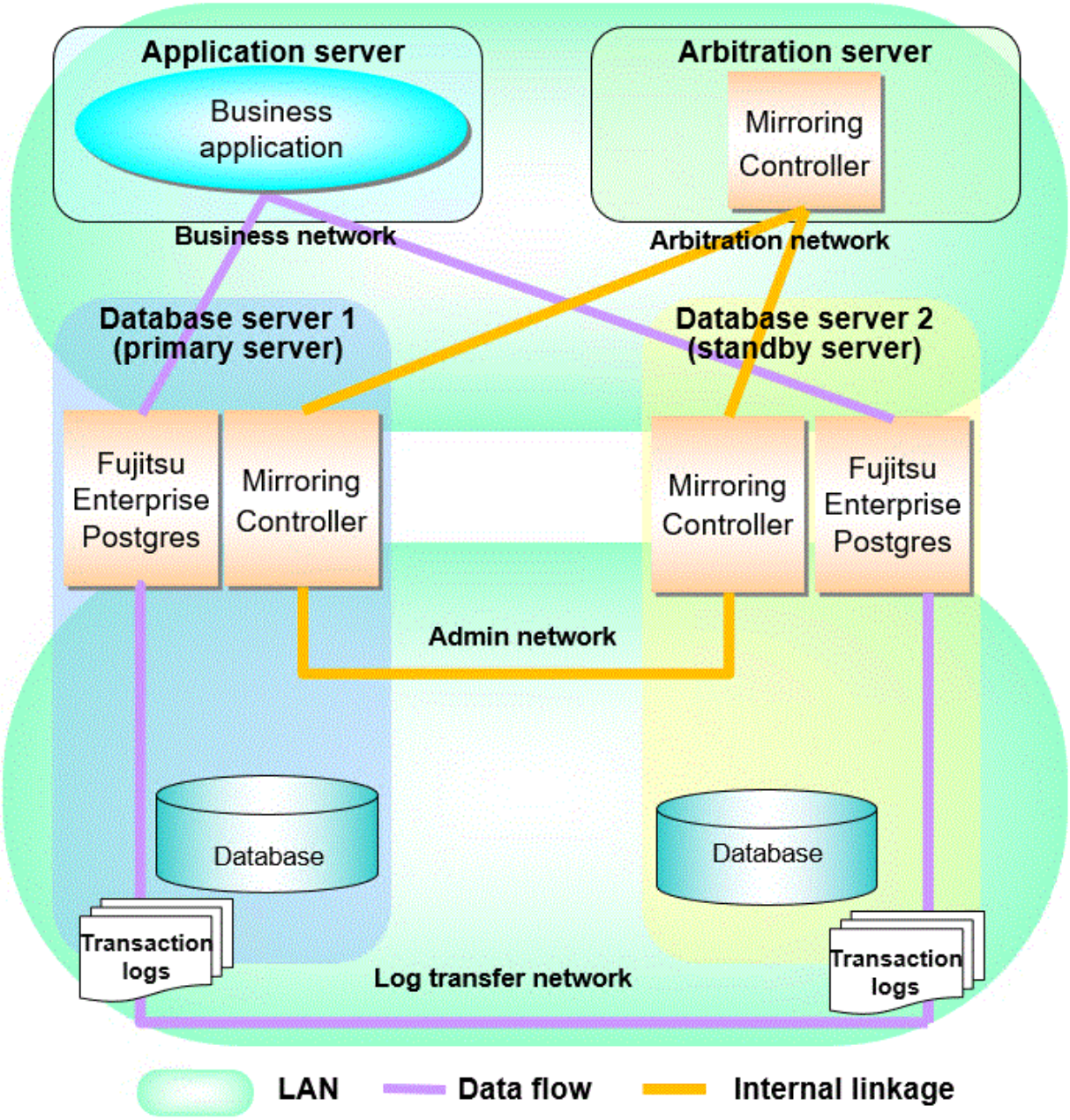
1.2 System Configuration for Database Multiplexing Mode

This section explains the products, features, and networks that are part of a database multiplexing system.

The following table shows the network types uses by database multiplexing systems.

Network type	Description
Job network	Network between the application that accesses the database, and the database server.
Arbitration network	Network used by the arbitration server to check the status of the primary server and standby server, and communicate with Mirroring Controller of the database servers. Additionally, if the job network is disconnected from outside, it can also be used as the arbitration network. Refer to "1.4 Security in Database Multiplexing" for details on network security.
Admin network	Network used by the primary server and the standby server to monitor each other using Mirroring Controller, and to control Mirroring Controller of other servers.
Log transfer network	Network used to transfer the transaction logs of the database, which is part of database multiplexing.

Figure 1.5 System configuration for database multiplexing mode



The arbitration server is installed to check the database server status as a third party, and to perform fencing. Therefore, to obtain the intended benefits, consider the following.

- Install the arbitration server on a different server to that of the database server.
- For the arbitration network, use a network that will not be impacted by line faults or the load on the admin network or log transfer network. This is necessary to correctly determine issues on the admin network or log transfer network.

Point

- The arbitration server can also be used as an application server. However, consider the server load.
- It is recommended to link the arbitration server with other cluster systems, in order to provide redundancy.
- Use the arbitration server in combination with the same version of Fujitsu Enterprise Postgres as that of the primary server and standby server.

- The arbitration server can be built on a different platform to that of the database server.

Note

Because the ping command of the operating system is used for heartbeat monitoring of the database server, configure the network so that ICMP can be used on the admin network and the arbitration network.

1.2.1 Mirroring Controller Resources

This section describes the database server and arbitration server resources of Mirroring Controller.

1.2.1.1 Database Server Resources

The only Mirroring Controller resource is the Mirroring Controller management directory, which stores the files that define the Mirroring Controller behavior, and the temporary files that are created when Mirroring Controller is active.

Note

- Do not create the Mirroring Controller management directory in a directory managed by Fujitsu Enterprise Postgres, otherwise it may be deleted by mistake or may cause unexpected problems when Fujitsu Enterprise Postgres recovery is performed (such as old version of files being restored).

  Refer to "Preparing Directories for Resource Deployment" in the Installation and Setup Guide for Server for information on the directories managed by Fujitsu Enterprise Postgres.

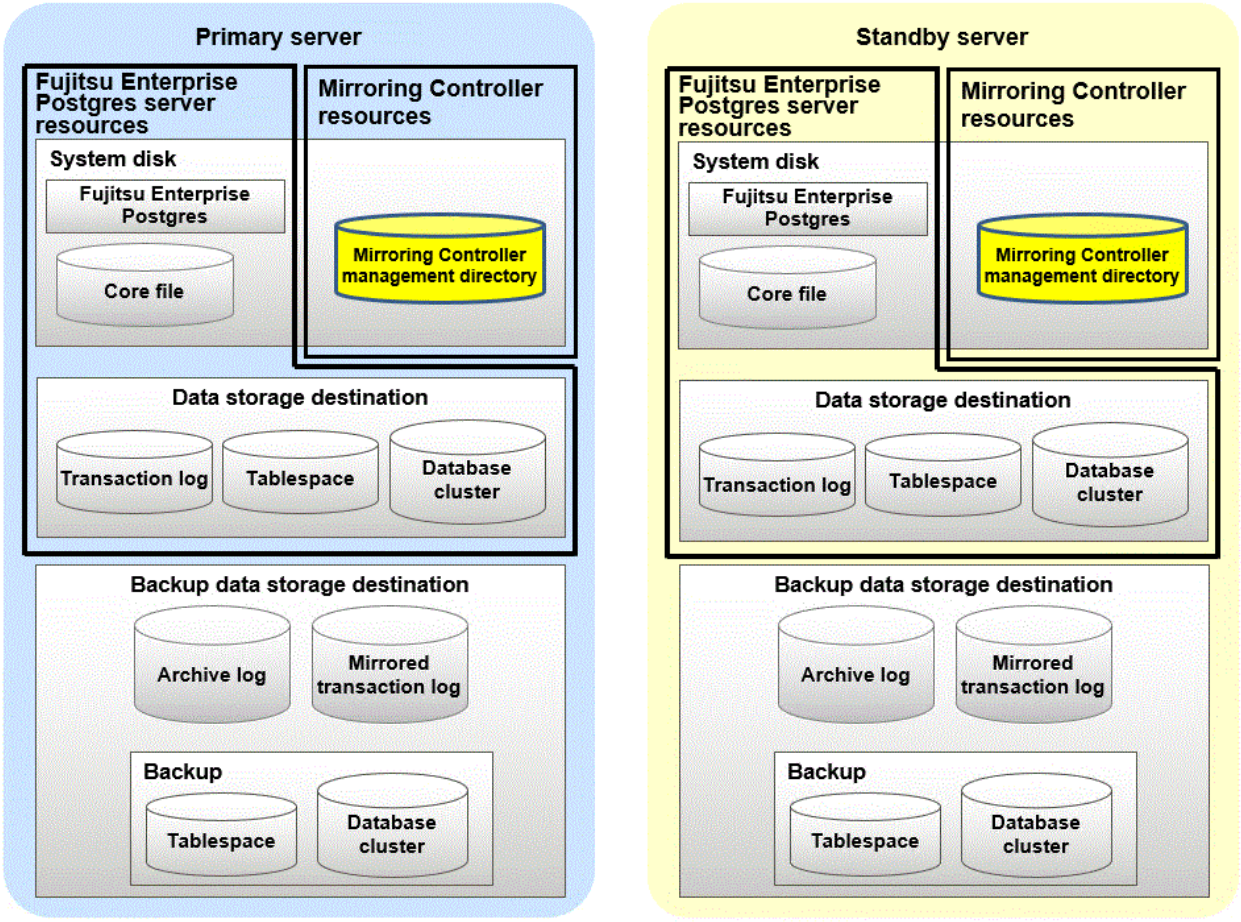
- The backup methods described in "Backing Up the Database" in the Operation Guide cannot be used to back up the Mirroring Controller resources. Therefore, users must obtain their own backup of Mirroring Controller resources, in addition to Fujitsu Enterprise Postgres server resources. Retrieve backups after stopping Mirroring Controller.

- If the automatic switch/disconnection is enabled, do not edit synchronous_standby_names parameter and synchronized_standby_slots parameter for the Mirroring Controller monitoring target instance. If Mirroring Controller is switched after editing, data may be lost or SQL access may stop.

- If you are building on a virtual machine or cloud, make sure the virtual machines are on different physical servers. Refer to your virtual machine software and cloud vendor documentation for instructions on how to deploy virtual machines.

The content on the primary server will be backed up. You cannot tell which server is the primary server to be backed up, because switching and failback may be performed between the servers. It is also impossible to tell which server is to be restored using the backed up data. Accordingly, ensure that you create a backup of each server when it is working as the primary server.

Figure 1.6 Configuration when backing up Mirroring Controller resources



1.2.1.2 Arbitration Server Resources

The only arbitration server resource is the Mirroring Controller arbitration process management directory. This directory stores the files that define the Mirroring Controller arbitration process behavior and the temporary files created when Mirroring Controller is active.

1.2.2 Mirroring Controller Processes

This section describes the database server and arbitration server processes of Mirroring Controller.

1.2.2.1 Database Server Processes

The database server processes comprise the Mirroring Controller process and Mirroring Controller monitoring process.

Process type	Description
Mirroring Controller process	Performs operating system/server and process heartbeat monitoring and disk abnormality monitoring between database servers. Additionally, it issues arbitration requests to the arbitration server.
Mirroring Controller monitoring process	Performs heartbeat monitoring of the Mirroring Controller process. If the Mirroring Controller process returns no response or is down, the monitoring process is restarted automatically.

1.2.2.2 Arbitration Server Process

The only arbitration process is the Mirroring Controller arbitration process.

Process type	Description
Mirroring Controller arbitration process	Performs rechecks for issues detected on the primary server or the standby server. Additionally, this process performs fencing if it determines that there is an issue on the primary server or the standby server.

1.2.3 Redundancy of the Admin and Log Transfer Networks

The admin network is an important one, because it is used by Mirroring Controller to check the status of each server.

Additionally, the log transfer network is an important one, because it is necessary to ensure data freshness.

Accordingly, configure a failure-resistant network by implementing network redundancy via channel bonding provided by the operating system or network driver vendor.

1.2.4 Notes on CPU Architecture and Products

A server using only PostgreSQL streaming replication cannot be specified as the database multiplexing system log transfer destination.

1.3 Deciding on Operation when a Heartbeat Abnormality is Detected

The operation to be performed when a heartbeat abnormality is detected using operating system/server heartbeat monitoring is decided on according to the environment where database multiplexing mode is performed or the operating method.

It is possible to select from the four operations below, and specify this in the parameters of Mirroring Controller:

- Use the arbitration server to perform automatic degradation (switch/disconnect)
- Call the user command that will perform the degradation decision, and perform automatic degradation
- Notification messages
- Perform automatic degradation unconditionally (switch/disconnect)

The table below shows if jobs can be continued on the primary server when an issue is detected during heartbeat monitoring of the operating system/server.

Continuation of jobs on the primary server when an issue is detected during heartbeat monitoring of the operating system/server

Operation	Abnormal event				
	Server/operating system failures or no responses		Admin network issue	Log transfer network issue	Issue on a network for both admin and log transfer
	Primary server	Standby server			
Automatic degradation using the arbitration server	Y (switch)	Y (disconnect)	Y	Y (disconnect)	Y (disconnect)
Call a user command and perform automatic degradation	Y (switch)	Y (disconnect)	Y	Y (disconnect)	Y (disconnect)
Notification messages	N (message notification only)	N (message notification only)	Y	Y (disconnect)	Y (disconnect)
Unconditional automatic degradation	Y (switch)	Y (disconnect)	N (split brain occurs)	Y (disconnect)	N (split brain occurs)

Y: Job can be continued

N: Job cannot be continued

1.4 Security in Database Multiplexing

The database server replicates the database on all servers that comprise the cluster system. It achieves this by transferring and reflecting the updated transaction logs of the database from the primary server to the standby server.

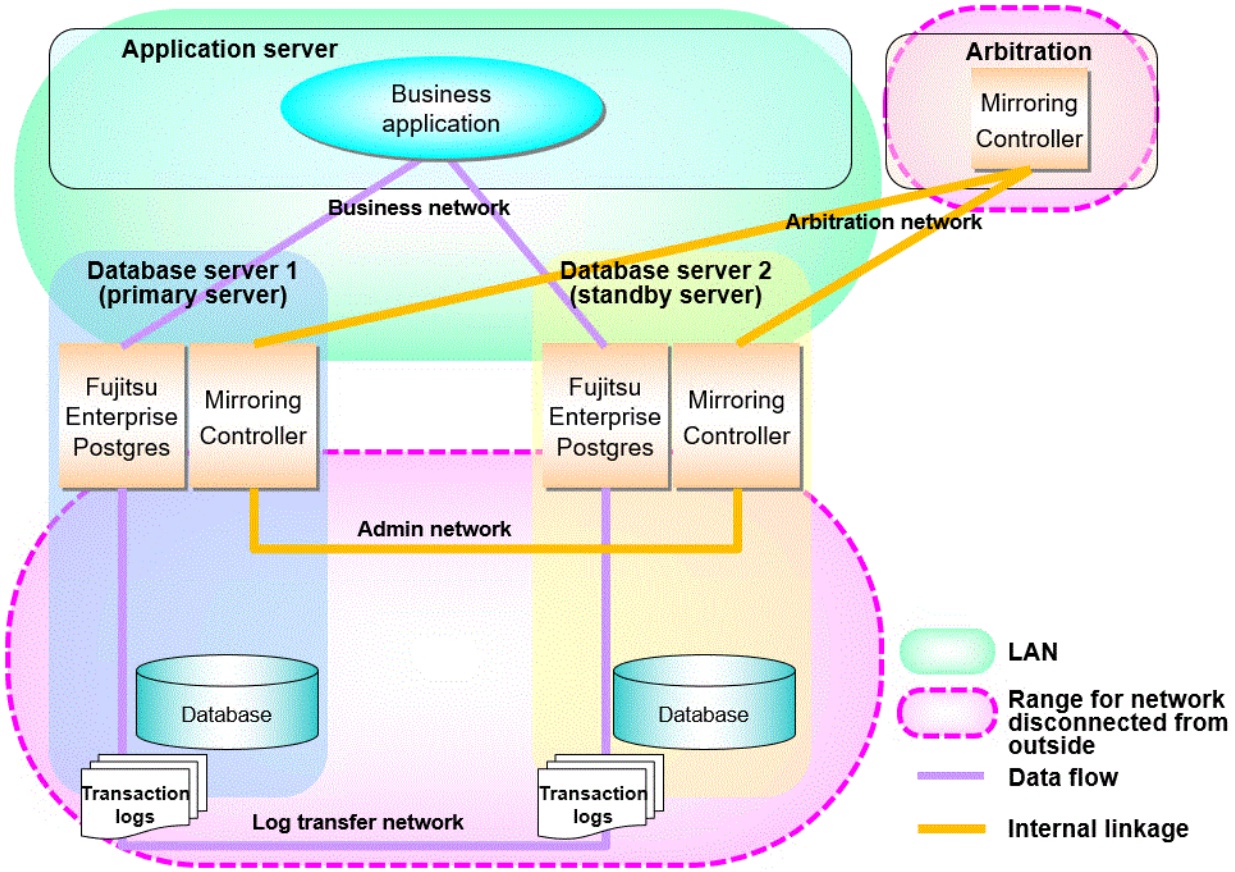
To safeguard the database against unauthorized access and preserve data confidentiality in transaction log transfers, carefully consider security and take note of the following when performing database multiplexing:

- Do not use trust authentication when using replication connection.
- Configure the admin network and the log transfer network so that they cannot be connected from the outside, as shown in Figure 1.7 Security.

Additionally, for the line on which Mirroring Controller connects from the database server to the arbitration server, take note of the following points and consider security carefully.

- Build a network with the arbitration server disconnected from outside, as shown in Figure 1.7 Security.

Figure 1.7 Security



However, it may not always be possible to adopt the configuration mentioned above. For example, you may want to place the servers in a nearby/neighboring office to minimize network delays.

In this case, combine the following features to enhance security:

- Authentication of the Standby Server
- Encryption of Transaction Logs Transferred to the Standby Server

When these features are combined, security will be achieved as shown below.

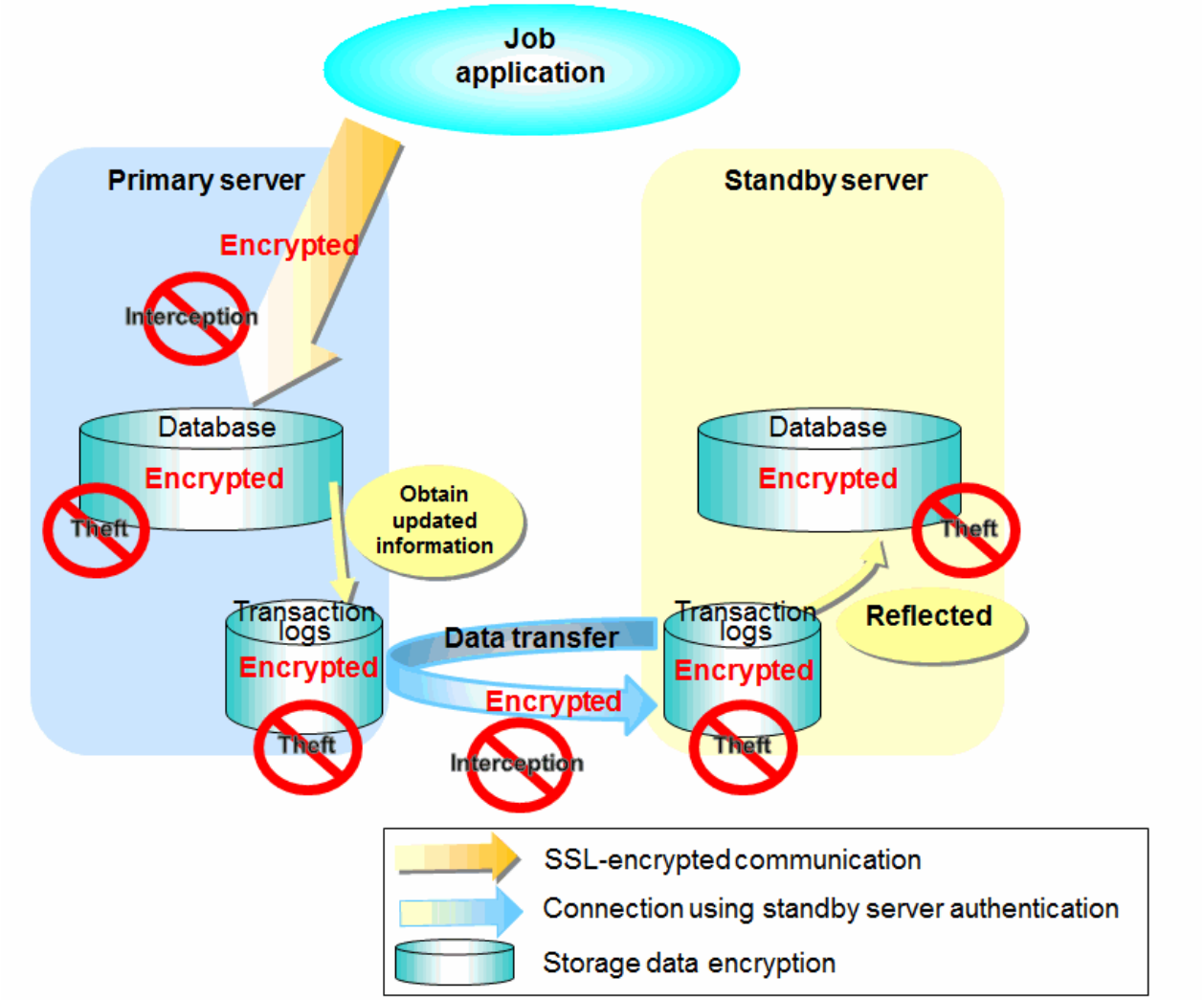
Point

If the job network is disconnected from outside, it can be used as the arbitration network. However, if a network is to be used as both a job network and arbitration network, consider the load on the network.

Note

If a port is blocked (access permission has not been granted) by a firewall, etc., enable use of the target port by granting access. Refer to the vendor document for information on how to open (grant access permission to) a port. Consider the security risks carefully when opening ports.

Figure 1.8 Security achieved when standby server authentication is combined with transaction log encryption



See

Refer to "Performing Database Multiplexing" under "Configuring Secure Communication Using Secure Sockets Layer" in the Operation Guide for information on encrypting SSL communications.

1.4.1 Authentication of the Standby Server

You can prevent spoofing connections from an external server purporting to be the standby server by using authentication with a user name and password.

Configure the setting in the primary server pg_hba.conf file so that authentication is performed for connections from the standby server in the same way as for connections from the client.

Refer to "Client Authentication" in the PostgreSQL Documentation for information on content that can be configured in pg_hba.conf.

Refer to "Policy-based Login Security" in the Operation Guide for information about security policy-based passwords operations for database multiplexing operations.

1.4.2 Encryption of Transaction Logs Transferred to the Standby Server

In case the authentication of the standby server is breached so that a malicious user purporting to be the standby server can spoof data, the transaction log data can be encrypted to prevent it from being deciphered. The transparent data encryption feature is used to encrypt the data.

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

Chapter 2 Setting Up Database Multiplexing Mode

This chapter describes how to set up database multiplexing mode, and how to check it.

Users who perform setup and operations on the database server

Setup and operations of the database server must be performed by the instance administrator user.

Users who perform setup and operations on the arbitration server

The following users may perform setup and operations on the arbitration server when it is used for automatic degradation.

- Any operating system user.

Point

- Mirroring Controller selects a database superuser as the user who will connect to the database instance. This enables instance administrator users and database superusers who operate the Mirroring Controller commands to run database multiplexing mode in different environments.
- The application name for connecting to the database instance is "mc_agent".

Matching the system times

Before starting the setup, ensure that the times in the primary server, standby server and arbitration server match, by using the operating system time synchronization feature, for example.

The tolerated difference is approximately one second.

If the system times are not synchronized (because the tolerated difference is exceeded, for example), problem investigation may be affected.

Configuring ICMP

Because the ping command of the operating system is used for heartbeat monitoring of the database server, configure the network so that ICMP can be used on the admin network and the arbitration network. Refer to the relevant operating system procedure for details.

Setup

The setup procedure is shown in the table below. However, the procedure on the arbitration server should be performed only when the arbitration server is used for automatic degradation. A distinction is made between the procedures on the primary server and standby server according to whether the arbitration server is used.

Step	Task			Refer to
	Primary server	Standby server	Arbitration server	
1	Installation			2.1 Installation
2	Preparing the database server		Preparing the arbitration server	2.2 Preparing for Setup
3			Configuring the arbitration server	2.3.1 Configuring the Arbitration Server
4			Creating a user command	2.3.2 Creating a User Command for the Arbitration Server
5			Starting the arbitration process	2.3.3 Starting the Mirroring Controller Arbitration Process
6	Setting up database multiplexing mode			2.4.1 Setting Up Database Multiplexing Mode on the Primary Server

Step	Task			Refer to
	Primary server	Standby server	Arbitration server	
7	Creating, setting, and registering the instance			2.4.2 Creating, Setting, and Registering the Primary Server Instance
8	Creating a user command			2.6 Creating a User Command for a Database Server
9	Starting Mirroring Controller			2.4.3 Starting Mirroring Controller on the Primary Server
10		Setting up database multiplexing mode		2.5.1 Setting Up Database Multiplexing Mode on the Standby Server
11		Creating, setting, and registering the instance		2.5.2 Creating, Setting, and Registering the Standby Server Instance
12		Creating a user command		2.6 Creating a User Command for a Database Server
13		Starting Mirroring Controller		2.5.3 Starting Mirroring Controller on the Standby Server
14	Confirming the streaming replication status			2.7 Confirming the Streaming Replication Status
15	Checking the connection status			2.8.1 Checking the Connection Status on a Database Server
16		Checking the connection status		2.8.1 Checking the Connection Status on a Database Server
17			Checking the connection status	2.8.2 Checking the Connection Status on the Arbitration Server
18	Creating applications			2.9 Creating Applications
19	Checking the behavior			2.10 Checking the Behavior

Explanations for each step are provided below.

Information

- The setup procedure is also the same when changing the mode on a single server to database multiplexing mode. In this case, omit the installation of Fujitsu Enterprise Postgres and the creation of the instance.

  Refer to "3.9.2 Changing from Single Server Mode to Database Multiplexing Mode" for details.

- The primary and standby server can be pseudo-configured on the same server for system testing, for example. In this case, the setup can be performed using the same procedure, however there will be some supplementary steps.

  Before performing the setup, refer to "Appendix B Supplementary Information on Building the Primary Server and Standby Server on the Same Server".

2.1 Installation

Refer to the manuals below, and then install the product.

See

- Refer to the Installation and Setup Guide for Server for details on how to install Fujitsu Enterprise Postgres.

- Refer to the Installation and Setup Guide for Server Assistant for information on installing the Server Assistant on the arbitration server.

Note

Do not use the arbitration server also as a database server. The arbitration server is installed to check the database server status as a third party, and to perform fencing. Using the arbitration server also as a database server nullifies the effectiveness of the arbitration server.

2.2 Preparing for Setup

This section describes the preparation required before setting up Mirroring Controller.

2.2.1 Preparing the Database Server

2.2.1.1 Preparing the Backup Disk

In Mirroring Controller, by performing a backup, recovery is possible even if all server disks are corrupted.

The content on the primary server should be backed up. However, through switching and failback, the standby server may also become the primary server. Accordingly, prepare each of the backup disk devices for the primary and standby servers. Perform backup on the primary server used at the time of the backup.

2.3 Setting Up the Arbitration Server

This section explains how to set up the arbitration server.

2.3.1 Configuring the Arbitration Server

This section explains how to set up database multiplexing mode on the arbitration server.

In database multiplexing mode, the files that are required for operations are managed in the Mirroring Controller arbitration process management directory.

There is one Mirroring Controller arbitration process management directory for each arbitration process.

Point

The arbitration process for each database multiplexing system can be started on a single arbitration server.

See

- Refer to the Reference for information on the mc_arb command.
- Refer to "Appendix A Parameters" for information on the parameters to be edited for the setup.

Perform the following procedure:

1. On the arbitration server, log in as any operating system user who starts and stops the arbitration process.
2. Configure the environment variables.

   Set the following environment variables:

   - PATH

     Add the installation directory "/bin".

- MANPATH

  Add the installation directory "/share/man".

Example

The following example configures environment variables when the installation directory is "/opt/fsepv<*x*>assistant".

Note that "<*x*>" indicates the product version.

sh, bash

```
$ PATH=/opt/fsepv<x>assistant/bin:$PATH ; export PATH
$ MANPATH=/opt/fsepv<x>assistant/share/man:$MANPATH ; export MANPATH
```

csh, tcsh

```
$ setenv PATH /opt/fsepv<x>assistant/bin:$PATH
$ setenv MANPATH /opt/fsepv<x>assistant/share/man:$MANPATH
```

3. Create the Mirroring Controller arbitration process management directory that will store the files required by the arbitration server.

   Use ASCII characters in the Mirroring Controller arbitration process management directory.

4. In the network configuration file (network.conf), define the Mirroring Controller network configuration that will be managed by the Mirroring Controller arbitration process.

   Create network.conf in the Mirroring Controller arbitration process management directory, based on the sample file. For network.conf, set read and write permissions only for the operating system user who starts and stops the arbitration process in step 1.

   If users other than this are granted access permissions, the mc_arb command will not work. Accordingly, users other than the operating system user who starts and stops the arbitration process in step 1 are prevented from operating the Mirroring Controller arbitration process.

   Sample file

   ```
   /installDir/share/mcarb_network.conf.sample
   ```

   In network.conf, specify the IP address or host name and port number of the primary server and standby server, and define the Mirroring Controller network configuration that will be managed by the Mirroring Controller arbitration process.

   Refer to "A.3 Network Configuration File" for details.

   A definition example is shown below.

   ```
   Example)
   The IDs of the servers are set to "server1" and "server2", and their port numbers are set to
   "27541".
   ```

   ```
   server1 192.0.3.100 27541
   server2 192.0.3.110 27541
   ```

5. In the arbitration configuration file (arbitration.conf), define the information related to control of the Mirroring Controller arbitration process.

   Create arbitration.conf in the Mirroring Controller arbitration process management directory, based on the sample file. For arbitration.conf, set read and write permissions only for the operating system user who starts and stops the arbitration process in step 1. If users other than this are granted access permissions, the mc_arb command will not work.

   Sample file

   ```
   /installDir/share/mcarb_arbitration.conf.sample
   ```

   Set the parameters shown in the table below in arbitration.conf.

Table 2.1 Parameters

Parameter	Content specified	Remarks
port	Port number of the Mirroring Controller arbitration process	The port number must be 0 to 65535. Ensure that the port number does not conflict with other software. Do not specify an ephemeral port that may temporarily be assigned by another program.
my_address	'*ipAddrOrHostNameThatAcceptsConnectionFromMirroringControllerProcessOnDbServer*' [Setting example] my_address = '192.0.3.120'	IPv4 and IPv6 addresses can be specified. Specify the IP address, enclosed in single quotation marks (').
syslog_ident	'*programName*'	Specify using single quotation marks (') to enclose the program name used to identify the Mirroring Controller arbitration process message in the system log. Use ASCII characters excluding spaces to specify this parameter. The default is 'MirroringControllerArbiter'.
fencing_command	'*fencingCmdFilePath*' [Setting example] fencing_command = '/arbiter/fencing_dir/execute_fencing.sh'	Specify the full path of the fencing command that fences a database server where it is determined that an error has occurred. Enclose the path in single quotation marks ('). Specify the path using less than 1024 bytes.
fencing_command_timeout	Timeout for fencing command (seconds)	If the command does not respond within the specified number of seconds, it is determined that fencing has failed and a signal (SIGTERM) is sent to the fencing command execution process. Specify a value between 1 and 2147483647. The default is 20 seconds.

Information

Refer to "A.4.2 Arbitration Configuration File" for information on the parameters and for other parameters.

2.3.2 Creating a User Command for the Arbitration Server

The only user command for the arbitration server is the fencing command.

The fencing command is a user command that is called by the Mirroring Controller arbitration process if Mirroring Controller performs arbitration processing and determines that a database server is unstable.

In the fencing command, the user implements a process that isolates a database server from a cluster system by, for example, stopping the target operating system or server. The fencing command that was created is to be specified for the parameter in the arbitration configuration file. Refer to "A.4.2 Arbitration Configuration File" for information on the parameters.

- Fencing the primary server during the switch
  - Prevent the Mirroring Controller management process on the primary server from communicating with the Mirroring Controller management process on the other server.

  - Prevent applications from connecting to the primary server instance.

- Fencing the standby server during disconnection
  - Prevent the Mirroring Controller management process on the standby server from communicating with the Mirroring Controller management process on the other server.
  - Prevent applications from connecting to the standby server instance.
  - Prevent the standby server from continuing streaming replication.

See

Refer to "Appendix C User Commands" for information on user commands.

2.3.3 Starting the Mirroring Controller Arbitration Process

This section explains how to start the Mirroring Controller arbitration process.

An operating system user who has logged in to the arbitration server can start the Mirroring Controller arbitration process by executing the mc_arb command in start mode.

Example)

```
$ mc_arb start -M /mcarb_dir/arbiter1
```

2.4 Setting Up the Primary Server

This section explains how to set up the primary server.

2.4.1 Setting Up Database Multiplexing Mode on the Primary Server

This section explains how to set up database multiplexing mode on the primary server.

In database multiplexing, the files that are required for operations are managed in the Mirroring Controller management directory.

There is one Mirroring Controller management directory for each instance.

Note

- Do not place the Mirroring Controller management directory in a directory managed by Fujitsu Enterprise Postgres, otherwise it may be deleted by mistake with the directories managed by Fujitsu Enterprise Postgres, and an old version of files may be restored.

See

- Refer to "Preparing Directories for Resource Deployment" in the Installation and Setup Guide for Server for details on the directories that are managed by Fujitsu Enterprise Postgres.
- Refer to "mc_ctl" in Reference for information on the command.
- Refer to "Appendix A Parameters" for details on each parameter to be edited for the setup.

Perform the following procedure:

1. Log in to the primary server.
2. Create the Mirroring Controller management directory that will store the files required by database multiplexing.

   Use ASCII characters in the Mirroring Controller management directory.

Additionally, grant "Write" permission to the instance administrator user for the Mirroring Controller management directory.

3. In the network configuration file (network.conf), define the network configuration that will link between the Mirroring Controller processes.

   Create the network.conf file in the Mirroring Controller management directory, based on the sample file. For network.conf, set read and write permissions for the instance administrator user only.

   If users other than the instance administrator user are granted access, the mc_ctl command will not work. In this way, users other than the instance administrator user are prevented from operating Mirroring Controller.

   Sample file

   ```
   /installDir/share/mc_network.conf.sample
   ```

   In network.conf, specify the IP address or host name and port number of the primary server and standby server, and define the network configuration that will link between the Mirroring Controller processes, and between Mirroring Controller processes and the Mirroring Controller arbitration process.

   Refer to "A.3 Network Configuration File" for details.

   A definition example is shown below.

   The content to be defined depends on the operation settings at the time a heartbeat abnormality is detected.

   When automatic degradation by the arbitration server is selected

   ```
   Example)
   The IDs of the primary server and standby server are set to "server1" and "server2", and their
   port numbers are set to "27540" and "27541". The ID of the server of the Mirroring Controller
   arbitration process is set to "arbiter", and its port number is set to "27541".
   ```

   ```
   server1 192.0.2.100,192.0.3.100 27540,27541 server
   server2 192.0.2.110,192.0.3.110 27540,27541 server
   arbiter 192.0.3.120 27541 arbiter
   ```

   Ensure that the port numbers set for the primary server, standby server, and arbitration server do not conflict with other software. In addition, when the arbitration server is used for automatic degradation, use a network in which the arbitration network is not affected by a line failure in the admin network.

   When the server type is "server", two IP addresses or host names, and two port numbers need to be specified in the following order:

   - IP address or host name of the database server used as the admin network
   - IP address or host name of the database server used as the arbitration network
   - Port number of the database server used as the admin network
   - Port number of the database server used as the arbitration network

   If the server type is "arbiter", specify the IP address or host name set for the my_address parameter and the port number set for the port parameter in arbitration.conf of the arbitration server.

   When operation other than automatic degradation by the arbitration server is selected

   ```
   Example)
   The IDs of the servers are set to "server1" and "server2", and their port numbers are set to
   "27540".
   ```

   ```
   server1 192.0.2.100 27540
   server2 192.0.2.110 27540
   ```

   Ensure that the port numbers for the primary and standby server do not conflict with other software.

   Register in /etc/services the port number of the primary server, because programs such as WebAdmin use it to search for available port numbers.

   Register any name as the service name.

4. Define the information related to Mirroring Controller monitoring and control in the *serverIdentifier*.conf file.

   Create the *serverIdentifier*.conf file in the Mirroring Controller management directory, based on the sample file.

   For *serverIdentifier*.conf, set read and write permissions for the instance administrator user only. If users other than the instance administrator user are granted access, the mc_ctl command will not work.

   As the file name for the *serverIdentifier*.conf file, use the server identifier name that was specified in the network.conf file in step 3.

   Sample file

   ```
   /InstallDir/share/mc_server.conf.sample
   ```

   Set the parameters shown in the table below in the *serverIdentifier*.conf file.

   Table 2.2 Parameters

Parameter	Content specified	Remarks
db_instance	'*dataStorageDestinationDir*'	Use ASCII characters, enclosed in single quotation marks (').
db_instance_password	'*passwordOfInstanceAdminUser*'	If password authentication is performed, you must specify this parameter in the settings used when Mirroring Controller connects to a database instance. Use ASCII characters, enclosed in single quotation marks ('). If the specified value of this parameter includes ' or \, write \' or \\, respectively.
enable_hash_in_password	on or off	Specify on to treat the # in the db_instance_password specification as a password character, or off to treat it as a comment. The default is "off".
syslog_ident	'*programName*'	Specify the program name to be used to identify the Mirroring Controller messages in the system log. Use ASCII characters excluding spaces, enclosed in single quotation marks ('). Use the same program name as the parameter in the postgresql.conf file ensures that the Mirroring Controller output content can be referenced transparently, so log reference is easy.
remote_call_timeout	Admin communication timeout	Specify the timeout value (milliseconds) of the Mirroring Controller agent process for communication between servers. Specify a value that is less than the operation system TCP connection timeout. Also, when using the Mirroring Controller arbitrage process for arbitrage, fencing, and state transition commands, specify a value that is greater than the sum of the timeout values.
heartbeat_error_action	Operation when a heartbeat abnormality is detected using operating system or server heartbeat monitoring	arbitration: Perform automatic degradation using the arbitration server.

Parameter	Content specified	Remarks
		command: Call a user command to determine degradation, and perform automatic degradation if required. message: Notify messages. fallback: Perform automatic degradation unconditionally. Set the same value on the primary server and standby server.
heartbeat_interval	Interval time for abnormality monitoring during heartbeat monitoring of the operating system or server (milliseconds)	Abnormality monitoring of the operating system or server is performed at the interval (milliseconds) specified in heartbeat_interval. This parameter setting is used as the default for database process heartbeat monitoring, streaming replication abnormality monitoring, and disk abnormality monitoring. When setting the monitoring time, there are some considerations to take into account to optimize degradation using abnormality monitoring. Refer to "2.11.4.1 Tuning for Abnormality Monitoring of the Operating System or Server" for details.
heartbeat_timeout	Timeout for abnormality monitoring during heartbeat monitoring of the operating system or server (seconds)	
heartbeat_retry	Number of retries for abnormality monitoring during heartbeat monitoring of the operating system or server (number of times)	
fencing_command	'*fencingCmdFilePath*' [Setting example] fencing_command = '/mc/fencing_dir/execute_fencing.sh'	Specify the full path of the fencing command that fences a database server where an error is determined to have occurred. Enclose the path in single quotation marks ('). This parameter must be specified when "command" is set for heartbeat_error_action. Specify the path using less than 1024 bytes.
fencing_command_timeout	Fencing command timeout (seconds)	If the command does not respond within the specified number of seconds, fencing is determined to have failed and a signal (SIGTERM) is sent to the fencing command execution process. Specify a value between 1 and 2147483647. The default is 20 seconds.
arbitration_timeout	Timeout for arbitration processing in the Mirroring Controller arbitration process (seconds)	The specified value must be at least equal to the heartbeat monitoring time of the operating system or server + fencing_command_timeout in the arbitration configuration file. If there is no response for at least the number of seconds specified, the primary server will not be switched and the standby server will not be disconnected. Therefore, perform degradation manually. Specify a value between 1 and 2147483647.

Parameter	Content specified	Remarks
		This parameter does not need to be set for operation that does not use the arbitration server.
arbitration_command	'*arbitrationCmdFilePath*' [Setting example] arbitration_command = '/mc/arbitration_dir/execute_arbitration_command.sh'	Specify the full path of the arbitration command to be executed when an abnormality is detected during heartbeat monitoring of the operating system or server. Enclose the path in single quotation marks ('). This parameter must be specified when "command" is set for heartbeat_error_action. Specify the path using less than 1024 bytes.
arbitration_command_timeout	Timeout for arbitration command (seconds)	If the arbitration command does not respond within the specified number of seconds, it is determined that execution of the arbitration command has failed and a signal (SIGTERM) is sent to the arbitration command execution process. Specify a value between 1 and 2147483647. This parameter can be specified only when "command" is set for heartbeat_error_action.

Information

Refer to "A.4.1 Server Configuration File for the Database Servers" for information on the parameters and for other parameters.

2.4.2 Creating, Setting, and Registering the Primary Server Instance

This section explains how to create, set, and register the primary server instance.

See

- Refer to "Client Authentication" in the PostgreSQL Documentation for information on the pg_hba.conf file.
- Refer to "A.1 Parameters Set on the Primary Server" for information on the postgresql.conf file.
- Refer to "mc_ctl" in Reference for information on the command.

Perform the following procedure:

1. Refer to "Setup" in the Installation and Setup Guide for Server, and then perform the Fujitsu Enterprise Postgres setup and create the Fujitsu Enterprise Postgres instance.

   Use ASCII characters in the data storage destination directory.

Note

- If degradation starts occurring due to an error during operations in database multiplexing mode, recovery is required for the standby server. There are some conditions to execute the pg_rewind command to recover the standby server. One of the conditions can be satisfied by enabling checksums when executing the initdb command. This is not mandatory. Refer to "4.1.1.1.3 Identify cause of error and perform recovery" for details.

2. When using transparent data encryption, configure the encryption settings for the storage data.

   Create a keystore file.

   If you want to use the key management system as a keystore, set the connection information for the key management system and declare the master encryption key.

   Refer to "Protecting Storage Data Using Transparent Data Encryption" or "Using Transparent Data Encryption with Key Management Systems as Keystores" in the Operation Guide for details, and then configure the settings.

3. Add the following entry to the pg_hba.conf file to authenticate connections from the standby server.

   Copy the file to the standby server later.

```
# TYPE    DATABASE        USER        ADDRESS                     METHOD
  host    replication     fsep        standbyServerAddress        authenticationMethod
  host    replication     fsep        primaryServerAddress        authenticationMethod
```

   For the primary and standby server addresses, specify the IP address that will connect to the log transfer network.

   Additionally, all servers can be used as the primary server or the standby server, so add entries for the addresses of all servers that comprise the database multiplexing system.

Point

Setting an authentication method other than trust authentication

If the primary server becomes the standby server, to perform automatic authentication of connections to the primary server, create the .pgpass file in the home directory of the instance administrator user, and then specify a password for the replication database. Accordingly, the instance administrator operating system user and the user registered in the database will be the same, so you can verify that the connection was not made by an unspecified user. Additionally, the password that was set beforehand will be used in the authentication, so that the connection will be automatic.

Note

If trust authentication is set, all OS users who can log in to the primary server will be able to connect, and if one of these is a malicious user, then that user can corrupt the standby server data, or cause the job system to fail, by sending an erroneous transaction log. Therefore, decide on the authentication method according to the security requirements of the system using database multiplexing mode.

Refer to "Authentication Methods" in the PostgreSQL Documentation for details on the authentication methods that can be set.

4. Configure this setting to enable the instance administrator user of the primary server to connect as a database application.

   This setting enables the connection to the instance using the user name of the instance administrator user, so that Mirroring Controller can monitor instance errors. Configure this setting to enable the connection to the postgres database.

   - If password authentication is used

     In the db_instance_password parameter of the *serverIdentifier*.conf file, specify the password for the instance administrator user. This password is used to connect to the database instance. If a password is not specified in the db_instance_password parameter, the connection to the database instance from Mirroring Controller will fail, and it will not be possible to perform the process monitoring of the instance.

   - If password authentication is not used

     There is no need to specify the password in the db_instance_password parameter.

     Even if the password for the instance administrator user is specified in the db_instance_password parameter, it will be ignored.

   - If certificate authentication using SSL is used

     Specify connection parameters for SSL in the db_instance_ext_ pq_conninfo parameter and db_instance_ext_jdbc_conninfo parameter in the *serverIdentifier*.conf file. If the pasrameters are not specified, the connection to the database instance from

Mirroring Controller will fail, and it will not be possible to perform the process monitoring of the instance. If certificate authentication using SSL is not performed, the parameters specification is not required.

For information about the db_instance_ext_pq_conninfo and db_instance_ext_jdbc_conninfo parameters, refer to "A.4.1 Server Configuration File for the Database Servers".

An example of setting the authentication method is shown below.

```
# TYPE    DATABASE        USER        ADDRESS         METHOD
  host    postgres        fsep        127.0.0.1/32    authenticationMethod
```

Note

Mirroring Controller uses the PostgreSQL JDBC 4.2 driver to connect to the database instance. Therefore, for the authentication method, specify a method supported by the JDBC driver. If an authentication method not supported by the JDBC driver is specified, Mirroring Controller will fail to start. Refer to the PostgreSQL JDBC Driver Documentation for information on authentication methods supported by the JDBC driver.

5. To use database multiplexing mode, specify the parameters shown in the table below in the postgresql.conf file.

   The postgresql.conf file is copied when the standby server instance is created. Accordingly, set the required parameters in the standby server.

   To use database multiplexing mode, specify the parameters shown in the table below in the postgresql.conf file. After editing the postgresql.conf file, restart the instance.

Table 2.3 Parameters

Parameter	Content specified	Remarks
wal_level	replica or logical	Specify "logical" when logical decoding is also to be used.
max_wal_senders	2 or more	Specify "2" when building a Mirroring Controller cluster system. When additionally connecting asynchronous standby servers to the cluster system, add the number of simultaneous connections from these standby servers.
synchronous_standby_names	'*standbyServerName*'	Specify the name that will identify the standby server. Enclose the name in single quotation marks ('). Do not change this parameter while Mirroring Controller is running. Do not specify multiple names to this parameter as the Mirroring Controller can manage only one standby server.
synchronized_standby_slots	'*physicalReplicationSlotName*'	Specify this parameter if the primary server will be a logical replication publication. Setting this parameter ensures that the subscriber is updated after WAL is sent to the standby server. This allows logical replication to continue if the primary server fails and the standby server is promoted. Do not change this parameter while the Mirroring Controller is running. Because the Mirroring Controller can manage only one standby server, do not specify multiple names for this parameter.
hot_standby	on	Specify whether queries can be run on the standby server.
wal_keep_size	WAL save size (megabytes)	If a delay exceeding the value set in this parameter occurs, the WAL segment required later by the primary server may be deleted.

Parameter	Content specified	Remarks
		Additionally, if you stop a standby server (for maintenance, for example), consider the stop time and set a value that will not cause the WAL segment to be deleted. Refer to "Estimating Transaction Log Space Requirements" in the Installation and Setup Guide for Server for information on estimating the WAL save size.
wal_log_hints	on	When using the pg_rewind command to recover a standby server, specify this parameter or enable checksums when executing the initdb command.
wal_sender_timeout	Timeout (milliseconds)	Specify the time period after which it is determined that an error has occurred in the transaction log transfer on the primary server. By aligning this value with the value for the database process heartbeat monitoring time, you can unify the time after which it is determined that an error has occurred.
archive_mode	on	Specify the archive log mode.
archive_command	'*installDir*/bin/pgx_walcopy.cmd "%p" "*backupDataStorageDestinationDirectory*/archived_wal/%f"'	Specify the command and storage destination to save the transaction log.
backup_destination	Backup data storage destination directory	Specify the name of directory where to store the backup data. Set the permissions so that only the instance administrator user can access the specified directory. Specify the same full path on all servers, so that the backup data of other servers can be used to perform recovery.
max_connections	Number of simultaneous client connections to the instance + superuser_reserved_connections	The value specified is also used to restrict the number of connections from client applications and the number of connections for the management of instances. Refer to "When an Instance was Created with the initdb Command" in the Installation and Setup Guide for Server, and "Connections and Authentication" in the PostgreSQL Documentation, for details.
superuser_reserved_connections	Add the number of simultaneous executions of mc_ctl status (*1) + 2	Specify the number of connections reserved for connections from database superusers. Add the number of connections from Mirroring Controller processes. Also reflect the added value in the max_connections parameter.
wal_receiver_timeout	Timeout (milliseconds)	Specify the time period after which it is determined that an error has occurred when the transaction log was received on the standby server. By aligning this value with the value for the heartbeat monitoring time of the database process, you can unify the time after which it is determined that an error has occurred.
restart_after_crash	off	If "on" is specified, or the default value is used for this parameter, behavior equivalent to restarting Fujitsu Enterprise Postgres, including crash recovery, will be performed when some server processes end abnormally.

Parameter	Content specified	Remarks
		However, when database multiplexing monitoring is used, a failover will occur after an error is detected when some server processes end abnormally, and the restart of those server processes is forcibly stopped. Specify "off" to prevent behavior such as this from occurring for no apparent reason.
synchronous_commit	on or remote_apply	Specify up to what position WAL send is to be performed before transaction commit processing returns a normal termination response to a client. Set "on" or "remote_apply" to prevent data loss caused by operating system or server down immediately after a switch or switch.
recovery_target_timeline	latest	Specify "latest" so that the new standby server (original primary server) will follow the new primary server when a switch occurs. This parameter is required when the original primary server is incorporated as a new standby server after the primary server is switched.

*1: Number of simultaneous executions of the mc_ctl command in the status mode.

2.4.3 Starting Mirroring Controller on the Primary Server

This section explains how to start Mirroring Controller on the primary server.

When the arbitration server is used for automatic degradation, start the Mirroring Controller arbitration process on the arbitration server in advance.

1. Start the Mirroring Controller process.

   Enabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode.

   Example)

   ```
   $ mc_ctl start -M /mcdir/inst1
   ```

   Disabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

   Example)

   ```
   $ mc_ctl start -M /mcdir/inst1 -F
   ```

Note

- When the arbitration server is used for automatic degradation, the database server must connect to the arbitration server, and as a result, Mirroring Controller startup may take longer than when the arbitration server is not used.
- If the parameter for heartbeat monitoring of operating systems or servers set by the abitration server is greater than parameter for heartbeat monitoring of operating systems and servers of the Mirroring Controller, the Mirroring Controller may fail to start. In this case, check the contents of the message notification and review the parameters for heartbeat monitoring of operating systems or servers for the arbitration server or Mirroring Controller.
- If the heartbeat_error_action parameter in *serverIdentifier*.conf is set to "message", even if automatic switch/disconnection is enabled and Mirroring Controller is started, only message output is performed when a heartbeat abnormality is detected during heartbeat monitoring of operating systems and servers - switch/disconnection is not performed.

- Mirroring Controller startup usually fails if the standby server is mistakenly started as the primary server or if the old primary server is not recovered after the switch and is then mistakenly started as the primary server. However, if the admin network is disconnected, then startup does not fail, and both servers may become primary servers. Therefore ensure that the admin network is connected before starting Mirroring Controller.

Point

- The mc_ctl command fails if the Mirroring Controller arbitration process has not been started on the arbitration server when the arbitration server is used for automatic degradation. However, if the Mirroring Controller arbitration process cannot be started in advance, it can be started by specifying the --async-connect-arbiter option in the mc_ctl command.
- After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

2. Obtain the backup.

   Use the pgx_dmpall command to collect the backup.

2.5 Setting Up the Standby Server

This section explains how to set up the standby server.

2.5.1 Setting Up Database Multiplexing Mode on the Standby Server

This section explains how to set up database multiplexing mode on the standby server.

In database multiplexing, the files that are required for operations are managed in the Mirroring Controller management directory.

There is one Mirroring Controller management directory for each instance.

Note

- Do not place the Mirroring Controller management directory in a directory managed by Fujitsu Enterprise Postgres, otherwise it may be deleted by mistake with the directories managed by Fujitsu Enterprise Postgres, and an old version of files may be restored.
- When creating a standby server for a large database, stop job system operations, specify a large value for the wal_keep_size parameter, or use replication slots.
  This is because WALs generated after the standby server is built using the pg_basebackup command, but before it is started, need to be retained. However, the number of WAL segments that can be retained is constrained by the wal_keep_size parameter.
  Additionally, setting the wal_keep_size parameter requires consideration regarding stabilization of the database multiplexing mode (refer to "2.11.1 Tuning to Stabilize the Database Multiplexing Mode" for details).

See

- Refer to "Preparing Directories for Resource Deployment" in the Installation and Setup Guide for Server for details on the directories that are managed by Fujitsu Enterprise Postgres.
- Refer to "pg_basebackup" in "Reference" in the PostgreSQL Documentation for information on the pg_basebackup command.
- Refer to "mc_ctl" in Reference for information on the command.
- Refer to "Appendix A Parameters" for details on each parameter to be edited for the setup.
- Refer to "Replication Slots" in the PostgreSQL Documentation for information on replication slots.

Perform the following procedure:

1. Log in to the standby server.

2. Create the Mirroring Controller management directory that will store the files required by database multiplexing.

   Use ASCII characters in the Mirroring Controller management directory.

   Additionally, grant "Write" permission to the instance administrator user for the Mirroring Controller management directory.

3. Copy, and then deploy, the network.conf file of the primary server.

   Copy the network.conf file that was defined in the primary server setup, and deploy it to the Mirroring Controller management directory of the standby server.

   Set read and write permissions for the instance administrator user only. If users other than the instance administrator user are granted access, the mc_ctl command will not work. Accordingly, users other than the instance administrator user are prevented from operating Mirroring Controller.

   Register in /etc/services the port number of the standby server that was specified in the network.conf file, because programs such as WebAdmin use it to search for available port numbers.

   Register any name as the service name.

4. Copy, and then deploy, the *serverIdentifier*.conf file of the primary server.

   Copy the *serverIdentifier*.conf file that was defined in the primary server setup, and deploy it to the Mirroring Controller management directory of the standby server.

   Set read and write permissions for the instance administrator user only. If users other than the instance administrator user are granted access permissions, the mc_ctl command will not work.

2.5.2 Creating, Setting, and Registering the Standby Server Instance

This section explains how to create, set, and register the standby server instance.

See

- Refer to "Appendix A Parameters" for details on each parameter.
- Refer to "mc_ctl" in Reference for information on the command.

Perform the following procedure:

1. Set the kernel parameters.

   Refer to "Configuring Kernel Parameters" in the Installation and Setup Guide for Server for details.

2. When using transparent data encryption, configure the encryption settings for the storage data.

   Refer to "Protecting Storage Data Using Transparent Data Encryption" or "Using Transparent Data Encryption with Key Management Systems as Keystores" in the Operation Guide for details, and then configure the settings.

3. Execute the pg_basebackup command to create a copy of the primary server instance on the standby server.

   Example)

```
$ pg_basebackup -D /database/inst1 -X fetch --waldir=/transaction/inst1 --progress --verbose -R
--dbname='application_name=standbyServerName' -h primaryServerIpAddress -p
primaryServerPortNumber
```

Note

- Use the pg_basebackup command with the -R option to create a standby.signal file. If you do not create the standby.signal file, the Mirroring Controller cannot be started as a standby server.
- If using a method that requires password authentication for connections to the primary server, you will need to ensure that authentication is performed automatically. If the -R option is specified for the pg_basebackup command and the password

parameter is specified for the --dbname option, the pg_basebackup command will set the password in the primary_conninfo parameter in postgresql.auto.conf file, enabling connections to be performed automatically.

If a password is not set in the primary_conninfo parameter in postgresql.auto.conf file, it will be necessary to create a .pgpass file in the home directory of the instance administrator user, and specify a password for the replication database.

- The primary_conninfo parameter should not be set in the postgresql.conf file, but only in the postgresql.auto.conf file using the pg_basebackup command.
- When executing the pg_basebackup command, consider the following for collection of transaction logs.
  - When "fetch" is specified for the -X option of the command

    Transaction logs are collected at the end of the backup, so it is necessary to ensure that transaction logs that occur during backup are not deleted from the primary server. Therefore, allow for a sufficient value for the wal_keep_size parameter in postgresql.conf.
  - When the -X option is omitted or "stream" is specified for the -X option of the command

    Transaction logs are streamed, so when Mirroring Controller is running on the primary server, the connection is changed to a synchronous standby server on detection of a streaming replication connection using this command. Therefore, if a job has started on the primary server, the primary server will be impacted, therefore execute this command after stopping only the Mirroring Controller process on the primary server.

See

Refer to "Hot Standby" in the PostgreSQL Documentation for information on the standby.signal file.

4. Set the parameters shown in the table below in the postgresql.conf file.

Table 2.4 Parameters

Parameter	Content specified	Remarks
synchronous_standby_names	'*primaryServerName*'	Required after switching the primary server and then changing the original primary server to the new standby server. Enclose the name in single quotation marks ('). Do not change this parameter while Mirroring Controller is running. Do not specify multiple names to this parameter as the Mirroring Controller can manage only one standby server.
streaming_wal_compression	Any number from -1 to 9	Specifies that you want to send a compressed WAL by streaming replication. -1: Compress at zlib's default compression level 0: Uncompressed 1-9: Compress at specified compression level, 9 is high compression Default is uncompressed.

See

Refer to "WAL Compression for Streaming Replication" in the Operation Guide for information on the streaming_wal_compression.

2.5.3 Starting Mirroring Controller on the Standby Server

This section explains how to start Mirroring Controller on the standby server.

When the arbitration server is used for automatic degradation, start the Mirroring Controller arbitration process on the arbitration server in advance.

1. After ensuring that the Mirroring Controller process of the primary server has started, start Mirroring Controller on the standby server.

   Enabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode with the -f option specified. This action enables automatic switch/disconnection.

   If you start Mirroring Controller and the instance without specifying the -f option, automatic switch/disconnection will not be enabled. To enable both, start Mirroring Controller and then execute the mc_ctl command in enable-failover mode or restart Mirroring Controller with the -f option specified.

   Example)

   ```
   $ mc_ctl start -M /mcdir/inst1
   ```

   Disabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

   Example)

   ```
   $ mc_ctl start -M /mcdir/inst1 -F
   ```

2. Check the status of the Mirroring Controller process.

   As the instance administrator user, execute the mc_ctl command in status mode. Ensure that "mirroring status" is switchable.

   Example)

   ```
   $ mc_ctl status -M /mcdir/inst1
   ```

Note

- When the arbitration server is used for automatic degradation, the time required for the database server to connect to the arbitration server is added on. Therefore, Mirroring Controller startup may take longer than when the arbitration server is not used.
- If the parameter for heartbeat monitoring of operating systems or servers set by the abitration server is greater than parameter for heartbeat monitoring of operating systems and servers of the Mirroring Controller, the Mirroring Controller may fail to start. In this case, check the contents of the message notification and review the parameters for heartbeat monitoring of operating systems or servers for the arbitration server or Mirroring Controller.
- If the heartbeat_error_action parameter in *serverIdentifier*.conf is set to "message", even if automatic switch/disconnection is enabled and Mirroring Controller is started, only message output is performed when a heartbeat abnormality is detected during heartbeat monitoring of operating systems and servers - switch/disconnection is not performed.
- Mirroring Controller startup usually fails if the standby server is mistakenly started as the primary server or if the old primary server is not recovered after the switch and is then mistakenly started as the primary server. However, if the admin network is disconnected, then startup does not fail, and both servers may become primary servers. Therefore, ensure that the admin network is connected before starting Mirroring Controller.

Point

- The mc_ctl command fails if the Mirroring Controller arbitration process has not been started on the arbitration server when the arbitration server is used for automatic degradation. However, if the Mirroring Controller arbitration process cannot be started in advance, it can be started by specifying the --async-connect-arbiter option in the mc_ctl command.
- After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

2.6 Creating a User Command for a Database Server

This section explains how to create a user command for a database server.

The following user commands are called by Mirroring Controller management processes.

The user can create user commands as required.

Specify the user commands that were created for the parameters in the server configuration file of the database server. Refer to "A.4.1 Server Configuration File for the Database Servers" for information on these parameters.

User command types

- Fencing command

  This user command performs fencing if Mirroring Controller performs arbitration processing and determines that a database server is unstable.

- Arbitration command

  This user command performs arbitration processing in lieu of the arbitration server when there is no arbitration server.

- State transition commands

  These user commands are called when Mirroring Controller performs state transition of a database server.
  It includes the following types:

  - Post-switch command

    This user command is called after a promotion from standby server to primary server.

  - Pre-detach command

    This user command is called before the standby server is disconnected from a cluster system.
    If the pre-detach command is specified on both the primary server and standby server, it is called first on the standby server and then on the primary server.

    If the settings are configured to forcibly stop the instance on the standby server when the standby server is disconnected, the pre-detach command is called on the standby server and then the instance on the standby server is stopped.

  - Post-attach command

    This user command is called after the standby server has been attached to a cluster system.
    If the post-attach command is specified on both the primary server and standby server, it is called first on the primary server and then on the standby server.

Point

When the arbitration server is used for automatic degradation and the requirements can be satisfied using the fencing command on the arbitration server only, the fencing command on the database server is not required. In addition, if the requirements can be satisfied using the fencing command on the database server only, create a fencing command on the arbitration server for termination processing only (without implementation).

Table 2.5 Availability of user commands, and database server calling the command

User command	Operation when a heartbeat abnormality is detected using operating system or server heartbeat monitoring				Database server calling the command	
	Message output	Automatic degradation by arbitration server	Automatic degradation by arbitration command	Unconditional automatic degradation	Primary server	Standby server
Fencing command	Y (*1)	Y (*2)	R	N	Y	Y

User command	Operation when a heartbeat abnormality is detected using operating system or server heartbeat monitoring				Database server calling the command	
	Message output	Automatic degradation by arbitration server	Automatic degradation by arbitration command	Unconditional automatic degradation	Primary server	Standby server
Arbitration command	N	N	R	N	Y	Y
Post-switch command	Y	Y	Y	Y	Y	N
Pre-detach command	Y	Y	Y	Y	Y	Y (*3)
Post-attach command	Y	Y	Y	Y	Y	Y (*3)

R: Required

Y: Can be used

N: Cannot be used

*1: Called only when the mc_ctl command is used to execute forced switching or forced disconnection.

*2: Creation of a fencing command on a database server is optional, but it must be created on the arbitration server.

*3: If message output or unconditional automatic degradation is selected, this command is called only from the primary server.

Refer to "Appendix C User Commands" for information on the interface for each user command.

2.7 Confirming the Streaming Replication Status

Before performing the setup of the database multiplexing mode, ensure that the prerequisite streaming replication feature has been set up correctly.

Perform the following procedure:

1. On the primary server, ensure that single-row searches can be performed using the pg_stat_replication statistics view.

   An example output of the psql command is shown below.

   Example)

```
postgres=# select * from pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
pid              | 10651
usesysid         | 10
usename          | fsep
application_name | standby
client_addr      | 192.0.2.210
client_hostname  |
client_port      | 55098
backend_start    | 2022-03-23 11:17:49.628793+09
backend_xmin     |
state            | streaming
sent_lsn         | 0/3000060
write_lsn        | 0/3000060
flush_lsn        | 0/3000060
replay_lsn       | 0/3000060
write_lag        |
flush_lag        |
replay_lag       |
```

```
sync_priority    | 1
sync_state       | sync
reply_time       | 2022-03-23 11:23:27.703366+09
```

2. Confirm the search results of step 1.

   Ensure that the connection established with the intended standby server is in synchronous mode.

   Table 2.6 Items to be checked

Item	Required value
application_name	Value specified for synchronous_standby_names parameter in the postgresql.conf file of the primary server.
client_addr	IP address of the standby server.
state	"streaming".
sync_state	"sync".

- Refer to "The Statistics Collector" in "Server Administration" in the PostgreSQL Documentation for information on the pg_stat_replication statistics view.
- Note that the pg_stat_replication statistics view may change in the future.

2.8 Checking the Connection Status

This section explains how to check the connection status from a database server or the arbitration server.

2.8.1 Checking the Connection Status on a Database Server

This section explains how to use a database server to check the connection status of the Mirroring Controller arbitration process and the Mirroring Controller process on the primary server and standby server.

Perform the following procedure:

1. On the primary server and standby server, execute the mc_ctl command in status mode with the --arbiter option specified.

   ```
   Example)
   The mc_ctl command is executed with the --arbiter option specified, and the status is output.
   ```

```
$ mc_ctl status --arbiter -M /mcdir/inst1

arbiter_id   host            status
--------------------------------------
arbiter      192.0.3.120     online
```

2. On the primary server and standby server, check the result displayed by executing the mc_ctl command in status mode in step 1.

   Items to be checked

   Check that the output status" is "online".

Refer to the Reference for information on the mc_ctl command.

2.8.2 Checking the Connection Status on the Arbitration Server

This section explains how to use the arbitration server to check the connection status of the Mirroring Controller arbitration process and the Mirroring Controller process on the primary server and standby server.

Perform the following procedure:

1. Execute the mc_arb command in status mode on the arbitration server.

   The example below executes the mc_arb command, and shows the status.

   Example)

```
$ mc_arb status -M /mcarb_dir/arbiter1

server_id    host            status
--------------------------------------
server1      192.0.3.100     online
server2      192.0.3.110     online
```

2. Check the result displayed by executing the mc_arb command in step 1.

   Items to be checked

   Check that the output status is "online" on both lines.

 See

Refer to the Reference for information on the mc_arb command.

2.9 Creating Applications

This section explains how to create applications using database multiplexing, and points that should be noted when you create the applications.

2.9.1 Application Connection Server Settings

If database multiplexing is used and a failover occurs, it will be necessary to switch the application connection server. Accordingly, use the application connection switch feature to create applications.

 See

Refer to "Application Connection Switch Feature" in the Application Development Guide for details.

2.10 Checking the Behavior

To check if the environment setup was performed correctly, start the application and then check the behavior of the switch and rebuild.

2.11 Tuning

This section explains how to tune database multiplexing mode.

2.11.1 Tuning to Stabilize the Database Multiplexing Mode

When large amounts of data are updated, the write-to load for the database will become great, and the multiplexing state may become unstable.

Accordingly, by editing the parameters below in the postgresql.conf file, a stable multiplexing state can be maintained. Refer to "Estimating Transaction Log Space Requirements" in the Installation and Setup Guide for Server for information on transaction log space requirements.

Table 2.7 Parameters

Parameter	Content
wal_keep_size	Refer to "2.4.2 Creating, Setting, and Registering the Primary Server Instance" for details.
max_wal_size	The transaction log is written out according to the checkpoint trigger. If a transaction log with the capacity of the value specified in this parameter is generated, the checkpoint will be executed. If a large value is specified in this parameter, the time required for crash recovery will increase. If a small value is specified in this parameter, many checkpoints will be generated, which will affect the performance of the applications that connect to the primary server.

2.11.2 Tuning to Stabilize Queries on the Standby Server

Queries made using reference jobs on the standby server may be canceled by jobs executed on the primary server.

To reduce the possibility of a job being canceled, specify as large a value as possible for the max_standby_archive_delay parameter in the postgresql.conf file.

 See

- Refer to "Handling Query Conflicts" in the PostgreSQL Documentation for details.
- Refer to "Standby Servers" in the PostgreSQL Documentation for details on the max_standby_archive_delay parameter.

2.11.3 Tuning to Stabilize Queries on the Standby Server (when Performing Frequent Updates on the Primary Server)

If jobs are updated on the primary server regularly and frequently, it will be easy for the query made by the reference job on the standby server to be canceled. In this case, edit one of the postgresql.conf file parameters shown in the table below.

Table 2.8 Parameters

Parameter	Description
hot_standby_feedback	When "on" is set, the deletion (vacuum) of the data area that was deleted or updated on the primary server is suppressed. Accordingly, the query on the standby server will not be canceled. (*1)
vacuum_defer_cleanup_age	The deletion (vacuum) of the data area that was deleted or updated on the primary server is delayed until the specified number of transactions is processed. Accordingly, the probability that the query on the standby server will be canceled decreases.

*1: Because the vacuum is delayed, the data storage destination disk space of the primary server comes under pressure.
Additionally, if there is conflict between accesses and queries executed on the standby server, transaction logs indicating this conflict will be transferred.
Accordingly, specify as large a value as possible for the max_standby_archive_delay parameter so that access conflicts do not occur.

 See

- Refer to "Standby Servers" in the PostgreSQL Documentation for details on the hot_standby_feedback parameter.
- Refer to "Primary Server" in the PostgreSQL Documentation for details on the vacuum_defer_cleanup_age parameter.

2.11.4 Tuning for Optimization of Degradation Using Abnormality Monitoring

Mirroring Controller uses a monitoring method that outputs an error if the timeout or number of retries is exceeded when accessing resources targeted for monitoring. Setting inappropriate values in these settings may lead to misdetection or a delay in automatic degradation, so you must design these values appropriately.

For example, the following type of issue occurs if the tuning related to abnormality monitoring is not performed appropriately.

- If the timeout is too short

  Results in redundant degradation and availability falls.

- If the timeout is too long

  It takes longer for automatic degradation to be performed even when an error affecting operational continuity occurs, potentially causing downtime.

You can optimize degrading operation by editing the values for the parameters in the server configuration file described below in accordance with the system. Refer to "A.4 Server Configuration File" for information on how to edit these parameters.

2.11.4.1 Tuning for Abnormality Monitoring of the Operating System or Server

Tuning for abnormal monitoring of the operating system or server depends on the operation when heartbeat abnormality is detected by the heartbeat monitoring of operating systems or servers.

See

Refer to "1.1.1 Monitoring Using Database Multiplexing Mode" for the operation when heartbeat abnormality is detected in the the heartbeat monitoring of operating systems or servers.

2.11.4.1.1 Tuning Abnormality Monitoring for Operations that Use an Arbitration Server for Automatic Degradation

In an operation that use an arbitration server for automatic degradation, the database server is periodically monitored for abnormalities so that the Mirroring Controller arbitration process can immediately respond to an arbitration request from the Mirroring Controller.The automatic degradation using the arbitration server can optimize the time from error detection to automatic degradation of the operating systems or servers by editing the following parameters.

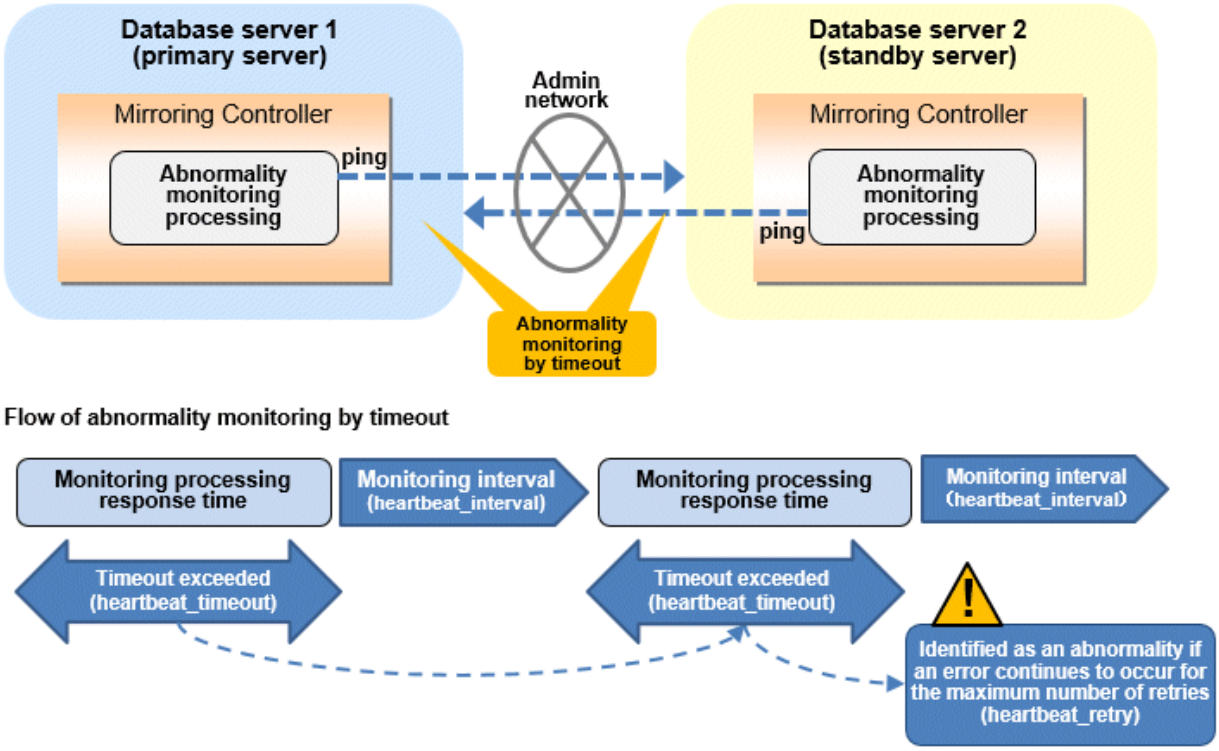
- Parameters for the abnormality monitoring of the operating system or server in the server configuration file of the database server
- Parameters for the abnormality monitoring of the operating system or server in the arbitration configuration file
- Parameters for the arbitration processing and fencing

Parameters for the abnormality monitoring of the operating system or server in the server configuration file of the database server

Table 2.9 Parameters for the abnormality monitoring of the operating system or server in the server configuration file of the database server

Parameter	Description
Abnormality monitoring interval (heartbeat_interval)	Mirroring Controller is configured so that abnormality monitoring does not place a load on the system. This parameter does not normally need to be set. (The default is 800 milliseconds.)
Abnormality monitoring timeout (heartbeat_timeout)	Take into account the time during which a load is placed continuously on the server or admin network performance. For example, it is envisaged that this parameter will be used in situations such as when performing high-load batch jobs or when a large number of online jobs occur continuously and concurrently. (The default is 1 second.)

Parameter	Description
Abnormality monitoring retries (heartbeat_retry)	This parameter can be set when needing a safety value for situations in which the value specified for heartbeat_timeout is exceeded, for example, when using systems with fluctuating loads, however, this parameter does not normally need to be set. (The default is 2 times.)



The expression for calculating the time required to detect an abnormality by Mirroring Controller is shown below.

```
Abnormality detection time of Mirroring Controller = ( heartbeat_timeout(seconds) +
heartbeat_interval(milliseconds) / 1000 ) x ( heartbeat_retry(number of times) + 1)
```

The abnormality detection time when the default value is used is shown below.

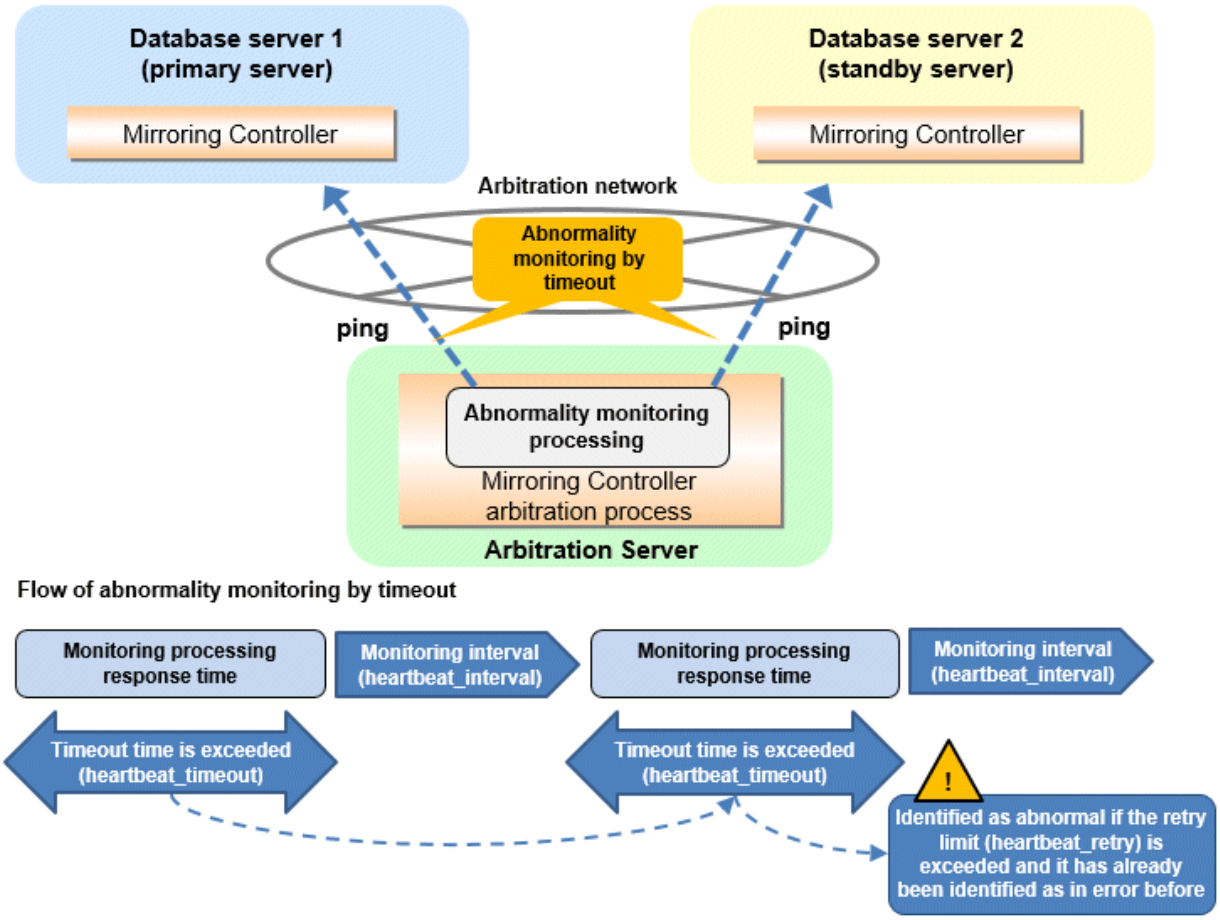
```
Abnormality detection time of Mirroring Controller = ( 1 + 800 / 1000 ) x ( 2 + 1 )
            = 5.4(seconds)
```

Parameters for the abnormality monitoring of the operating system or server in the arbitration configuration file

Table 2.10 Parameters for the abnormality monitoring of the operating system or server in the arbitration configuration file

Parameter	Description
Abnormality monitoring interval (heartbeat_interval)	Mirroring Controller arbitration process is configured so that abnormality monitoring does not place a load on the system. This parameter does not normally need to be set. (The default is the value set in heartbeat_interval in the server configuration file of the database server.) (milliseconds).
Abnormality monitoring timeout (heartbeat_timeout)	Take into account the time during which a load is placed continuously on the server and arbitration network capabilities. (The default is the value set in heartbeat_timeout in the server configuration file of the database server.) (seconds).

Parameter	Description
Abnormality monitoring retries (heartbeat_retry)	This parameter can be set when needing a safety value for situations in which the value specified for heartbeat_timeout is exceeded, for example, when using systems with fluctuating loads, however, this parameter does not normally need to be set. (The default is the value set in heartbeat_retry in the server configuration file of the database server.) (number of times)



The expression for calculating the time required to detect an abnormality by Mirroring Controller arbitration process is shown below.

```
Abnormality detection time of Mirroring Controller arbitration process = ( heartbeat_timeout(seconds)
+ heartbeat_interval(milliseconds) / 1000 ) x ( heartbeat_retry(number of times) + 1)
```

The abnormality detection time when the default value is used is shown below.

```
Abnormality detection time of Mirroring Controller arbitration process = ( 1 + 800 / 1000 ) x ( 2 + 1 )
            = 5.4(seconds)
```

Point

The abnormality detection time of the operation for automatic degradation using the arbitration server can be calculated as follows.

```
Abnormality detection time = Max(Abnormality detection time by Mirroring Controller, Abnormality
detection time by Mirroring Controller arbitration process)
```

Note

If the heartbeat_interval is set in the arbitration configuration file, the relationship between the parameter for operating system or server abnormality monitoring specified in the server configuration file of the database server file and the heartbeat_interval of the arbitration configuration file must satisfy the following relational expression.

```
 Heartbeat_interval in the arbitration configuration file (milliseconds) / 1000 ) <
( heartbeat_timeout(seconds) + heartbeat_interval(milliseconds) / 1000 ) * heartbeat_retry(number of
times) + heartbeat_timeout(seconds)
```

Parameters for the arbitration processing and fencing

Table 2.11 Parameters for the arbitration processing and fencing

Parameter	Description
Arbitration processing timeout (arbitration_timeout in the server configuration file of the database server)	Take into account the time to perform arbitration processing on the Mirroring Controller arbitration process. The value must be greater than or equal to abnormality detection time of Mirroring Controller arbitration process + fencing_command_timeout in the arbitration configuration file (seconds).
Fencing timeout (fencing_command_timeout in the arbitration configuration file)	Take into account the time to execute the fencing command (seconds).

Flow from the abnormality detection to the automatic degeneracy

When performing automatic degradation using the arbitration server, the flow from the abnormality detection in the operating system or server to the occurrence of automatic degeneracy and the parameters is shown below.

<table>
<tr><th>Flow from the abnormality detection to the automatic degeneracy</th><th>Description</th><th colspan="2">Parameter</th></tr>
<tr><td>(1) Abnormality detection</td><td>Mirroring Controller detect the database server operating system or server errors.</td><td colspan="2">Parameters for the abnormality monitoring of the operating system or server in the server configuration file of the database server</td></tr>
<tr><td>(2) Arbitration request</td><td>Mirroring Controller that detect the operating system or server error asks the Arbitration Server to check the status of the other server's operating system or server.</td><td>-</td><td rowspan="3">arbitration_timeout in the server configuration file of the database server</td></tr>
<tr><td>(3) Arbitration processing</td><td>The Mirroring Controller arbitration process checks the status of the other server's operating system or server. However, if the result of the operating system or server abnormality monitoring by the arbitration server has been determined before the arbitration request from the Mirroring Controller of the database server, this process is not performed.</td><td>Parameters for the abnormality monitoring of the operating system or server in the arbitration configuration file</td></tr>
<tr><td>(4) Fencing</td><td>If the Mirroring Controller arbitration process determines that the other server is an abnomaly of the operating system or server, it fences the other server and isolates it from the cluster system. If the Mirroring Controller arbitration process determines that the operating system or server</td><td>fencing_command_timeout in the arbitration configuration file</td></tr>
</table>

Flow from the abnormality detection to the automatic degeneracy	Description	Parameter	
	status is normal, this process and the (6) are not performed.		
(5) Return of the arbitration results	Returns the results of the arbitration to the Mirroring Controller of the database server that requested the arbitration.	-	
(6) Automatic degradation	The automatic degradation is performed. If fencing fails in (4), this procedure is not performed.	-	

-: No associated parameters

Note

If the fencing_command parameter is specified in the server configuration file of the database server, the fencing command is invoked on the database server if fencing is successful on the arbitration server. In that case, add the value of the fencing_command_timeout parameter in the server configuration file of the database server to the estimate.

Figure 2.1 When the Mirroring Controller on the primary server detects an operating system or server error

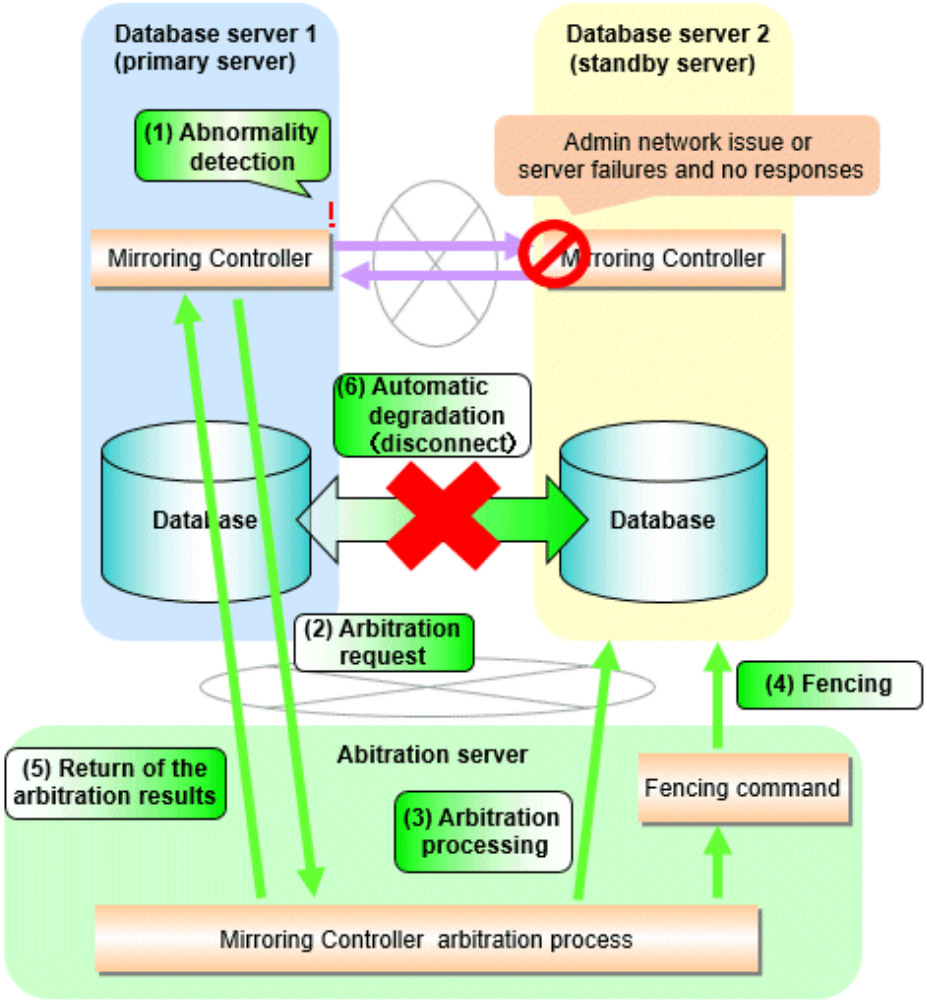
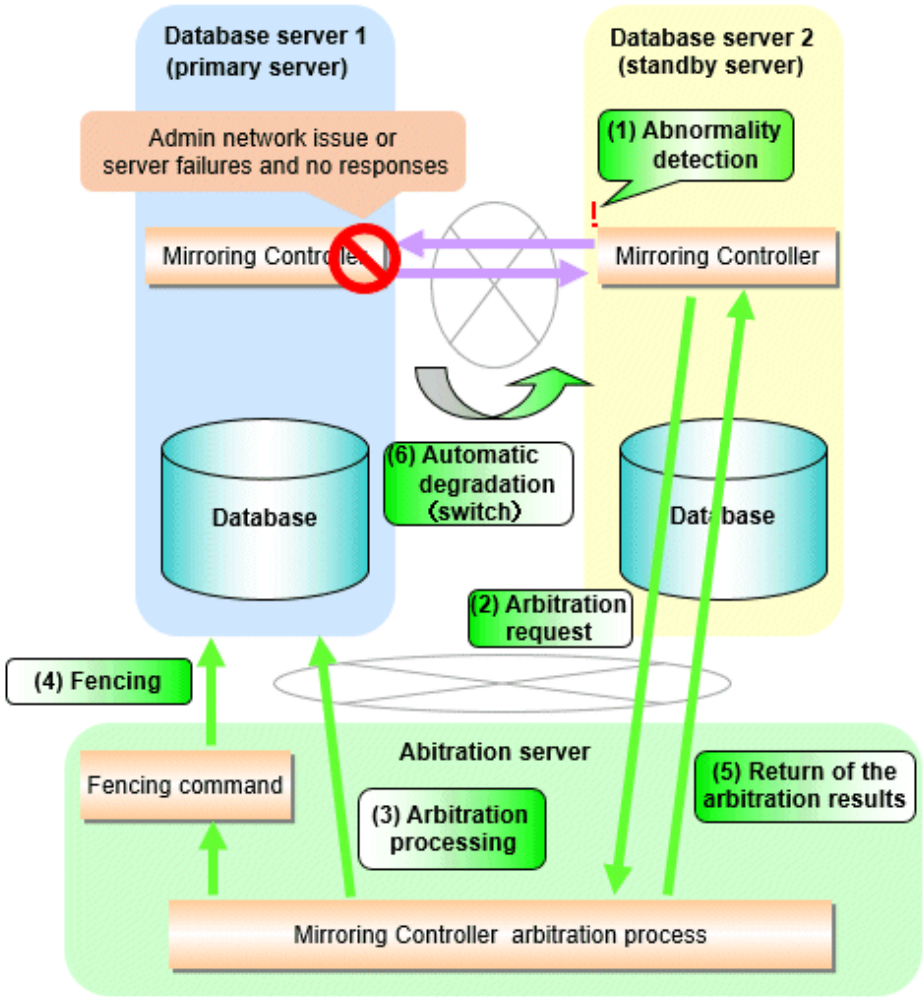


Figure 2.2 When the Mirroring Controller on the standby server detects an operating system or server error



2.11.4.1.2 Tuning Abnormality Monitoring for Operations that Perform Automatic Degradation by Calling a User Command that Determines Degradation

In an operation that perform automatic degradation by calling a user command that determines degradation, you can optimize the time from operating system or server abnormality detection to automatic degradation by editing the operating system or server abnormality monitoring parameters and parameters related to arbitration processing and fencing in the server configuration file of the database server. Refer to "Parameters for the abnormality monitoring of the operating system or server in the server configuration file of the database server" for information on the operating system or server abnormality monitoring parameters in the server configuration file of the database server.

Table 2.12 Parameters for the arbitration processing and fencing

Parameter	Description
Arbitration processing timeout (arbitration_command_timeout)	Take into account the time to execute the arbitration command(seconds).
Fencing timeout (fencing_command_timeout)	Take into account the time to execute the fencing command (seconds).

Flow from the abnormality detection to the automatic degeneracy

When performing automatic degradation by calling a user command that determines degradation, the flow from the abnormality detection in the operating system or server to the occurrence of automatic degeneracy and the parameters is shown below.

Flow from the abnormality detection to the automatic degeneracy	Description	Parameter
(1) Abnormality detection	Mirroring Controller detect the database server operating system or server errors.	Parameters for the abnormality monitoring of the operating system or server in the server configuration file of the database server
(2) Arbitration processing	An arbitration command is executed to check the status of the other server's operating system or server.	arbitration_command_timeout in the server configuration file of the database server
(3) Fencing	If the operating system or server status of the other server is abnomal in (2), it fences the other server and isolates it from the cluster system. If the operating system or server status of the other server is nomal in (2), this process and (4) are not executed.	fencing_command_timeout in the server configuration file of the database server
(4) Automatic degradation	The automatic degradation is performed. If fencing fails in (3), this procedure is not performed.	-

-: No associated parameters

Figure 2.3 When the Mirroring Controller on the primary server detects an operating system or server error

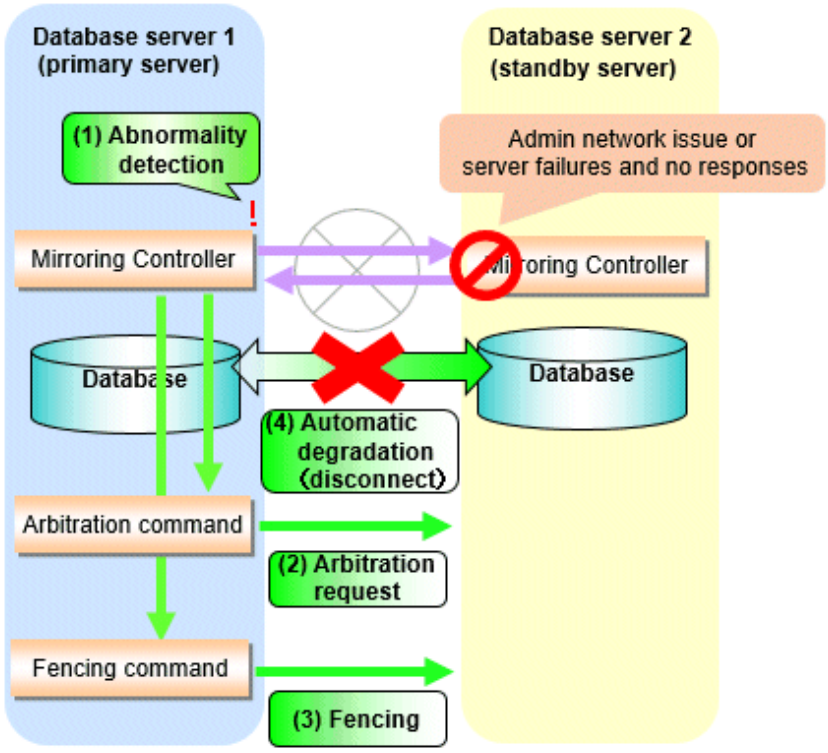
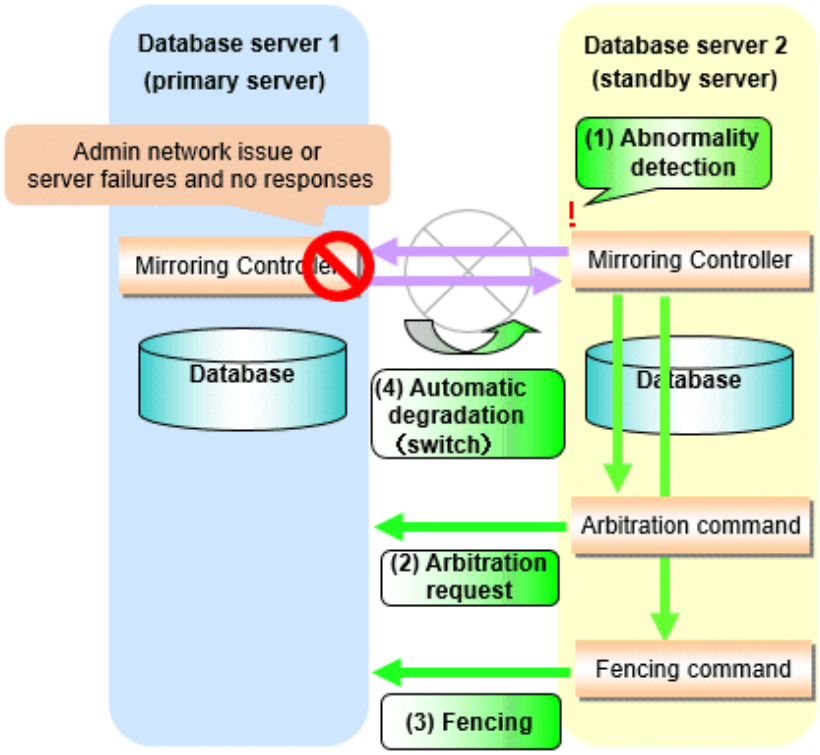


Figure 2.4 When the Mirroring Controller on the standby server detects an operating system or server error



2.11.4.1.3 Tuning Abnormality Monitoring for Operations that Notify Messages

In an operation that notify messages, you can optimize the abnormality detection time by editing the operating system or server abnormality monitoring parameters in the server configuration file of the database server. Refer to "Parameters for the abnormality monitoring of the operating system or server in the server configuration file of the database server" for information on the operating system or server abnormality monitoring parameters in the server configuration file of the database server. In addition, when the Mirroring Controller detects an error, it does not perform the arbitration processing, fencing, or automatic degradation, but only notification messages is performed.

2.11.4.1.4 Tuning Abnormality Monitoring for Operations that Perform Automatic Degenerate Unconditionally due to Heartbeat Abnormality

In an operation that perform automatic degenerate unconditionally due to heartbeat abnormality, you can optimize the time from operating system or server abnormality detection to automatic degradation by editing the operating system or server abnormality monitoring parameters in the server configuration file of the database server. Refer to "Parameters for the abnormality monitoring of the operating system or server in the server configuration file of the database server" for information on the operating system or server abnormality monitoring parameters in the server configuration file of the database server. In addition, when the Mirroring Controller detects an error, it does not perform the arbitration processing, fencing, or automatic degradation, but only automatic degenerate unconditionally is performed.

Note

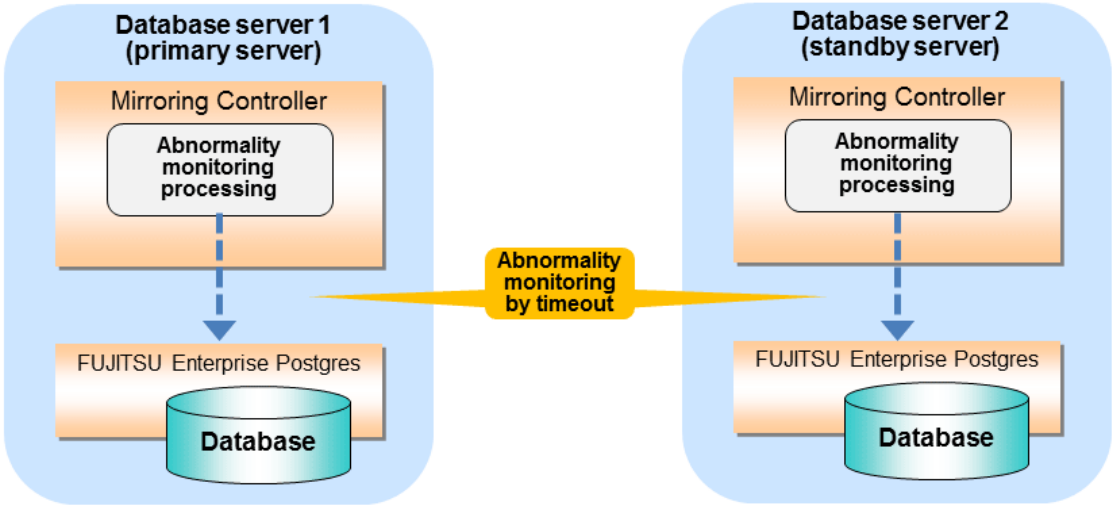
Refer to "Appendix D Notes on Performing Automatic Degradation Immediately after a Heartbeat Abnormality" for notes on the operation that perform automatic degenerate unconditionally due to heartbeat abnormality.

2.11.4.2 Tuning for Abnormality Monitoring of Darabase Processes

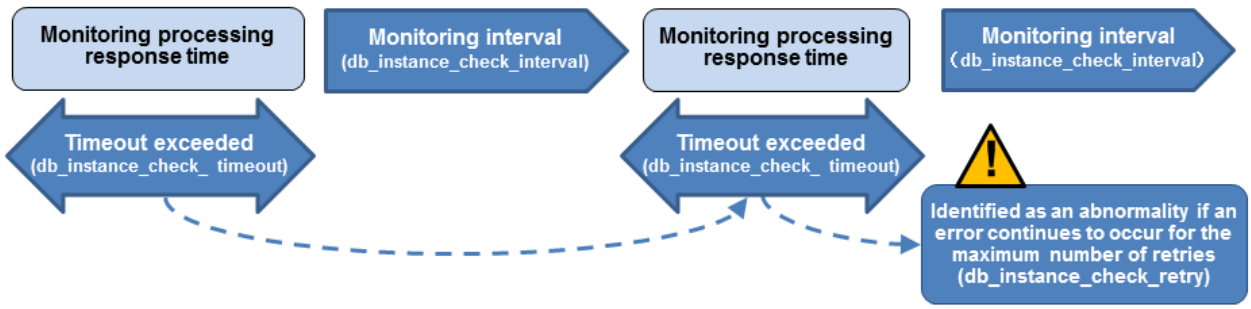
You can optimize database pocesses abnormality monitoring by editing the following parameters in the server configuration file of the database server.

Table 2.13 Parameters for abnormality monitoring of database processes

Parameter	Description
Abnormality monitoring interval (db_instance_check_interval)	Abnormality monitoring by Mirroring Controller is set so as not to place load on the system, but normally it does not need to be set. (The default is the value set in heartbeat_interval.) (milliseconds)
Timeout for abnormality monitoring of database processes (db_instance_check_timeout)	Take into account the time during which a load is placed continuously on the database. For example, it is envisaged that this parameter will be used in situations such as when performing high-load batch jobs or when a large number of online jobs occur continuously and concurrently. (The default is the value set in heartbeat_timeout.) (seconds)
Abnormality monitoring retries (db_instance_check_retry)	This parameter can be set when needing a safety value for situations in which the value specified for db_instance_check_timeout is exceeded, for example, when using systems with fluctuating loads, however, this parameter does not normally need to be set. (The default is the value set in heartbeat_retry.) (number of times)



Flow of abnormality monitoring by timeout



The expression for calculating the time required to detect an abnormality is shown below.

```
Abnormality detection time = ( db_instance_check_timeout(seconds) +
db_instance_check_interval(milliseconds) / 1000 ) x ( db_instance_check_retry(number of times) + 1 )
```

The abnormality detection time when the default value is used is shown below.

```
Abnormality detection time = ( 1 + 800 / 1000 ) x ( 2 + 1 )
             = 5.4(seconds)
```

Note

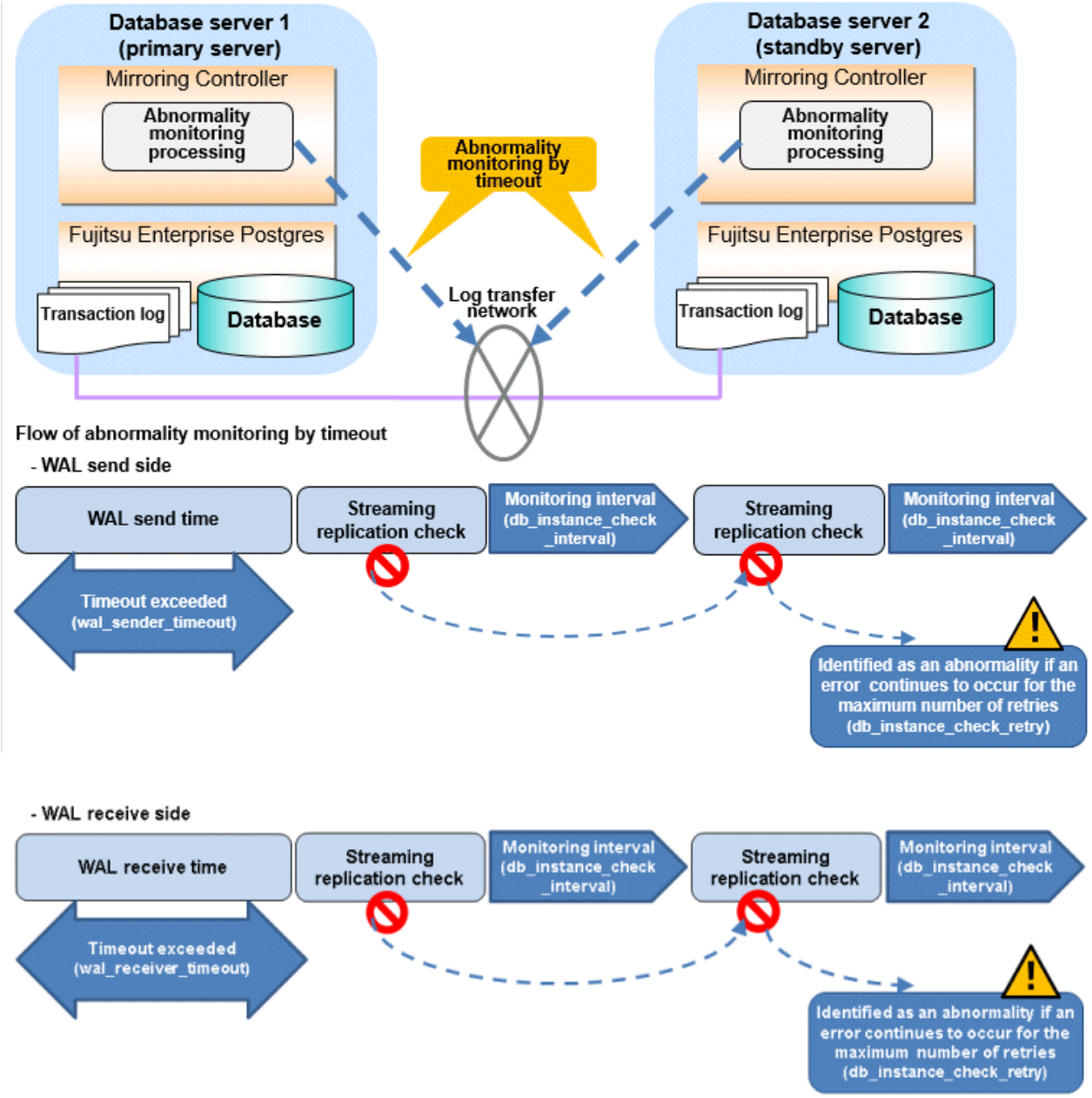
- If the db_instance_timeout_action parameter in *serverIdentifier*.conf is set to "message", and the db_instance_check_timeout parameter is set to a short value, a crash of the database process will be detected as "no response", and it may take time for automatic degradation to occur. Therefore, specify an appropriate timeout for db_instance_check_timeout.
- If a high load on the database and an event that prevents connection to an instance occur at the same time, it is judged as abnormal without retrying monitoring.

2.11.4.3 Tuning for Abnormality Monitoring of Streaming Replication

You can optimize streaming replication abnormality monitoring by editing the following parameters in the server configuration file of the database server.

Table 2.14 Parameters for abnormality monitoring of streaming replication

Parameter	Description
Abnormality monitoring interval (db_instance_check_interval)	Abnormality monitoring by Mirroring Controller is set so as not to place load on the system, but normally it does not need to be set. (The default is the value set in heartbeat_interval.) (milliseconds)
Abnormality monitoring retries (db_instance_check_retry)	This parameter can be set when needing a safety value, such as when it is anticipated that a temporary log transfer LAN error may occur, but it does not normally need to be set. (The default is the value set in heartbeat_retry.) (number of times)
Timeout for abnormality monitoring of streaming replication (wal_sender_timeout and wal_receiver_timeout in postgresql.conf)	Take into account the capacity and load of the log transfer network and the time during which a load is placed continuously on the database. For example, if there is a succession of data update jobs that generate a high WAL volume, you must configure the settings to avoid misdetection. (The default is 60 seconds.)



The expression for calculating the time required to detect an abnormality is shown below.

```
Abnormality detection time = ( wal_sender_timeout(seconds) +
db_instance_check_interval(milliseconds) / 1000 x ( disk_check_retry(number of times) + 1 ) )  Or,
            = ( wal_receiver_timeout(seconds) + db_instance_check_interval(milliseconds) / 1000 x
( disk_check_retry(number of times) + 1 ) )
```

The abnormality detection time when the default value is used is shown below.

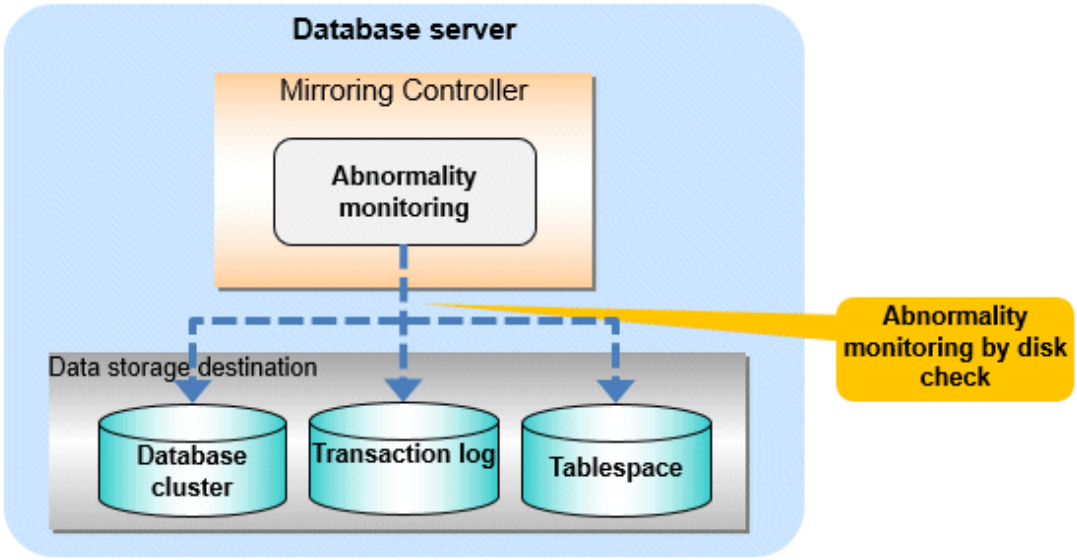
```
Abnormality detection time = 60 + (800 / 1000 x (2 + 1))
            = 62.4(seconds)
```

2.11.4.4 Tuning for Disk Abnormality Monitoring

You can optimize disk abnormality monitoring by editing the following parameters in the server configuration file of the database server.

Table 2.15 Parameters for disk abnormality monitoring

Parameter	Description
Abnormality monitoring interval (disk_check_interval)	Abnormality monitoring by Mirroring Controller is set so as not to place load on the system, but normally it does not need to be set. Set a value larger than the disk access time. (The default is the value set in heartbeat_interval.) (milliseconds)
Abnormality monitoring retries (disk_check_retry)	This parameter can be set when needing a safety value, such as when it is anticipated that a temporary disk input/output error may occur, but normally it does not need to be set. (The default is the value set in heartbeat_retry.) (number of times)
Abnormality monitoring timeout time (disk_check_timeout)	The time allowed from the start time of the next disk_check_interval after a disk error occurs until the error is determined to be due to timeout. (The default is 2147483.) (seconds). You can specify an integer between 0 and 2147483.
Upper limit on the number of threads used for abnormality monitoring (disk_check_max_threads)	Upper limit on the number of threads for disk monitoring. (The default is the number of processors available to the JVM.) You can specify an integer between 1 and 2147483647, but setting a value greater than the threads available on the machine may result in a system error. When you run the mc_ctl status command separately from the monitoring process, each mc_ctl status temporarily uses the same number of threads as the monitoring process. When setting disk_check_max_threads, consider the machine's thread limit, the number of tablespaces you plan to use, and the number of mc_ctl status commands that may be executed at the same time.



In disk error monitoring, a disk check is performed, and degradation is performed when an error is first detected within the error detection time or the time set in disk_check_timeout. However, in order to disconnect the standby server when disk_check_timeout detects an error on the standby server, shutdown_detached_synchronous_standby must be set to on.

The following shows how to calculate the abnormality detection time (the time until an error is determined).

```
Abnormality detection time = disk_check_interval (milliseconds) / 1000 x ( disk_check_retry(number of
times) + 1 )
```

The abnormality detection time when the default value is used is shown below.

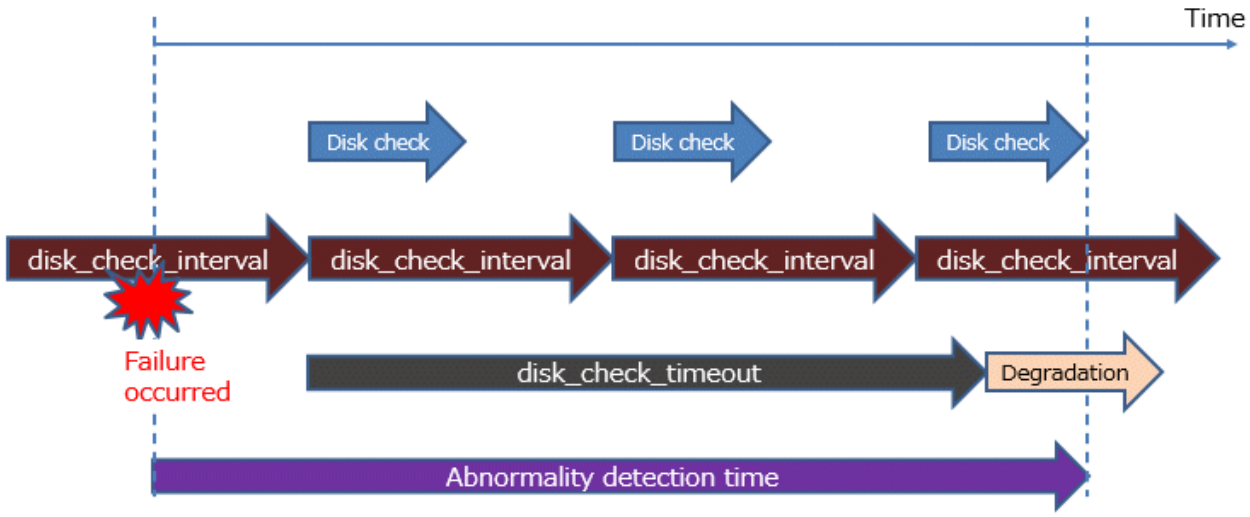
```
Abnormality detection time = 800 / 1000  x ( 2 + 1 )
            = 2.4(seconds)
```

An example of detecting disk abnormality monitoring is shown below.

Example 1)

One thread monitors one disk, set disk_check_retry = 2.

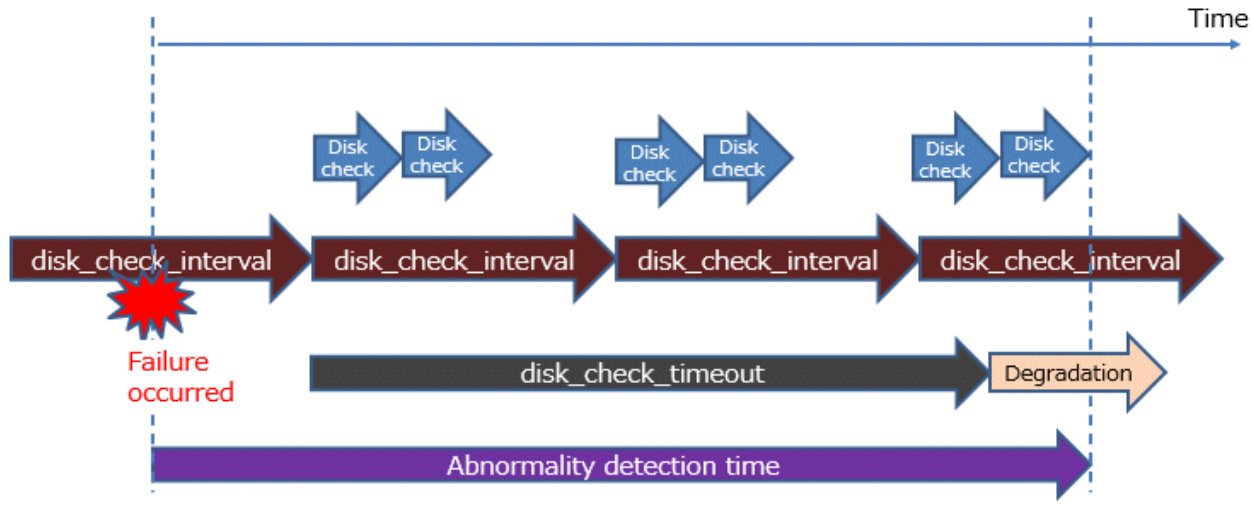
When the read or write cannot be completed within the disk access time because the response to the disk is slow.



Example 2)

One thread monitors two disks, set disk_check_retry = 2.

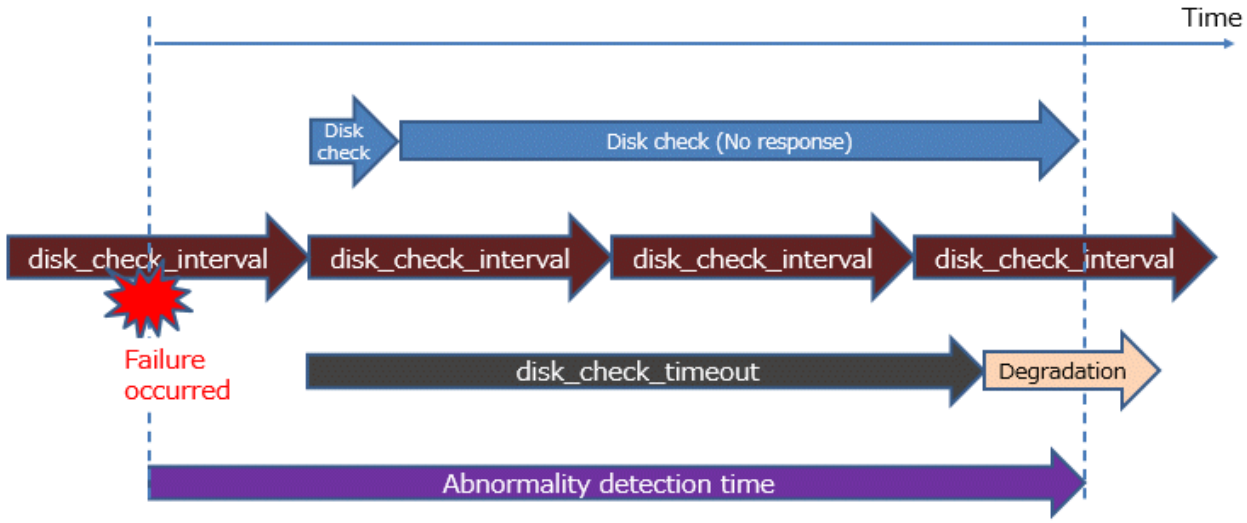
Due to the slow response to the disk, reading or writing to disk 1 could not be completed within the disk access time on the first and second attempts, but the reading or writing was successful on the third retry. All reads or writes to disk 2 fail within the disk access time.



Example 3)

One thread monitors two disks, set disk_check_retry = 2.

If disk 2 becomes unresponsive and monitoring results cannot be obtained within the timeout period.



Note

- The tuning described above impacts on the time taken from detection of a timeout until switching the primary server. Therefore, modify the values while taking into account the switch/disconnection time, using a design for which misdetection does not occur.
- Immediately selecting automatic degradation when a heartbeat abnormality occurs in operating system or server heartbeat monitoring risks causing split brain. Refer to "Appendix D Notes on Performing Automatic Degradation Immediately after a Heartbeat Abnormality" for details.

Information

Mirroring Controller uses connections to database instances and SQL access to monitor abnormality in some resources targeted for monitoring. The connection destination database names and connection user names used for abnormality monitoring conform to the parameters in the server configuration file. The application name is "mc_agent".

2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances

Multiplexed instances and Mirroring Controller can be started and stopped automatically in line with the starting and stopping of the operating system of the database server.

Note

To guarantee the startup sequence of Mirroring Controller on the primary and standby servers, first confirm that the primary server has started, and then start the standby servers in sequence.

The startup sequence of the Mirroring Controller process on the database server and the Mirroring Controller arbitration process on the arbitration server is not guaranteed. If the arbitration server cannot be started first, execute the mc_ctl command in start mode with the --async-connect-arbiter option specified to start the Mirroring Controller process.

If you start the Mirroring Controller and multiplexed instances, wait for time correction, network setup, and so on.

Perform the following procedure:

1. Create a unit file

   Copy the unit file sample stored in the directory below, and revise it to match the target instance.

Sample file

```
/installDir/share/mcoi.service.sample
```

Example)

In the following example, the installation directory is "/opt/fsepv<x>server64", and the instance name is "inst1". Note that "<x>" indicates the product version.

```
# cp /opt/fsepv<x>server64/share/mcoi.service.sample /usr/lib/systemd/system/mcoi_inst1.service
```

Revise the underlined portions of the options below in the unit file.

Section	Option	Specified value	Description
Unit	Description	Fujitsu Enterprise Postgres MirroringController *instanceName*	Specifies the feature overview. Specifies the name of the target instance. (*1)
Service	ExecStart	/bin/bash -c '*installDir*/bin/mc_std start *installDir* *MirroringControllerManagementDir*' *mc_ctlOption*'	Command to be executed when the service is started. Specify the option you want to add when the mc_ctl command is executed without the -M option in the mc_ctl option. Note that the content specified in this mc_ctl option is carried over from the mc_std command to the mc_ctl command. (*2)
	ExecStop	/bin/bash -c '*installDir*/bin/mc_std stop *installDir* *MirroringControllerManagementDir*' *mc_ctlOption*'	Command to be executed when the service is stopped. Specify the option you want to add when the mc_ctl command is executed without the -M option in the mc_ctl option. However, to use the --mc-only option to stop only Mirroring Controller, you must use the --mc-only option at startup. Note that the content specified in this mc_ctl option is carried over from the mc_std command to the mc_ctl command.
	User	User	OS user account of the instance administrator user.
	Group	Group	Group to which the instance administrator user belongs.

*1: The instance name should be as nameThatIdentifiesTheInstance.
The naming conventions for identifying the instance are as follows:

- Up to 16 bytes
- The first character must be an ASCII alphabetic character
- The other characters must be ASCII alphanumeric characters

*2: When the arbitration server is used for automatic degradation, start the Mirroring Controller arbitration process on the arbitration server and then start the Mirroring Controller process on the database server. If the arbitration server cannot be started first, specify the --async-connect-arbiter option to start the Mirroring Controller process.

2. Enable automatic start and stop

   As the OS superuser, use the systemctl command to enable automatic start and stop.

   Example)

```
# systemctl enable mcoi_inst1.service
```

Note

If automatic start and stop of Mirroring Controller has been configured, to stop Mirroring Controller, do not use the mc_ctl command, but instead use the systemctl command as the OS superuser.

Example)

```
# systemctl stop mcoi_inst1.service
```

If the instance does not stop, refer to "Actions in Response to Failure to Stop an Instance" in the Operation Guide to stop the instance. Then, specify the -e option in the mc_ctl command to forcibly stop Mirroring Controller.

Example)

```
$ mc_ctl stop -M /mcdir/inst1 -e
```

If Mirroring Controller is stopped using the mc_ctl command, the message below is output to the system log, however there is no issue because automatic stop is executed by systemd.

Message

```
FATAL: failed to stop Mirroring Controller target server:"{0}" (MCA00043)
```

2.13 Setting Automatic Start and Stop of the Mirroring Controller Arbitration Process

You can automatically start or stop the Mirroring Controller arbitration process when the operating system on the arbitration server is started or stopped.

Note

If you start the Mirroring Controller arbitration process, wait for time correction, network setup, and so on.

Perform the following procedure:

1. Create a unit file.

   Copy the unit file sample stored in the directory below, and revise it to match the target instance.

   Sample file

   ```
   /installDir/share/mcarboi.service.sample
   ```

   Example)
   In the following example, the installation directory is "/opt/fsepv<x>assistant", and the identifier of the arbitration process is "arbiter1". Note that "<x>" indicates the product version.

   ```
   # cp /opt/fsepv<x>assistant/share/mcarboi.service.sample /usr/lib/systemd/system/
   mcarboi_arbiter1.service
   ```

   Revise the underlined portions of the options below in the unit file.

Section	Option	Specified value	Description
Unit	Description	Fujitsu Enterprise Postgres Mirroring Controller Arbiter *<arbitrationProcessId>*	Specifies the feature overview.

Section	Option	Specified value	Description
			Specifies the identifier of the targeted arbitration process. (*1)
Service	ExecStart	/bin/bash -c '*installDir*/bin/mc_arb_std start *installDir mirroringControllerArbitrationProcessMgmtDir mc_arbOption*'	Command to be executed when the service is started. Specify the option you want to add when the mc_arb command is executed without the -M option in the mc_arb option. Note that the content specified in this mc_arb option is carried over from the mc_arb_std command in "Specified value" to the mc_arb command.
	ExecStop	/bin/bash -c '*installDir*/bin/ mc_arb_std stop *installDir mirroringControllerArbitrationProcessMgmtDir mc_arbOption*'	Command to be executed when the service is stopped. Specify the option you want to add when the mc_arb command is executed without the -M option in the mc_arb option. Note that the content specified in this mc_arb option is carried over from the mc_arb_std command in "Specified value" to the mc_arb command.
	User	User	Specify the account of the operating system user.
	Group	Group	Specify the group to which the user belongs.

*1: The arbitration process identifier used here is a name for identifying the Mirroring Controller arbitration process.

The naming conventions for identifying the Mirroring Controller arbitration process are as follows:

- Up to 16 bytes
- The first character must be an ASCII alphabetic character
- The other characters must be ASCII alphanumeric characters

2. Enable automatic start and stop.

As the operating system superuser, use the systemctl command to enable automatic start and stop.

Example)

```
# systemctl enable mcarboi_arbiter1.service
```

2.14 Backup Operation

This section explains the backup operation for database multiplexing mode.

2.14.1 Backing up Database Multiplexing Mode Information

When changing the Mirroring Controller settings, in addition to backing up the database, back up the configuration file in the Mirroring Controller management directory so that the Mirroring Controller settings are not lost.

When the arbitration server is used for automatic degradation, also back up the configuration file in the Mirroring Controller arbitration process management directory.

2.14.2 Database Backup Operation

Using database multiplexing mode is the same as obtaining the backup data on the standby server as a safeguard against a disk failure. Note that all server disks may be corrupted due to some cause.

As a safeguard against this type of case, execute the pgx_dmpall command on the primary server to create the backup data.

However, it is not definite as to which server runs as the primary server, so ensure that the pgx_dmpall command is executed periodically on all servers, so that the backup data will be obtained. For example, create a script to obtain the backup data, and set it in the operation management software.

Point

When the pgx_dmpall command is executed on the standby server, it will not match the statuses, however the error message shown below will be output and return the value "1".

If a script that ignores only this type of error is executed on all servers, the backup data of the primary server can be obtained.

```
Error message
```

```
ERROR:recovery is in progress (10095)
```

Note

- Consider the possibility that the server that runs as the primary server may be destroyed alongside the backup data, so it is recommended to promote another server to become the primary server, and then back up the data on the new primary server without waiting for the next scheduled backup.
- Specify the same backup directory name for the primary and standby servers. If different backup directory names are specified, and recovery is performed using the backup data of the other server, the recovery cannot be performed correctly.

See

- Period backups allow shorter recovery time and reduction in disk usage. Refer to "Backing Up the Database" in the Operation Guide for details on the backup operation.
- Refer to "Chapter 4 Action Required when an Error Occurs in Database Multiplexing Mode" for details on recovery based on the backup data that was obtained using the pgx_dmpall command.

Chapter 3 Operations in Database Multiplexing Mode

This chapter describes the periodic operations that are performed when running database multiplexing mode.

The periodic operations are the same as the operations on a single server.

See

Refer to "Periodic Operations" in the Operation Guide for information on the periodic operations.

3.1 Starting and Stopping the Mirroring Controller Arbitration Process

This section describes how to start and stop the Mirroring Controller arbitration process.

3.1.1 Starting the Mirroring Controller Arbitration Process

While the Mirroring Controller arbitration process is in a stopped state, execute the mc_arb command in start mode to start the Mirroring Controller arbitration process.

Example)

```
$ mc_arb start -M /mcarb_dir/arbiter1
```

See

Refer to the Reference for information on how to specify the mc_arb command.

3.1.2 Stopping the Mirroring Controller Arbitration Process

While the Mirroring Controller arbitration process is running, execute the mc_arb command in stop mode to stop the Mirroring Controller arbitration process.

Example)

```
$ mc_arb stop -M /mcarb_dir/arbiter1
```

See

Refer to the Reference for information on how to specify the mc_arb command.

Note

- The arbitration server will be forcibly stopped when the service is stopped.
- Before shutting down the operating system on the arbitration server, either stop the Mirroring Controller on the primary server or standby server or shut down the operating system on the primary server or standby server.

3.2 Starting and Stopping Mirroring Controller

When database multiplexing mode is used, use the mc_ctl command to start and stop the instance and Mirroring Controller at the same time.

Do not start or stop the instance by itself.

Starting Mirroring Controller

While Mirroring Controller is in a stopped state, execute the mc_ctl command in start mode to start Mirroring Controller.

Enabling automatic switch/disconnection

Execute the mc_ctl command in start mode.

Example)

```
$ mc_ctl start -M /mcdir/inst1
```

When only the instance is started and stopped, the following will happen:

- When only the instance is started

  Features such as automatic switch and automatic disconnection will not work until Mirroring Controller is started.

- When only the instance is stopped

  Mirroring Controller determines that an error has occurred in the instance, and performs an unnecessary automatic switch.

  Automatic switch may also stop working correctly in some cases.

Disabling automatic switch/disconnection

Execute the mc_ctl command in start mode with the -F option specified.

Example)

```
$ mc_ctl start -M /mcdir/inst1 -F
```

When only the instance is started and stopped, the following will happen:

- When only the instance is started

  Errors indicated in "1.1 What is Database Multiplexing Mode" will not be detected until Mirroring Controller is started.

- When only the instance is stopped

  Mirroring Controller determines that an error has occurred in the instance, and outputs an error to the system log.

Point

- To start the Mirroring Controller process only, execute the mc_ctl command in start mode with the --mc-only option specified.
- After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.
- When the arbitration server is used for automatic degradation, the Mirroring Controller process startup fails on the database server if the Mirroring Controller arbitration process has not been started on the arbitration server in advance. However, even if the Mirroring Controller arbitration process cannot be started in advance, the Mirroring Controller process can be started by specifying the --async-connect-arbiter option in the mc_ctl command.

Note

- When the arbitration server is used for automatic degradation, the database server must connect to the arbitration server, and as a result, Mirroring Controller startup may take longer.
- Mirroring Controller startup usually fails if the standby server is mistakenly started as the primary server or if the old primary server is not recovered after the switch and is then mistakenly started as the primary server. However, if the admin network is disconnected,

then startup does not fail, and both servers may become primary servers. Therefore, ensure that the admin network is connected before starting Mirroring Controller.

Stopping Mirroring Controller

While Mirroring Controller is running, execute the mc_ctl command in stop mode to stop Mirroring Controller process.

Example)

```
$ mc_ctl stop -M /mcdir/inst1
```

Point

To stop the Mirroring Controller process only, execute the mc_ctl command in stop mode with the --mc-only option specified.

Note

To prevent an unintended automatic switch, before shutting down the operating system on the primary server, you must stop the Mirroring Controller, or shut down the operating system on the standby server.

See

Refer to the Reference for information on how to specify the mc_ctl command.

3.3 Checking the Database Multiplexing Mode Status

3.3.1 Checking the Status of the Database Server

This section describes how to check the status of the database server.

Check the multiplexed database status by executing the mc_ctl command in status mode.

Additionally, errors can be detected by monitoring the Mirroring Controller messages. If the status or messages are monitored periodically, you can react quickly following an automatic switch failure.

Checking the status of the multiplexing database

When the mc_ctl command is executed, the details of the multiplexing configuration, information about whether switch is possible following the error, and location and details of the error that caused the switch or disconnection are displayed.

After starting database multiplexing mode, execute the mc_ctl command in status mode to check the multiplexing status.

An example of the status displayed when the mc_ctl command is executed is shown below.

Example)

```
$ mc_ctl status -M /mcdir/inst1

mirroring status
----------------
switchable
server_id  host_role       host            host_status  db_proc_status       disk_status
-------------------------------------------------------------------------------------------
----
server1    primary         192.0.2.100     normal       normal                 normal
server2    standby         192.0.2.110     normal       normal                 normal
```

Checking the status of connection to the Mirroring Controller arbitration process

When the arbitration server is used for automatic degradation, the status of the connection to the Mirroring Controller arbitration process can be checked by specifying the --arbiter option. If the output status is "online", it indicates that an arbitration request can be made from the database server to the arbitration server. When the arbitration server is used for automatic degradation, regularly execute the command in status mode with the --arbiter option specified and check that the output status is "online".

Example)
The mc_ctl command is executed with the --arbiter option specified, and the status is output.

```
$ mc_ctl status --arbiter -M /mcdir/inst1

arbiter_id   host            status
-------------------------------------
arbiter      192.0.3.120     online
```

Checking the status of data synchronization

Additionally, by referencing the pg_stat_replication statistics view on the primary server, the data synchronization status can be confirmed. However, when creating the monitoring program, note that the content of pg_stat_replication may be changed in the future.

The following example shows that the locations of the transaction log after it is sent and received (sent_lsn, replay_lsn) match, and that they are fully synchronized.

Example)

```
postgres=# select * from pg_stat_replication;
-[ RECORD 1 ]----+------------------------------
pid              | 10651
usesysid         | 10
usename          | fsep
application_name | standby
client_addr      | 192.0.2.210
client_hostname  |
client_port      | 55098
backend_start    | 2022-03-23 11:17:49.628793+09
backend_xmin     |
state            | streaming
sent_lsn         | 0/3000060
write_lsn        | 0/3000060
flush_lsn        | 0/3000060
replay_lsn       | 0/3000060
write_lag        |
flush_lag        |
replay_lag       |
sync_priority    | 1
sync_state       | sync
reply_time       | 2022-03-23 11:23:27.703366+09
```

See

- Refer to "mc_ctl" in Reference for information on the command.
- Refer to "Notes on Application Compatibility" in the Application Development Guide for information on retaining application compatibility.
- Refer to "The Statistics Collector" in "Server Administration" in the PostgreSQL Documentation for details on pg_stat_replication.

3.3.2 Checking the Status of the Arbitration Server

This section describes how to check the status of the arbitration server.

The status of the connection between the Mirroring Controller arbitration process and primary server/standby server can be checked by executing the mc_arb command in status mode.

The example below executes the mc_arb command, and shows the status.

Example)

```
$ mc_arb status -M /mcarb_dir/arbiter1

server_id    host            status
-------------------------------------
server1      192.0.3.100     online
server2      192.0.3.110     online
```

3.4 Manually Switching the Primary Server

The primary server cannot be switched automatically in the following case:

- If automatic switch/disconnection is disabled
- If output of messages is selected for heartbeat abnormalities during heartbeat monitoring of the operating system or server and the operating system/server crashes or becomes unresponsive

In this case, to manually switch the primary server, execute the mc_ctl command in switch mode on either the primary server or the standby server.

Example)

```
$ mc_ctl switch -M /mcdir/inst1
```

Point

If automatic switch/disconnection is enabled, it is possible to perform switch of primary server at any time.

3.5 Manually Disconnecting the Standby Server

The procedure to perform disconnection of the standby server differs depending on whether the automatic switch/disconnection is enabled or disabled.

If automatic switch/disconnection is enabled

Execute the mc_ctl command in stop mode on the standby server.

Example)

```
$ mc_ctl stop -M /mcdir/inst1
```

If automatic switch/disconnection is disabled

1. Execute the mc_ctl command in stop mode on the standby server.

   Example)

   ```
   $ mc_ctl stop -M /mcdir/inst1
   ```

2. Comment out the synchronous_standby_names parameter in the postgresql.conf file on the primary server. If you have set the synchronized_standby_slots parameter, comment out the synchronized_standby_slots parameter as well.

3. Execute the pg_ctl command in reload mode on the primary server.

   Example)

```
$ pg_ctl reload -D /database/inst1
```

Point

If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

3.6 Action Required when a Heartbeat Abnormality is Detected

The message below is output when a heartbeat abnormality is detected during heartbeat monitoring of operating systems or servers:

```
detected an error on the monitored object "server(server identifier name)": no response:ping timeout
(MCA00019)
```

If the heartbeat_error_action parameter in *serverIdentifier*.conf is set to "message", even if automatic switch/disconnection is enabled and Mirroring Controller is started, automatic switch/disconnection is not performed when a heartbeat abnormality is detected. Therefore, user action will be necessary.

This section explains the action required when the heartbeat_error_action parameter is set to "message" and a heartbeat abnormality is detected.

1. Identify the cause of the heartbeat abnormality. The possible causes are below:
   - The remote operating system or server crashed or is unresponsive
   - An admin network issue occurred
2. Address the cause identified in step 1.
   - The remote operating system or server crashed or is unresponsive

     Manually perform switch or disconnection using the mc_ctl command.
   - An admin network issue occurred

     Refer to "Chapter 4 Action Required when an Error Occurs in Database Multiplexing Mode", and recover the database multiplexing system.

3.7 Monitoring Mirroring Controller Messages

The messages that are output by Mirroring Controller are output to both the database server and the arbitration server. If the automatic switch fails, for example, an important message related to the continuation of the operation may be output, so ensure that the system log messages are monitored.

If the arbitration server is used for automatic degradation, monitor messages on both the database server and the arbitration server.

Message output destination on the database server

Messages are output to the system log.

Message output destination on the arbitration server

Messages are output to the system log.

Point

- To monitor message types considered to be important, an operating system setting must be configured beforehand. Refer to the operating system manuals, check if the message is of a message type that is monitored to be output to the system log, and configure the setting if required.
- If the heartbeat_error_action parameter in *serverIdentifier*.conf is set to "message", only message output is performed when a heartbeat abnormality is detected during heartbeat monitoring of operating systems and servers - automatic switch/disconnection is not

performed. Therefore users need to monitor the messages. Refer to "3.6 Action Required when a Heartbeat Abnormality is Detected" for details.

Display format on the database server

```
programName[processId]: messageType:messageText (messageNumber)
```

Specify the program name in the syslog_ident parameter of the serverIdentifier.*conf* file of the database server.

The message types output by Mirroring Controller, their severity, and their corresponding value in the system log are shown in the table below.

Table 3.1 Message type, severity, and corresponding value in the system log

Message type	Severity	Meaning	System log
INFO	Information	Provides information that does not fall under LOG or NOTICE.	INFO
LOG		Provides information recognized as a particularly important event in tracing the operation history. (Example: Automatic switch is complete)	
NOTICE	Notice	Outputs information that takes into account the user instructions within the program in response to an executed or automatically executed process.	NOTICE
WARNING	Warning	Provides a warning, for example it will soon be impossible to maintain the multiplexing state.	WARNING
ERROR	Error	Reports that an error other than FATAL or PANIC has occurred.	ERROR
FATAL		Reports that an abnormality was detected in multiplexed database systems requiring recovery of the system, and also the content and cause of the abnormality.	CRIT
PANIC		Reports that an abnormality was detected in all multiplexed database systems requiring immediate recovery of the system, and also the content and cause of the abnormality.	ALERT

The message severity has the following meanings:

- Information

  Informational status. A message that was reported by the system is displayed. No action is required.

- Notice

  Informational status, but a message that should be noted is displayed. If necessary, take the actions described in the "Action" section of the message.

- Warning

  No error has occurred, but the user is requested to check, and take action. Take the actions described in the "Action" section of the message.

- Error

  An error has occurred. Take the actions described in the "Action" section of the message.

Display format on the arbitration server

```
programName[processId]: messageType: messageText (messageNumber)
```

Specify the program name in the syslog_ident parameter of the arbitration.conf file of the arbitration server.

The message types output by Mirroring Controller, their severity, and their corresponding value in the output destination log are shown in the table below.

Table 3.2 Message type, severity, and corresponding value in the output destination log

Message type	Severity	Meaning	System log
INFO	Information	Provides information not categorized as LOG or NOTICE.	INFO
LOG		Provides information recognized as a particularly important event in tracing the operation history. (Example: Automatic switch is complete)	
NOTICE	Notice	Outputs information that takes into account the user instructions within the program in response to an executed or automatically executed process.	NOTICE
WARNING	Warning	Provides a warning, for example it will soon be impossible to perform the arbitration process.	WARNING
ERROR	Error	Reports that an error other than FATAL or PANIC has occurred.	ERROR
FATAL		Reports that an abnormality was detected in the arbitration server requiring recovery of the system, and also the content and cause of the abnormality.	CRIT
PANIC		Reports that an abnormality was detected in the arbitration server requiring immediate recovery of the system, and also the content and cause of the abnormality.	ALERT

The message severity has the following meanings:

- Information

  Informational status. A message that was reported by the system is displayed. No action is required.

- Notice

  Informational status, but a message that should be noted is displayed. If necessary, take the actions described in the "Action" section of the message.

- Warning

  No error has occurred, but the user is requested to check, and take action. Take the actions described in the "Action" section of the message.

- Error

  An error has occurred. Take the actions described in the "Action" section of the message.

3.8 Server Maintenance

To perform maintenance tasks such as periodic server inspections and the application of updates for software products including the operating system, you must perform a planned stop of the server, and then perform the maintenance.

3.8.1 Rolling Updates

In database multiplexing mode, rolling updates, that perform the maintenance for the servers that comprise the cluster system, can be performed while jobs continue.

First, perform the maintenance for the standby server, and then switch the standby server to the primary server. Then, perform the maintenance for the original primary server that was switched to the standby server. This enables maintenance to be performed while jobs continue.

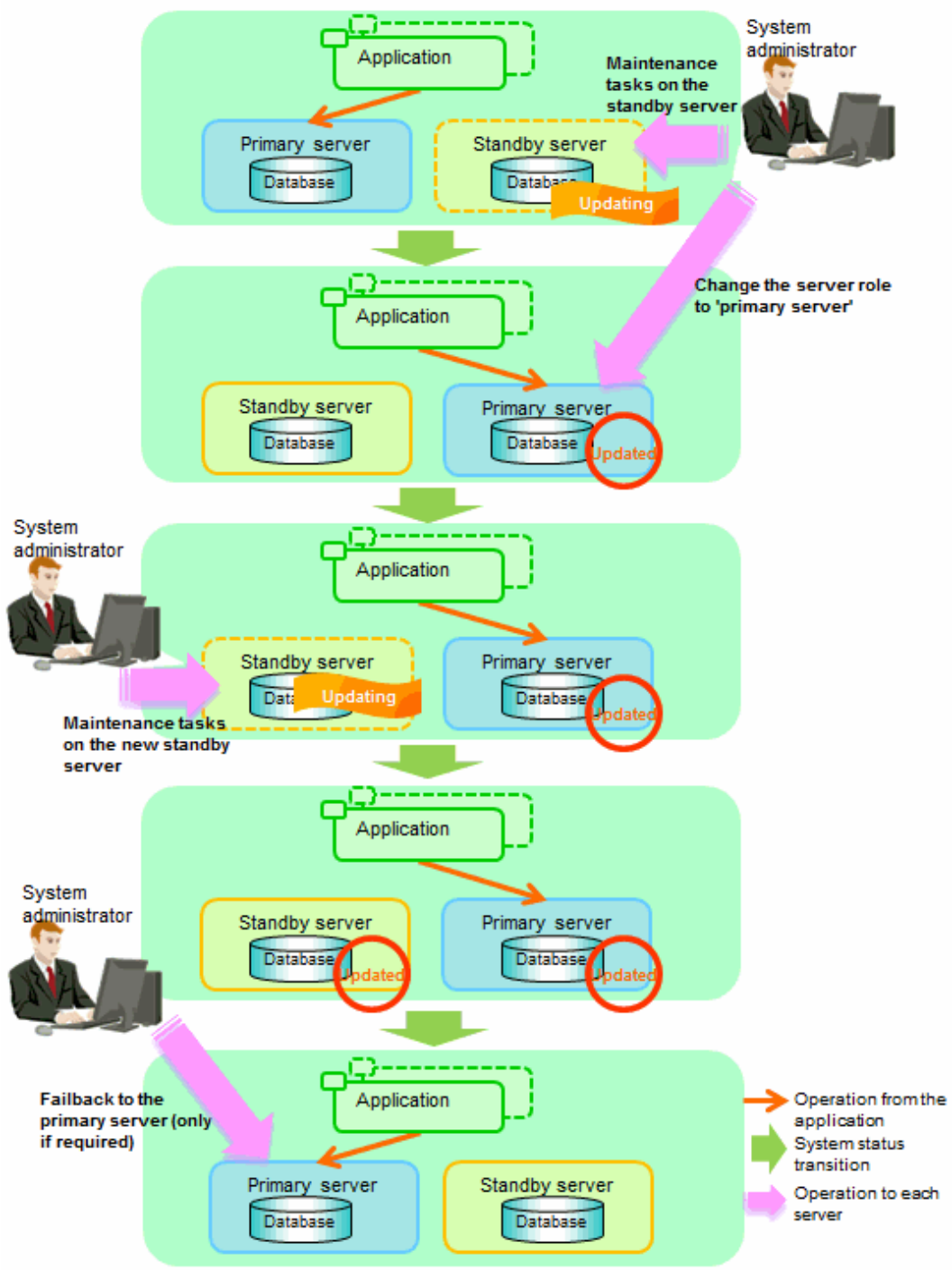
Note that arbitration server maintenance can be performed without affecting database server operation, so it is not necessary to consider rolling update.

See

If the downtime due to the maintenance of the standby server is expected to be long, refer to "Standby server downtime" in "3.9.1 Changes Required when the Standby Server is Stopped".

The flow of a rolling update is shown below.

Figure 3.1 Performing a Rolling Update



Perform the following procedure as shown in the above figure:

Standby server maintenance tasks

1. To perform the maintenance on the standby server, stop Mirroring Controller.

   Example)

   ```
   $ mc_ctl stop -M /mcdir/inst1
   ```

2. Ensure that Mirroring Controller has completely stopped.

   If the multiplexed instances and Mirroring Controller have been configured on the standby server to start and stop automatically when the operating system of the database server is started or stopped, cancel the setting to start and stop automatically.

    See

   Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for information on how to configure the multiplexed instances and Mirroring Controller to start and stop automatically when the operating system of the database server start and stops.

   As the OS superuser, execute the systemctl command to disable automatic start and stop.

   The example below disables automatic start and stop of "mcoi_inst1.service".

   Example)

   ```
   # systemctl disable mcoi_inst1.service
   ```

3. Perform maintenance tasks.

4. Create a copy of the primary server instance on the standby server.

   Execute the pg_basebackup command to create data in the standby server by synchronizing with the primary server.

   Example)

   ```
   $ pg_basebackup -D /database/inst1 -X fetch --waldir=/transaction/inst1 --progress --verbose -R
   --dbname='application_name=standbyServerName' -h primaryServerHostName -p
   primaryServerPortNumber
   ```

    See

   The procedure for copying the primary server instance to the standby server is the same as the procedure for setting up the standby server.

   Refer to "2.5.2 Creating, Setting, and Registering the Standby Server Instance", and then perform the recovery.

5. Check the settings for automatic start and stop of the multiplexed instances and Mirroring Controller.

   If the multiplexed instances and Mirroring Controller were configured in step 2 to not start and stop automatically when the operating system of the database server starts and stops, then change the settings back. This step can be skipped if automatic start and stop are not required.

   As the OS superuser, execute the systemctl command to enable automatic start and stop.

   The example below disables automatic start and stop of "mcoi_inst1.service".

   Example)

   ```
   # systemctl enable mcoi_inst1.service
   ```

6. Start (rebuild) Mirroring Controller on the standby server.

   This operation is required when determining the maintenance tasks on the standby server.

Enabling automatic switch/disconnection

As the instance administrator user, execute the mc_ctl command in start mode.

Example)

```
$ mc_ctl start -M /mcdir/inst1
```

Disabling automatic switch/disconnection

As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

Example)

```
$ mc_ctl start -M /mcdir/inst1 -F
```

Point

After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

Switching to the primary server

To perform the maintenance on the primary server, execute the mc_ctl command in the switch mode on the primary server or the standby server.

Example)

```
$ mc_ctl switch -M /mcdir/inst1
```

When the switch is complete, the synchronous_standby_names parameter and synchronized_standby_slots parameter in the postgresql.conf file of the new primary server will be commented as follows:

Example)

```
#synchronous_standby_names = 'primary'
#synchronized_standby_slots = 'slot'
```

New standby server maintenance tasks

1. Stop the Mirroring Controller.

   On the new standby server (the primary server before the switch), execute the mc_ctl command in stop mode.

   If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

   Example)

   ```
   $ mc_ctl stop -M /mcdir/inst1
   ```

2. Ensure that Mirroring Controller has completely stopped.

   If the multiplexed instances and Mirroring Controller have been configured on the new standby server to start and stop automatically when the operating system of the database server is started or stopped, cancel the setting to start and stop automatically now.

See

Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for information on how to configure the multiplexed instances and Mirroring Controller to start and stop automatically when the operating system of the database server starts and stops.

As the OS superuser, execute the systemctl command to disable automatic start and stop.

The example below disables automatic start and stop of "mcoi_inst1.service".

Example)

```
# systemctl disable mcoi_inst1.service
```

3. Perform the maintenance on the new standby server that was stopped.
4. Create a copy of the new primary server instance on the new standby server.

   Execute the pg_basebackup command to create data in the new standby server by synchronizing with the new primary server.

   Example)

```
$ pg_basebackup -D /database/inst1 -X fetch --waldir=/transaction/inst1 --progress --verbose -R
--dbname='application_name=standbyServerName' -h primaryServerHostName -p
primaryServerPortNumber
```

See

The procedure for copying the primary server instance to the standby server is the same as the procedure for setting up the standby server.

Refer to "2.5.2 Creating, Setting, and Registering the Standby Server Instance", and then perform the recovery.

5. Check the settings for automatic start and stop of the multiplexed instances and Mirroring Controller.

   If the multiplexed instances and Mirroring Controller were configured in step 2 to not start and stop automatically when the operating system of the database server starts and stops, then change the settings back. This step can be skipped if automatic start and stop are not required.

   As the OS superuser, execute the systemctl command to enable automatic start and stop.

   The example below disables automatic start and stop of "mcoi_inst1.service".

   Example)

```
# systemctl enable mcoi_inst1.service
```

6. After the maintenance is complete, edit the following parameters in the postgresql.conf file of the standby server as required.

   Copying an instance results in the value of the synchronous_standby_names parameter becoming the specified value on the primary server. Therefore, correct it to the specified value on the standby server. If the parameter was commented out, then you must uncomment it.

7. On the standby server, start (rebuild) Mirroring Controller.

   Enabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode.

   Example)

```
$ mc_ctl start -M /mcdir/inst1
```

   Disabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

   Example)

```
$ mc_ctl start -M /mcdir/inst1 -F
```

Point

After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

Failback of the Primary Server

Revert the primary server and standby server to the original server configuration. Do this to execute the main job on the previous primary server. Refer to "4.1.1.3 Failback of the Primary Server" for details.

Note

Obtain a backup as soon as this task is complete.

3.8.2 Stopping for Maintenance

Perform this procedure to stop all servers for periodic inspections, for example. On the server on which Mirroring Controller is running, execute the mc_ctl command in stop mode to stop the instance and Mirroring Controller.

If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

After that, on the server where the Mirroring Controller arbitration process is running, execute the mc_arb command in stop mode to stop the Mirroring Controller arbitration process.

Stopping Mirroring Controller

```
Example)
```

```
$ mc_ctl stop -M /mcdir/inst1 -a
```

Stopping the Mirroring Controller arbitration process

```
Example)
```

```
$ mc_arb stop -M /mcarb_dir/arbiter1
```

3.8.3 Arbitration Server Maintenance

Arbitration server maintenance can be performed without affecting database server operation.

Follow the procedure below to perform arbitration server maintenance.

1. Execute the mc_arb command in stop mode to forcibly stop the Mirroring Controller arbitration process.

   ```
   Example)
   ```

   ```
   $ mc_arb stop -M /mcarb_dir/arbiter1 -e
   ```

2. Perform maintenance tasks.
3. Execute the mc_arb command in start mode to restart the Mirroring Controller arbitration process.

   ```
   Example)
   ```

   ```
   $ mc_arb start -M /mcarb_dir/arbiter1
   ```

4. Execute the mc_arb command in status mode to check that the arbitration server is connected to the database server.

   The example below executes the mc_arb command, and shows the status.

   Example)

```
$ mc_arb status -M /mcarb_dir/arbiter1

server_id    host            status
-------------------------------------
server1      192.0.3.100     online
server2      192.0.3.110     online
```

5. Check the command output.

   Items to be checked

   Check that the output status is "online" on both lines.

3.9 Changes in Operation

The following changes in operation may be required:

- Changes required when the standby server is stopped
- Changing from single server mode to database multiplexing mode
- Changing from database multiplexing mode to single server mode
- Changing to database multiplexing mode when the arbitration server is used for automatic degradation
- Changing parameters
- Uninstalling in the database multiplexing mode

3.9.1 Changes Required when the Standby Server is Stopped

Operation when the standby server is stopped

Before performing maintenance for the primary server instance when the standby server has been stopped, stop Mirroring Controller on the primary server, comment out the synchronous_standby_names parameter and synchronized_standby_slots parameter in the postgresql.conf file of the primary server, and then execute the pg_ctl command in reload mode.

If this operation is not performed, operations performed on the primary server for the instance will remain in a wait state.

 See

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

Standby server downtime

If you specified the synchronous_standby_names parameter of the postgresql.conf file and then the standby server instance is stopped, consider the points below.

- The wal_sender_timeout parameter in the postgresql.conf file

  If the standby server is stopped after the timeout set in this parameter was exceeded, an error stating that the transaction log could not be received may be output to the primary server system log, and all transaction logs that should be transferred to the standby server are accumulated.

- The wal_keep_size parameter in the postgresql.conf file

  If a transaction log that exceeds the value set in this parameter was generated while the standby server was stopped, the transaction log may be deleted.

Additionally, setting this parameter requires consideration regarding stabilization of the database multiplexing mode. Refer to "2.11.1 Tuning to Stabilize the Database Multiplexing Mode" for details.

Note

The standby server must be rebuilt if the pending transaction log to be transferred to the standby server is lost when the standby server is started after the maintenance task is complete.

Take the action advised in the recovery operation that starts from "4.1.1.1.3 Identify cause of error and perform recovery" through to "4.1.1.2 Rebuild the Standby Server".

3.9.2 Changing from Single Server Mode to Database Multiplexing Mode

The procedure for switching single server mode to database multiplexing mode for the purposes of high reliability and load distribution of the system is explained below.

This procedure is equivalent to the setup procedure explained in "Chapter 2 Setting Up Database Multiplexing Mode".

Note

If the data storage destination directory name is not comprised of ASCII characters

Stop the application job and then migrate to a directory with a name that uses only ASCII characters:

1. Stop the database instance on the primary server.
2. Change the name of the data storage destination directory to one that uses only ASCII characters.

See

When encrypting the storage data, refer to "Database Multiplexing Mode" in the Operation Guide, and then perform the setup for encryption on the primary and standby servers.

1. Install on the arbitration server

   Perform this step only if the arbitration server is used for automatic degradation.

   Install the Server Assistant on the server where the Mirroring Controller arbitration process is started.

   Refer to "Installation" in the Installation and Setup Guide for Server Assistant for information on how to install the Server Assistant.

2. Install on the standby server

   Install Fujitsu Enterprise Postgres on the server to be started as the standby server.

   Refer to "Installation" in the Installation and Setup Guide for Server for information on how to install Fujitsu Enterprise Postgres.

   Use ASCII characters in the data storage destination directory.

3. Stop the application jobs

   Stop the application jobs to be connected to the primary server.

4. Change the primary server settings

   To allow connections from the server to be started as the standby server, configure the settings in step 2 and thereafter of "2.4.2 Creating, Setting, and Registering the Primary Server Instance" on the primary server.

5. Set up the arbitration server

   Refer to "2.3 Setting Up the Arbitration Server" for details.

   Perform this step only if the arbitration server is used for automatic degradation.

6. Set up database multiplexing mode on the primary server

   Refer to "2.4.1 Setting Up Database Multiplexing Mode on the Primary Server" for details.

7. Set up database multiplexing mode on the standby server

   Refer to "2.5.1 Setting Up Database Multiplexing Mode on the Standby Server" for details.

8. Create the standby server instance and start it

   Refer to "2.5.2 Creating, Setting, and Registering the Standby Server Instance" for details.

After the above steps are completed, refer to the remaining explanations in "Chapter 2 Setting Up Database Multiplexing Mode" and ensure that the required settings and operations are completed.

3.9.3 Changing from Database Multiplexing Mode to Single Server Mode

The procedure for stopping database multiplexing mode and changing to single server mode is explained below.

Some tasks must be performed on the database server, and others must be performed on the arbitration server.

The tasks on the arbitration server are required only if the arbitration server is used for automatic degradation.

Tasks on the database server

1. Determine the server for which the instance is to be stopped, and switch this server

   Determine the server that is to be excluded as the database multiplexing mode target, and for which the instance is to be stopped.

   If the server for which the instance is to be stopped is the primary server, execute the mc_ctl command in the switch mode to switch the standby server to the primary server.

   The standby server after the switch is complete will be the server for which the instance is to be stopped.

   If the server for which the instance is to be stopped is the standby server, there is no need to perform the switch operation.

   Example)

```
$ mc_ctl switch -M /mcdir/inst1
```

2. Stop Mirroring Controller and the instance, and delete the file resources

   On the server that was determined in step 1, execute the mc_ctl command in stop mode to stop Mirroring Controller and the instance.

   If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

   Example)

```
$ mc_ctl stop -M /mcdir/inst1
```

   Then, delete the following file resources:

   - Data storage destination directory
   - Mirroring Controller management directory

   Example)

```
$ rm -rf /database/inst1
$ rm -rf /mcdir/inst1
```

See

Refer to "Security-Related Notes" in the Operation Guide for details on deleting the data securely.

3. Stop the application jobs

   Stop the application jobs to be connected to the primary server.

4. Stop Mirroring Controller and the instance on the primary server

   Execute the mc_ctl command in stop mode on the primary server.

   If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

   Example)

```
$ mc_ctl stop -M /mcdir/inst1
```

5. Delete the database multiplexing mode settings that were configured for the primary server instance.

   Reset the postgresql.conf file parameters to their values before the database multiplexing operation was set.

   Delete the file resources from the Mirroring Controller management directory.

   If the backup operation was performed, delete the following resources:

   - Mirroring Controller management directory backup data obtained in database multiplexing mode
   - Instance backup data obtained in database multiplexing mode

   Additionally, if the primary_conninfo parameter is set in the postgresql.auto.conf file, execute the ALTER SYSTEM RESET statement to delete the setting.

   Example)
   An example execution of the psql command is shown below.

```
postgres=# ALTER SYSTEM RESET primary_conninfo;
```

   After these actions are performed, ensure that the backup data is collected when starting the single operation.

See

- Refer to "Security-Related Notes" in the Operation Guide for details on deleting the data securely.
- Refer to "2.14 Backup Operation" for details on the backup operation.
- Refer to "Appendix A Parameters" for details on the postgresql.conf file parameters.

Point

In the above procedure, if the postgresql.conf file of the single primary server can be changed by reloading the file, the operation mode can be changed without stopping the application job.

In that case, execute the mc_ctl command in stop mode with the --mc-only option specified to stop only Mirroring Controller in relation to stopping the primary server.

Tasks on the arbitration server

1. Execute the mc_arb command in stop mode to stop the Mirroring Controller arbitration process.

   Example)

```
$ mc_arb stop -M /mcarb_dir/arbiter1
```

2. Delete the Mirroring Controller arbitration process management directory.

   Example)

```
$ rm -rf /mcarb_dir/arbiter1
```

3.9.4 Changing to Database Multiplexing Mode when the Arbitration Server is Used for Automatic Degradation

This section provides the procedure to change to database multiplexing mode using the Mirroring Controller only on the database server when the arbitration server is used for automatic degradation.

Some tasks must be performed on the database server, and others must be performed on the arbitration server.

Tasks on the arbitration server

1. Set up the arbitration server.

   Refer to "2.3 Setting Up the Arbitration Server" for information on how to set up the arbitration server.

Tasks on the database server

1. On the server where Mirroring Controller is running, execute the mc_ctl command in stop mode to stop Mirroring Controller on the primary server and standby server.

   Example)

   ```
   $ mc_ctl stop -M /mcdir/inst1 -a --mc-only
   ```

2. Edit the network.conf file of the primary server and standby server to add the information of the arbitration server.

   Refer to "A.3 Network Configuration File" for details.

   The definition example of the network.conf file of the primary server is shown below:

   Example)

   The IDs of the primary server and standby server are set to "server1" and "server2", and their port numbers are set to "27540" and "27541". The ID of the server of the Mirroring Controller arbitration process is set to "arbiter", and its port number is set to "27541".

   ```
   server1 192.0.2.100,192.0.3.100 27540,27541 server
   server2 192.0.2.110,192.0.3.110 27540,27541 server
   arbiter 192.0.3.120 27541 arbiter
   ```

- Ensure that the port numbers set for the primary server, standby server, and arbitration server do not conflict with other software. Also do not configure the same segment for the admin network and arbitration network.
- If the server type is "server", two IP addresses or host names, and two port numbers need to be specified in the following order:
  - IP address or host name of the database server used as the admin network
  - IP address or host name of the database server used as the arbitration network
  - Port number of the database server used as the admin network
  - Port number of the database server used as the arbitration network
- If the server type is "arbiter", specify the IP address or host name set for the my_address parameter and the port number set for the port parameter in arbitration.conf.

3. Edit the *serverIdentifier*.conf file of the primary server and standby server to add parameters required for the operation where the arbitration server is used for automatic degradation.

   Refer to "A.4.1 Server Configuration File for the Database Servers" for information on the parameters required when the arbitration server is used for automatic degradation.

4. On the primary server and standby server, execute the mc_ctl command in start mode to start the Mirroring Controller process.

   Example)

```
$ mc_ctl start -M /mcdir/inst1 --mc-only
```

Common tasks

1. Check the connection status from the database server or arbitration server.

   Refer to "2.8 Checking the Connection Status" for details.

3.9.5 Changing Parameters

Stop Mirroring Controller before editing the Mirroring Controller server configuration file and network configuration file.

If the Mirroring Controller process crashes or becomes unresponsive, restart is performed automatically by the Mirroring Controller monitoring process, and the configuration file is reloaded. Therefore, if the configuration file was being edited, unintended behavior will occur.

3.9.6 Uninstalling in Database Multiplexing Mode

This section explains how to uninstall Fujitsu Enterprise Postgres on a server using database multiplexing mode.

Some tasks must be performed on the database server, and others must be performed on the arbitration server.

The tasks on the arbitration server are required only if the arbitration server is used for automatic degradation.

Tasks on the database server

1. Stop the multiplexed instances and Mirroring Controller

   Refer to "3.2 Starting and Stopping Mirroring Controller" for information on how to stop the instance.

2. Uninstall Fujitsu Enterprise Postgres

   Refer to "Uninstallation" in the Installation and Setup Guide for Server for information on how to uninstall Fujitsu Enterprise Postgres.

Tasks on the arbitration server

Refer to "Uninstallation" in the Installation and Setup Guide for Server Assistant, and uninstall the Server Assistant.

Chapter 4 Action Required when an Error Occurs in Database Multiplexing Mode

This chapter describes the action required if an error occurs in database multiplexing mode.

In database multiplexing mode, when an error is detected, the switch or disconnection of the standby server is performed automatically, so that only the primary server starts degrading. In this case, the recovery tasks will be required for the standby server on which the switch or disconnection was performed.

Other possible cases are as follows:

- When automatic switch fails
- When automatic disconnection fails
- When all servers or instances were stopped

4.1 Action Required when Server Degradation Occurs

If the server has started degrading, the recovery tasks will vary depending on whether the cause was the switch (failover or switchover), or the disconnection.

Execute the mc_ctl command in status mode, or refer to the system log, and check if the cause of the server degradation was the switch or the disconnection.

In the example below, the mc_ctl command is executed in status mode.

If a switch has occurred, "switched" (the switch is complete and the server is in a degrading state) is displayed for "mirroring status".

Example)

```
$ mc_ctl status -M /mcdir/inst1
mirroring status
----------------
switched

   :
```

If a disconnection has occurred, "not-switchable" (disconnection was performed so the server cannot be switched) is displayed for "mirroring status".

Example)

```
$ mc_ctl status -M /mcdir/inst1
mirroring status
----------------
not-switchable

  :
```

Note

If Mirroring Controller detects any errors on the server on which operations are continuing during recovery to database multiplexing mode from a degrading operation state, perform the procedure in "4.1.3 Addressing Errors During Degrading Operation", and then recover to database multiplexing mode.

4.1.1 Operations when the Server has Started Degrading after a Switch has Occurred

This section explains the operations when the server has started degrading after a switch has occurred.

Note

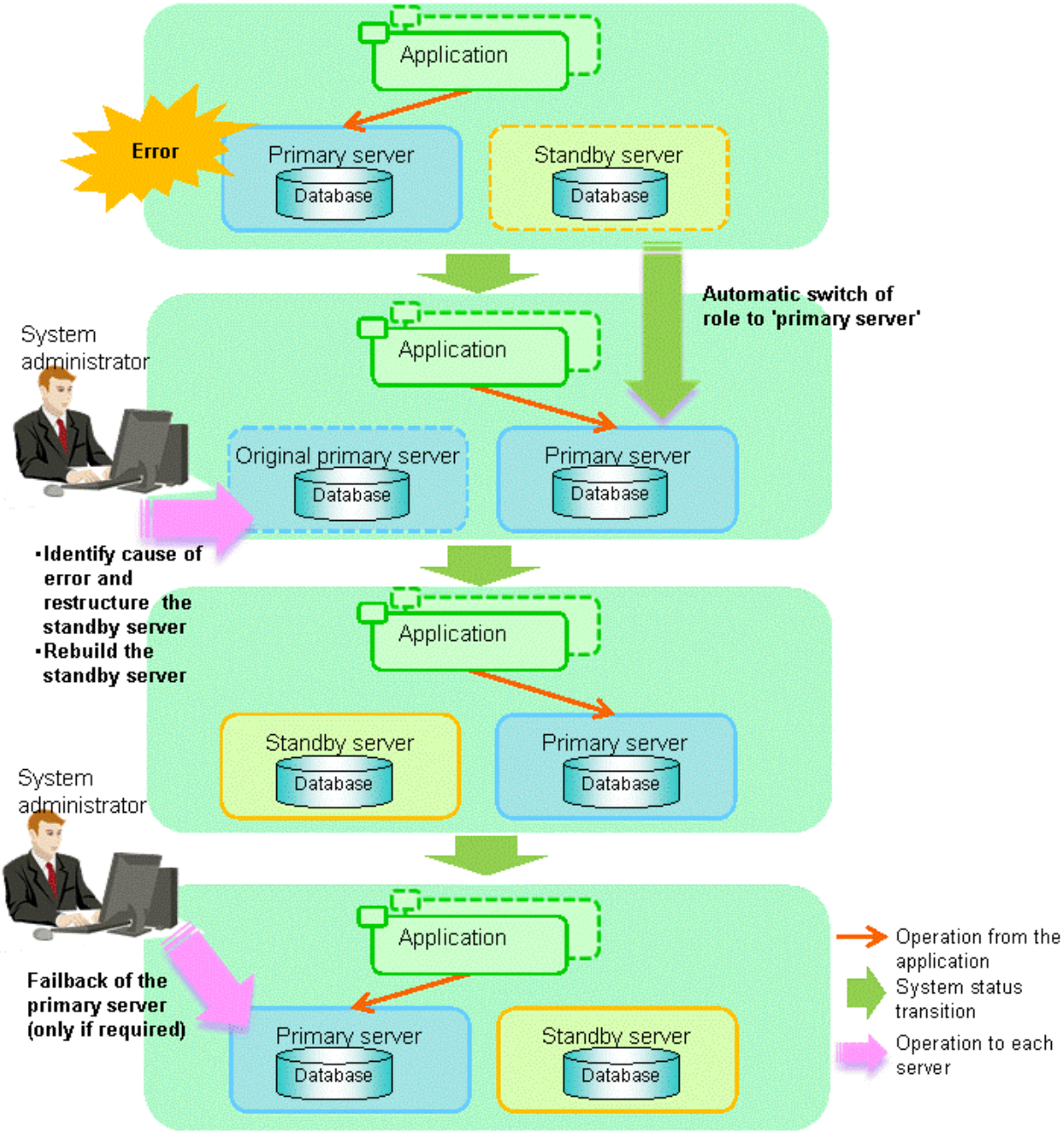
- After a switch has occurred as a result of an abnormality on the primary server, the database will not have a multiplexed configuration until the standby server is rebuilt. Remove the cause of the error as quickly as possible, and then rebuild the standby server.
- If the reference job was executed on the standby server, and the servers are switched because an error occurred on the primary server, the load is concentrated on the new primary server. Accordingly, pause the reference job on the original standby server, rebuild the original primary server as the new standby server, and then resume the reference job for the new standby server.
- If the instance on the new primary server is stopped before the original primary server where the error occurred is rebuilt as the new standby server, a split brain occurs at startup from the instance on the original primary server. Therefore, start the instance on the new primary server before rebuilding the standby server.

If the switch occurred and the server has started degrading, perform the following operations to recover the standby server and revert it to its original state:

- Identify Cause of Error and Restore the Standby Server
- Rebuild the Standby Server
- Failback of the Primary Server (only if required)

The flow of these operations is shown in the figure below.

Figure 4.1 Flow of operations



4.1.1.1 Identify Cause of Error and Restore the Standby Server

Perform the recovery according to the following procedure:

1. Stop Mirroring Controller
2. Recovery of the Mirroring Controller management directory
3. Identify cause of error and perform recovery

4.1.1.1.1 Stop Mirroring Controller

Execute the mc_ctl command in stop mode for the original primary server on which the error occurred.

If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

Example)

```
$ mc_ctl stop -M /mcdir/inst1
```

This also stops the instance that is required to perform the recovery.

Note

If the instance does not stop, refer to "Actions in Response to Failure to Stop an Instance" in the Operation Guide, and then stop the instance.

Then, specify the -e option in the above command to forcibly stop Mirroring Controller.

4.1.1.1.2 Recovery of the Mirroring Controller management directory

Copy the files in the Mirroring Controller management directory from the backup data, and then perform the recovery.

4.1.1.1.3 Identify cause of error and perform recovery

Refer to the system log of the primary server and the standby server to identify the cause of the error, and then perform recovery.

The following commands can be used to recover a standby server. Select depending on the recovery and the situation.

- pg_basebackup

  Creates a copy of all resources of the primary server instance.

- pg_rewind

  Creates a copy of only the updated files on the new primary server. For this reason, if this command is used to incorporate a new standby server, recovery time can be shortened. To use this command to build the original primary server as a new standby server, at least one of the following must be met:

  a. Checksums were enabled when an instance was created, or

  b. The wal_log_hints parameter of postgresql.conf was enabled when an instance was started.

  Additionally, full_page_writes must be enabled, which is its default value.

See

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

The example below executes the pg_rewind command to perform recovery by synchronizing data on the original primary server with the new primary server.

1. Wait for the application of unapplied update transaction logs on the new primary server.

   Execute the SQL below on the new primary server, and wait until the result is false.

   ```
   # select pg_is_in_recovery();
   ```

   Example)

   ```
   $ psql -h hostNameOfNewPrimaryServer -p portNumOfNewPrimaryServer -d dbName -c "select
   pg_is_in_recovery();"
   ```

   Any database can be connected to.

Note

If the pg_rewind command is executed immediately after promotion of the new primary server, the processing in steps 1 and 2 is required. If update-type SQL can be executed on the new primary server and checkpoint processing is executed after promotion, the processing in steps 1 and 2 will not be necessary.

2. Update the timeline ID.

   Execute checkpoint processing, and update the timeline ID.

   ```
   $ psql -h hostNameOfNewPrimaryServer -p portNumOfNewPrimaryServer -d dbName -c "checkpoint;"
   ```

   Any database can be connected to.

3. Create a copy of the new primary server instance in the original primary server (new standby server).

   Execute the pg_rewind command to synchronize the new standby server data with the new primary server.

   Example)

   ```
   $ pg_rewind -D /database/inst1 -R --source-server='user=userName host=newPrimaryServerHostName
   port=newPrimaryServerPortNumber dbname=dbName application_name=newStandbyServerName'
   ```

   ### Note

   - Use the pg_rewind command with the -R option to create a standby.signal file. If you do not create the standby.signal file, the Mirroring Controller cannot be started as a standby server.
   - If using a method that requires password authentication for connections to the primary server, you will need to ensure that authentication is performed automatically. If the -R option is specified for the pg_rewind command and the password parameter is specified for the --dbname option, the pg_rewind command will set the password in the primary_conninfo parameter in postgresql.auto.conf file, enabling connections to be performed automatically.

     If a password is not set in the primary_conninfo parameter in postgresql.auto.conf file, it will be necessary to create a .pgpass file in the home directory of the instance administrator user, and specify a password for the replication database.
   - If you need to set a connection string other than host, port and application_name, include it in the setting of the primary_conninfo parameter.
   - The primary_conninfo parameter should not be set in the postgresql.conf file, but only in the postgresql.auto.conf file using the pg_rewind command.

4. Specify parameters in the postgresql.conf file of the original primary server (new standby server).

   Set the parameters required for the standby server in postgresql.conf.

   Refer to "Table 2.4 Parameters" for information on the parameters to set in postgresql.conf.

See

- Refer to "Hot Standby" in the PostgreSQL Documentation for details on the standby.signal file.
- Refer to "Setting Up a Standby Server" in the PostgreSQL Documentation for details on the primary_conninfo.

Note

A new timeline is branched for the new primary server due to promotion, so 'latest' needs to be specified for the recovery_target_timeline parameter so that the old primary server (new standby server) follows the new primary server.

4.1.1.2 Rebuild the Standby Server

The starting of the recovered original primary server as the standby server is referred to as the "standby server rebuild".

On the original primary server, start Mirroring Controller and the instance.

Enabling automatic switch/disconnection

As the instance administrator user, execute the mc_ctl command in start mode.

Example)

```
$ mc_ctl start -M /mcdir/inst1
```

Disabling automatic switch/disconnection

As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

Example)

```
$ mc_ctl start -M /mcdir/inst1 -F
```

Point

After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

4.1.1.3 Failback of the Primary Server

To revert the primary server and standby server to the original server configuration after rebuilding the standby server, perform failback for the primary server.

Do this to execute the main job on the previous primary server.

Perform the following procedure:

1. Failback of the primary server

   Execute the mc_ctl command in switch mode on the primary server or the standby server.

   Example)

   ```
   $ mc_ctl switch -M /mcdir/inst1
   ```

   After executing the mc_ctl command in switch mode, the status will be as follows:

   Example)

   ```
   $ mc_ctl status -M /mcdir/inst1
   mirroring status
   ----------------
   switched
   server_id  host_role                  host         host_status  db_proc_status       disk_status
   ------------------------------------------------------------------------------------------------
   server1    primary                  192.0.2.100 normal        abnormal(postmaster) normal
   server2    none(inactivated primary) 192.0.2.110 normal        abnormal(postmaster) normal
   ```

2. Stop the original primary server

   On the original primary server, execute the mc_ctl command in stop mode to stop Mirroring Controller and the instance.

   If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

   Example)

```
$ mc_ctl stop -M /mcdir/inst1
```

3. Create a copy of the new primary server instance in the original primary server (new standby server)

   Execute the pg_basebackup command to create data in the new standby server by synchronizing with the new primary server.

   Example)

```
$ pg_basebackup -D /database/inst1 -X fetch --waldir=/transaction/inst1 --progress --verbose -R
--dbname='application_name=standbyServerName' -h primaryServerHostName -p
primaryServerPortNumber
```

See

The procedure for copying the new primary server instance to the new standby server is the same as the procedure for setting up the new standby server.

Refer to "2.5.2 Creating, Setting, and Registering the Standby Server Instance", and then perform the recovery.

4. Rebuild the standby server

   On the standby server, start Mirroring Controller and the instance.

   Enabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode.

   Example)

```
$ mc_ctl start -M /mcdir/inst1
```

   Disabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

   Example)

```
$ mc_ctl start -M /mcdir/inst1 -F
```

Point

After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

4.1.2 Operations when the Server has Started Degrading after a Disconnection has Occurred

This section explains the operations when the server has started degrading after a disconnection has occurred.

Note

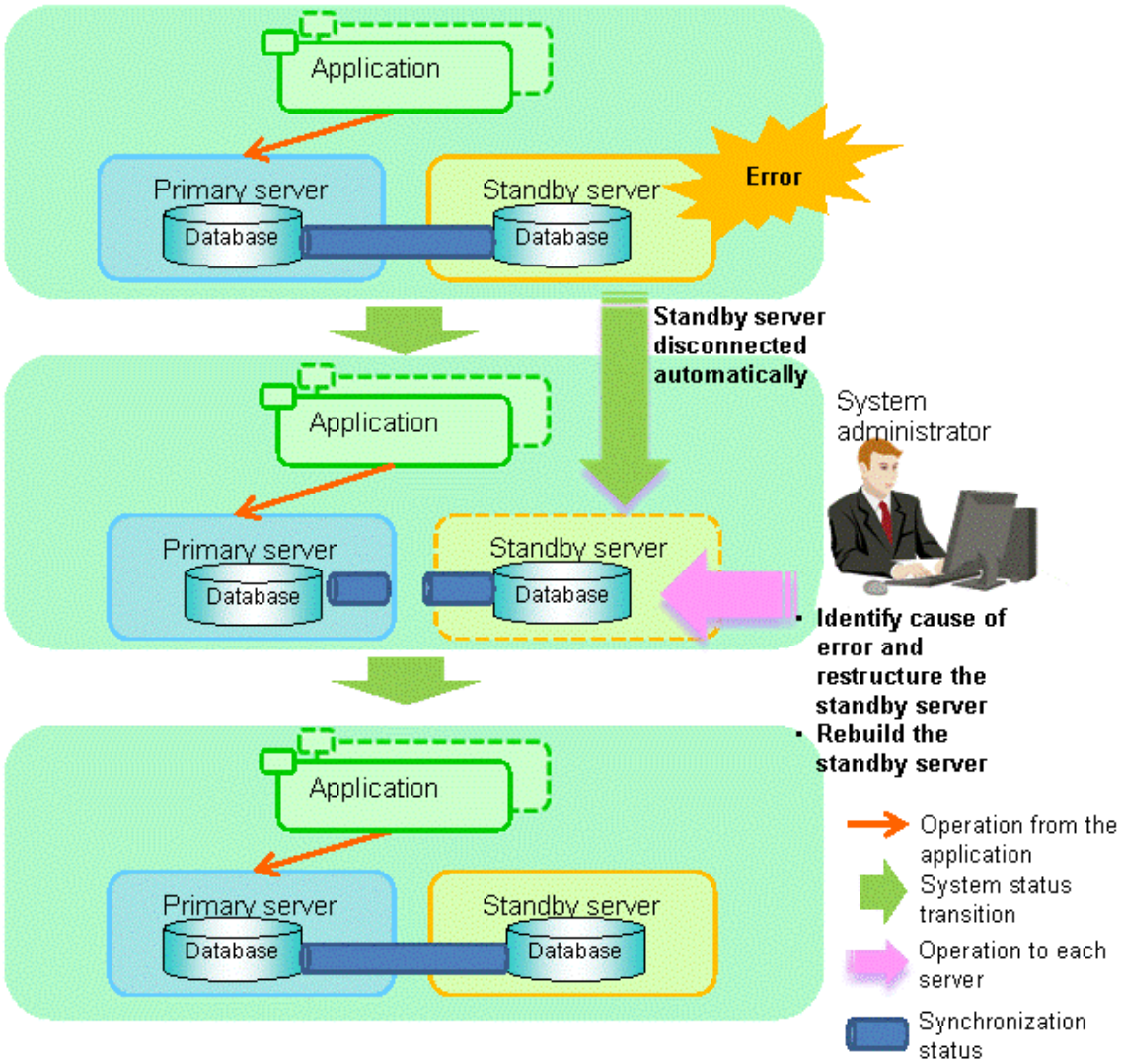
After a disconnection has occurred as a result of an abnormality on the standby server, the database will not have a multiplexed configuration until the standby server is rebuilt. Remove the cause of the error as quickly as possible, and then rebuild the standby server.

If the disconnection occurred and the server has started degrading, perform the following operations to recover the standby server and revert it to its original state:

- Identify Cause of Error and Restore the Standby Server
- Rebuild the Standby Server

The flow of these operations is shown in the figure below.

Figure 4.2 Flow of operations



4.1.2.1 Identify Cause of Error and Restore the Standby Server

Perform the recovery according to the following procedure:

1. Stop Mirroring Controller
2. Recovery of the Mirroring Controller management directory
3. Identify cause of error and perform recovery

4.1.2.1.1 Stop Mirroring Controller

Execute the mc_ctl command in stop mode for the standby server on which the error occurred.

If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

Example)

```
$ mc_ctl stop -M /mcdir/inst1
```

This also stops the instance that is required to perform the recovery.

Note

If the instance does not stop, refer to "Actions in Response to Failure to Stop an Instance" in the Operation Guide, and then stop the instance.

Then, specify the -e option in the above command to forcibly stop Mirroring Controller.

4.1.2.1.2 Recovery of the Mirroring Controller management directory

Copy the files in the Mirroring Controller management directory from the backup data, and then perform the recovery.

4.1.2.1.3 Identify cause of error and perform recovery

Refer to the system logs of the primary server and the standby server to identify the cause of the error, and then perform recovery.

Execute the pg_basebackup command to perform recovery by synchronizing data in the primary server with the standby server.

```
Example)
```

```
$ pg_basebackup -D /database/inst1 -X fetch --waldir=/transaction/inst1 --progress --verbose -R --
dbname='application_name=standbyServerName' -h primaryServerHostName -p primaryServerPortNumber
```

See

This recovery procedure is the same as the procedure for setting up the standby server.

Refer to "2.5.2 Creating, Setting, and Registering the Standby Server Instance", and then perform the recovery.

4.1.2.2 Rebuild the Standby Server

Start the Mirroring Controller and the instance of the standby server, and rebuild the standby server.

Enabling automatic switch/disconnection

As the instance administrator user, execute the mc_ctl command in start mode.

```
Example)
```

```
$ mc_ctl start -M /mcdir/inst1
```

Disabling automatic switch/disconnection

As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

```
Example)
```

```
$ mc_ctl start -M /mcdir/inst1 -F
```

Point

After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

4.1.3 Addressing Errors During Degrading Operation

This section explains how to address errors that may occur on the server on which operation is continuing during degrading operation triggered by a switch or disconnection.

If needing to recover from backup data

If it is necessary to recover the database using backup data due to data becoming corrupted from disk failure or user operation error, refer to the following for information on recovery to database multiplexing mode:

- Action Required when All Database Servers or Instances Stopped
- Recovering from an Incorrect User Operation

If a temporary error occurs

If a temporary error occurs, such as due to a high load on the server or insufficient system resources, remove the cause of the error and restart Mirroring Controller, and then refer to the following for details on recovery to database multiplexing mode:

- Operations when the Server has Started Degrading after a Switch has Occurred
- Operations when the Server has Started Degrading after a Disconnection has Occurred

See

Refer to "3.2 Starting and Stopping Mirroring Controller" for information on restarting Mirroring Controller.

4.2 Action Required when Automatic Switch Fails

If the system behavior is unstable, for example there are insufficient temporary system resources, the Mirroring Controller automatic switch may fail.

Perform the switch manually using one of the following methods:

- Refer to the procedures in "3.4 Manually Switching the Primary Server".
- In the standby server, execute the mc_ctl command in switch mode with the -force option specified to forcibly perform the switch.

  Example)

```
$ mc_ctl switch -M /mcdir/inst1 --force
```

Point

- Even if connection cannot be established between database servers, it is possible to fence the primary server and forcibly switch by executing the mc_ctl command in switch mode with the --force option specified.
- The primary server is not fenced in the cases below, so stop Mirroring Controller and instances of the primary server database in advance:
  - The --no-fencing option is specified when performing forced switch.
  - The heartbeat_error_action parameter in *serverIdentifier*.conf is set to "message" and the fencing command is not configured to be used (the fencing_command parameter is omitted in *serverIdentifier*.conf).
  - The heartbeat_error_action parameter in *serverIdentifier*.conf is set to "fallback".

See

Recovery to database multiplexing mode

Refer to "4.1.1.2 Rebuild the Standby Server" and "4.1.1.3 Failback of the Primary Server" for information on recovery to database multiplexing mode.

4.3 Action Required when Automatic Disconnection Fails

If the system behavior is unstable, for example there are insufficient system resources such as available memory or free disk space, automatic disconnection using Mirroring Controller may not be possible.

Perform the disconnection manually using one of the following methods:

- Refer to the procedures in "3.5 Manually Disconnecting the Standby Server".
- In the primary server, execute the mc_ctl command in detach mode to perform forced disconnection.

  Example)

```
$ mc_ctl detach -M /mcdir/inst1
```

Point

- Even if connection cannot be established between database servers, it is possible to fence the standby server and forcibly disconnect by executing the mc_ctl command in detach mode.
- In the cases below, stop Mirroring Controller and instances of the standby server database in advance so that the standby server is not fenced:
  - The --no-fencing option is specified when performing forced disconnection.
  - The heartbeat_error_action parameter in *serverIdentifier*.conf is set to "message" and the fencing command is not configured to be used (the fencing_command parameter is omitted in *serverIdentifier.conf*).
  - The heartbeat_error_action parameter in *serverIdentifier*.conf is set to "fallback".

See

Recovery to database multiplexing mode

Refer to "4.1.2.2 Rebuild the Standby Server" for information on recovery to database multiplexing mode.

4.4 Action Required when All Database Servers or Instances Stopped

This section explains what happens when all database servers or instances on the database server have stopped, so jobs cannot continue.

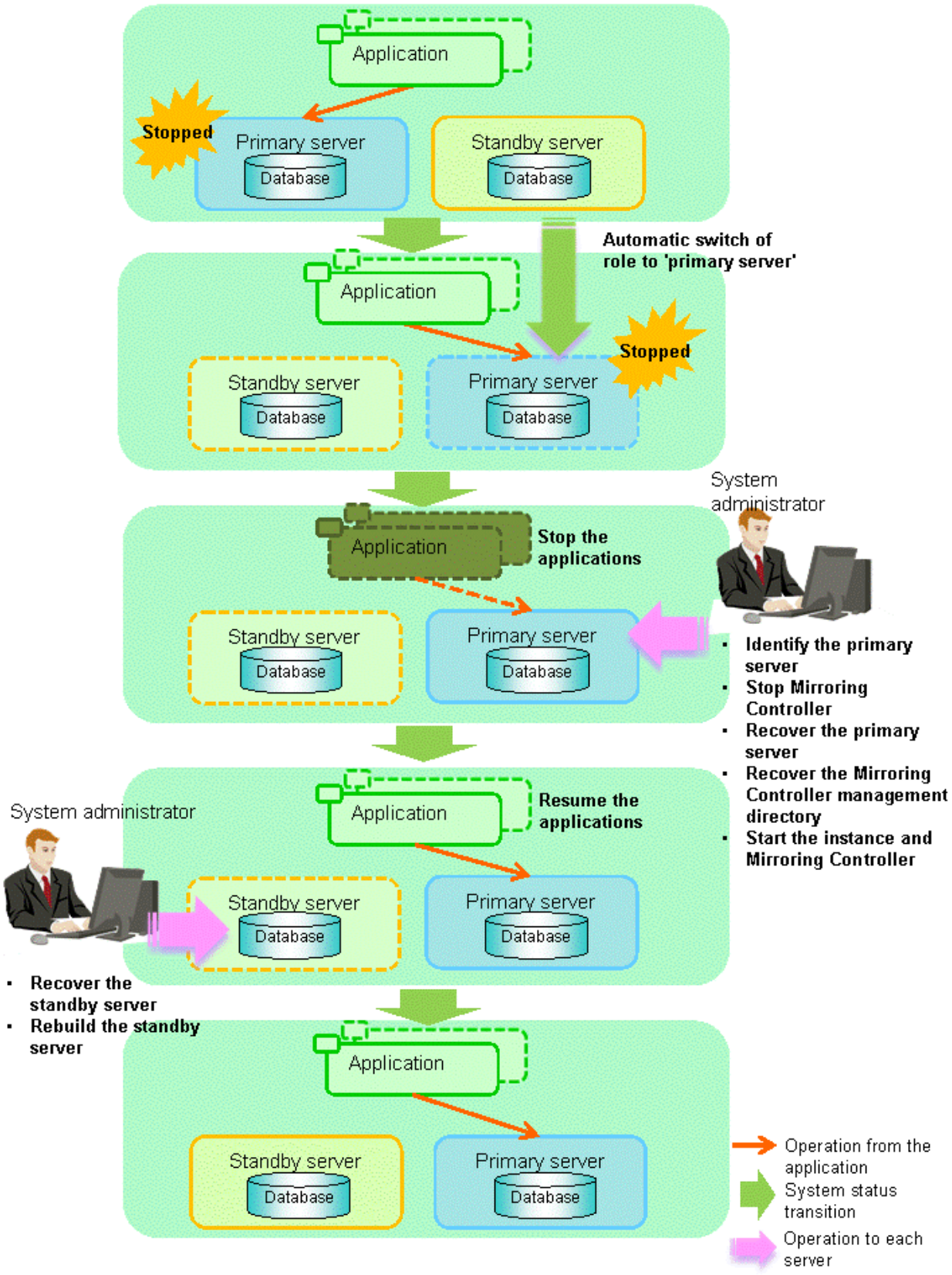
See

Recovery to database multiplexing mode

Refer to "4.1.1.2 Rebuild the Standby Server" and "4.1.1.3 Failback of the Primary Server" for information on recovery to database multiplexing mode.

The flow of these recovery operations is shown in the figure below.

Figure 4.3 Flow of operations



Perform the following procedure:

1. Stop the applications

   Stop the applications from running.

2. Identify the primary server

   Use one of the following methods to identify the primary server that was running before the servers or instances stopped:

   - Refer to the system log on each server and identify the server where the following message was output.

     Message:

     ```
     MirroringControllerOpen[30017]: LOG:  promotion processing completed (MCA00062)
     ```

   - On each server, execute the mc_ctl command in status mode to search the servers for which "none(inactivated primary)" is displayed.

3. Stop Mirroring Controller on the primary server

   Execute the mc_ctl command in stop mode on the primary server.

   If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

   Example)

   ```
   $ mc_ctl stop -M /mcdir/inst1
   ```

   Note

   Forcibly stopping Mirroring Controller

   If Mirroring Controller does not stop, specify the -e option in the stop mode of the mc_ctl command and then execute the command.

   Example)

   ```
   $ mc_ctl stop -M /mcdir/inst1 -e
   ```

4. Recover the primary server

   First, refer to "Actions when an Error Occurs" in the Operation Guide, and then identify the cause of the error and perform recovery.

   Next, recover the primary server using the recovery method that uses the pgx_rcvall command based on the backup data.

   If the backup operation was performed using the pgx_dmpall command based on the instructions in "2.14.2 Database Backup Operation", perform the following procedure for the recovery:

   a. Perform the following operations on both the primary server and the standby server, and check the server containing the backup data and the archive log that show the latest date.

      - Execute the pgx_rcvall command with the -l option specified and identify the backup data that shows the latest date.
      - Identify the archive log that shows the latest date, as shown below.

        Example)

        ```
        $ ls -ltr backupDataStorageDir/*_wal
        ```

   b. If the latest backup data exists on the standby server, copy (*1) the backup data and overwrite (*2) it to each backup storage destination directory on the primary server.

   c. If the latest archive log and transaction log file exist on the standby server, copy (*1) the archive log and overwrite (*2) it to the backup storage destination directory on the primary server.

   d. Execute the pgx_rcvall command on the primary server, specifying the backup storage destination directory of the primary server.

Note

*1: The backup data may contain a symbolic link, so copy the backup data so that the symbolic link is not converted to an ordinary file (with the tar command, for example).

*2: If you can save a copy of the backup storage destination directory, do so without overwriting it.

See

Refer to "Actions when an Error Occurs" in the Operation Guide for information on the pgx_rcvall command.

5. Recover the Mirroring Controller management directory

   Copy the files in the Mirroring Controller management directory from the backup data, and then perform the recovery.

6. Start the primary server instance and Mirroring Controller

   Enabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode.

   Example)

   ```
   $ mc_ctl start -M /mcdir/inst1
   ```

   Disabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

   Example)

   ```
   $ mc_ctl start -M /mcdir/inst1 -F
   ```

Point

After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

7. Resume applications

   Resume the applications.

8. Stop Mirroring Controller on the standby server

   Execute the mc_ctl command in stop mode on the standby server.

   If automatic start and stop of Mirroring Controller has been configured using systemd, do not use the mc_ctl command, but instead use the systemctl command. Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

   Example)

   ```
   $ mc_ctl stop -M /mcdir/inst1
   ```

9. Recover the standby server

   Refer to "2.5.2 Creating, Setting, and Registering the Standby Server Instance", and then recover (set up) the standby server from the primary server.

10. Rebuild the standby server

    On the standby server, start Mirroring Controller and the instance.

Enabling automatic switch/disconnection

As the instance administrator user, execute the mc_ctl command in start mode.

Example)

```
$ mc_ctl start -M /mcdir/inst1
```

Disabling automatic switch/disconnection

As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

Example)

```
$ mc_ctl start -M /mcdir/inst1 -F
```

Point

After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

4.5 Recovering from an Incorrect User Operation

This section describes how to recover an instance when data has been corrupted due to incorrect user operation.

For example, when data has been corrupted due to incorrect user operation, such as data being unintentionally changed or deleted by an application or command, it is necessary to restore the original data on the primary server and resynchronize with the standby server.

Use the following procedure to perform recovery.

1. Identify the primary server

   Execute the mc_ctl command in status mode on each server, and search for a server for which "primary" or "none(inactivated primary)" is displayed.

2. Stop the applications and commands that caused the incorrect operation to occur

   Stop applications and commands that are running on the primary server. This will minimize the impact caused by the incorrect data.

   Also, if any applications used for reference by the standby server are running, stop them too.

3. Stop the instance and Mirroring Controller

   Stop the instance and Mirroring Controller on both the primary server and standby server.

   Example)

   ```
   $ mc_ctl stop -a -M /mcdir/inst1
   ```

4. Recover the database on the primary server

   Recover the database using the recovery method in which the pgx_rcvall command uses the backup data to recover the database to a restore point prior to the time when the incorrect operation was performed.

   See

   Refer to "Recovering from an Incorrect User Operation" in the Operation Guide for information on using the pgx_rcvall command to recover the database to a restore point, and then perform only the database recovery procedure while the instance is in a stop state.

5. Start the instance and Mirroring Controller

   Start the instance and Mirroring Controller on the primary server.

   Enabling automatic switch/disconnection

   As the instance administrator user, execute the mc_ctl command in start mode.

Example)

```
$ mc_ctl start -M /mcdir/inst1
```

Disabling automatic switch/disconnection

As the instance administrator user, execute the mc_ctl command in start mode with the -F option specified.

Example)

```
$ mc_ctl start -M /mcdir/inst1 -F
```

Point

After Mirroring Controller is started, automatic switch/disconnection can be enabled or disabled using the enable-failover or disable-failover mode of the mc_ctl command.

6. Build the new standby server

   Refer to "2.5 Setting Up the Standby Server" for information on building (setting up) a standby server from the primary server.

Chapter 5 Managing Mirroring Controller Using WebAdmin

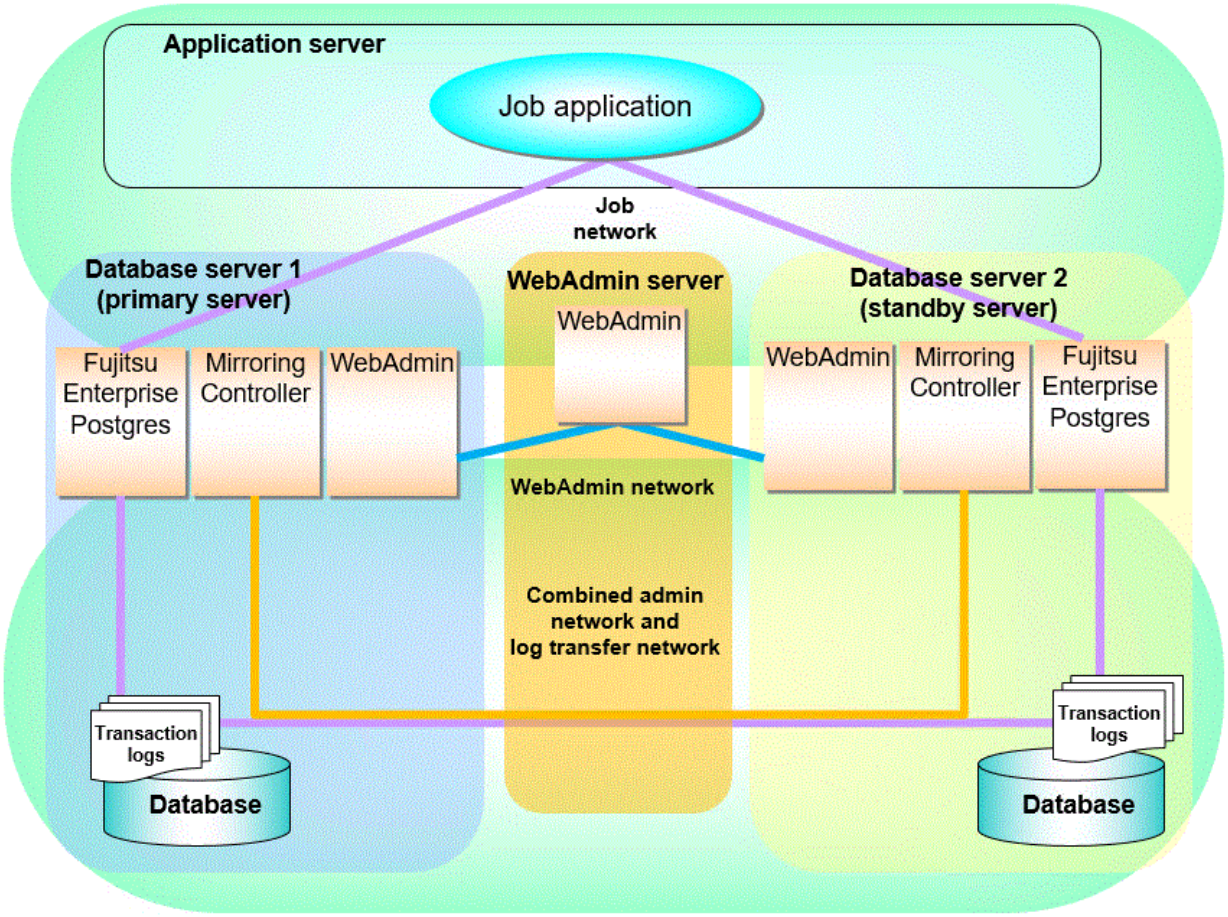
This chapter describes how to set up and manage Mirroring Controller in a streaming replication cluster using WebAdmin.

Mirroring Controller can be used to monitor a streaming replication cluster and perform automatic switching or disconnect synchronous replication when there is an error.

WebAdmin can be used to set up Mirroring Controller in an existing replication cluster. Mirroring Controller can be set up for either synchronous standby instances or asynchronous standby instances.

The configuration of the database multiplexing system built using WebAdmin is shown below:

Figure 5.1 Configuration of database multiplexing operation system using WebAdmin



Point

- If Mirroring Controller is set up to the replication cluster using WebAdmin, the network with the host name (or IP address) specified in [Host name] will be used as the admin network and the log transfer network.
- To use a network other than the job network as the log transfer network, before building the replication cluster specify a host name other than the job network one in [Host name].

Note

If you set up the arbitration server using WebAdmin, install WebAdmin on the arbitration server.

5.1 Mirroring Controller Setup

Perform the following procedure to set up Mirroring Controller in a streaming replication cluster.

1. In the [Instances] tab, select the standby instance on which Mirroring Controller needs to be set up.
2. Click  .
3. Enter the information for the Mirroring Controller to be set up.
   - [Enable automatic switch over]: Toggles the automatic switch/disconnection functionality. Select "Yes". The default is "No".
   - [Mirroring Controller management directory]: Directory where the Mirroring Controller configuration files will be stored. When the [Mirroring Controller management directory] is entered, WebAdmin will search the Mirroring Controller configuration files in the entered directory based on the [Data storage path] of the corresponding DB instance. If Mirroring Controller configuration files are found, the Mirroring Controller fields will be auto filled.
   - [Mirroring Controller port]: Port number of Mirroring Controller.
   - [Heartbeat interval (milliseconds)]: Number of milliseconds between two consecutive heartbeat checks. The default is "800".
   - [Heartbeat timeout (seconds)]: Number of seconds for the heartbeat timeout. The default is "1".
   - [Heartbeat retry]: Number of retries for heartbeat monitoring, before failover occurs. The default is "2".
   - [Heartbeat error action]: Operation when a heartbeat abnormality is detected. The default is "Fallback".

     When using FUJITSU Enterprise Postgres 11 instance which was created with FUJITSU Enterprise Postgres WebAdmin 11, the mode in the [Heartbeat error action] for Mirroring Controller setup cannot be changed.

   When the [Heartbeat error action] is set to "Arbitration", the following extra items are displayed:

   - [Arbitration network IP address]: IP address of the arbitration network.
   - [Mirroring Controller Arbitration port]: Port number of Mirroring Controller for communicating with the arbitration server.

   The [Arbitration server configuration] section is also displayed with the following items. The [Arbitration server configuration] will not be auto filled.

   - [Location]: Location of the arbitration server. "Local" or "Remote" can be selected depending on your configuration.

     If the arbitration server and WebAdmin server are located on the same server, you can select "Local" and the following items are displayed:

     - [Arbitration management directory]: Directory where the arbitration server configuration files will be stored.
     - [Arbitration server host or IP address]: Host name or IP address of the arbitration server.
     - [Arbitration process port]: Port number for the arbitration process.
     - [Fencing command]: Full path of the fencing command that fences a database server when an abnormality is detected.

     If "Remote" is set for the item, the items below are displayed in addition to the above items.

     - In the [Arbitration server configuration] section, [Operating system credential] is displayed where you can enter the following information:

       [User name]: User name to access the arbitration server.

       [Password]: Password to access the arbitration server.

     - In the [Remote WebAdmin for Arbitration server] section, the following items are displayed:

       [Remote WebAdmin address]: IP address of the remote WebAdmin installed on the arbitration server.

       [Remote WebAdmin port]: Port number for the WebAdmin installed on the arbitration server.

When the [Heartbeat error action] is set to "Command", the following extra items are displayed:

- [Arbitration command]: Full path of the arbitration command to be executed when an abnormality is detected.
- [Fencing command]: Full path of the fencing command that fences a database server when an abnormality is detected.

4. Click to set up Mirroring Controller.
5. Upon successful completion, Mirroring Controller will be started on master and standby instances.

After the Mirroring Controller has been set up, ([Edit Mirroring Controller] button) and ([Mirroring Controller Configuration] button) are available.

When the [Heartbeat error action] is "Arbitration", the following information is displayed: whether the arbitration status is "online" or "offline", the arbitration server IP address and the arbitration process port.

Note

Operating system credential (User name, Password) should not contain hazardous characters. Refer to "Appendix E WebAdmin Disallow User Inputs Containing Hazardous Characters".

5.2 Edit Mirroring Controller Setup

Settings made in "5.1 Mirroring Controller Setup" can be updated in either the master instance or a standby instance using WebAdmin.

Perform the following procedure to edit Mirroring Controller configuration:

1. In the [Instances] tab, select the instance for which the Mirroring Controller configuration is to be edited.
2. Click .
3. Enter the information for the Mirroring Controller to be updated. Refer to "5.1 Mirroring Controller Setup".
4. Click to update the Mirroring Controller.
5. Upon successful completion, Mirroring Controller will be started on master and standby instances.

Editing and saving the [Edit Mirroring Controller] page will reset all other settings that are not listed on this page to default values.

5.3 Mirroring Controller Configuration

The information related to Mirroring Controller monitoring and control (refer to "A.4.1 Server Configuration File for the Database Servers") and the information related to arbitration and control of the Mirroring Controller arbitration process (refer to "A.4.2 Arbitration Configuration File") can be set using WebAdmin. You can view and update the configuration on either the master instance or the standby instance.

Perform the following procedure:

1. In the [Instances] tab, select the instance for the Mirroring Controller configuration you want to view.
2. Click to view the Mirroring Controller configuration.
3. Click to show the editing page for the Mirroring Controller configuration. The Mirroring Controller configurations defined during [Mirroring Controller Setup] are read-only on this page. Refer to "5.1 Mirroring Controller Setup".

   Additionally, refer to the "Appendix A Parameters" for information about the settings and the corresponding parameter names.

   The items common to all [Heartbeat error action] are:

   - Target DB
   - Core file path
   - Syslog facility

- Syslog identity
- Remote call timeout (milliseconds)
- Agent alive timeout (seconds)
- DB instance check interval (milliseconds)
- DB instance check timeout (seconds)
- DB instance check retry
- DB instance timeout action
- Disk check interval (milliseconds)
- Disk check retry
- Tablespace directory error action
- Post-switch command
- Post-promote command
  (Post-promote command is replaced in FUJITSU Enterprise Postgres 12. The Post-promote command is still valid and will be displayed when it is used in the server configuration file of Mirroring Controller.)
- Post-attach command
- Pre-detach command
- State transition command timeout (seconds)
- Check synchronous standby names validation

When the [Heartbeat error action] is set to "Arbitration", the following extra items are displayed:

- Arbitration timeout (seconds)
- Arbiter alive interval (milliseconds)
- Arbiter alive retry
- Arbiter alive timeout (seconds)
- Arbiter connect interval (milliseconds)
- Arbiter connect timeout (seconds)
- Fencing command
- Fencing command timeout (seconds)
- Shutdown detached synchronous standby

When the [Heartbeat error action] is set to "Arbitration", the [Arbitration server configuration] section is displayed with the following items:

- Core file path
- Syslog facility
- Syslog identity
- Fencing command timeout (seconds)
- Heartbeat interval (milliseconds)
- Heartbeat timeout (seconds)

- Heartbeat retry

When the [Heartbeat error action] is set to "Command", the following extra items are available:

- Fencing command timeout (seconds)
- Arbitration command timeout (seconds)
- Shutdown detached synchronous standby

When the [Heartbeat error action] is set to "Message", the following extra items are available:

- Fencing command
- Fencing command timeout (seconds)

In addition, the following configurations are provided:

- DB instance JDBC connection SSL parameters
- DB instance libpq connection SSL parameters

4. Click  to update the Mirroring Controller configurations.

5.4 Stopping Mirroring Controller

Mirroring Controller can be stopped either in master instance or in standby instance using WebAdmin.

Perform the following procedure to stop Mirroring Controller.

1. In the [Instances] tab, select the instance where to stop Mirroring Controller.
2. Click  .
3. In the confirmation dialog box, click [Yes].

Mirroring Controller will be stopped on the selected instance. The Mirroring Controller status will be updated, and a confirmation message entry will be displayed in the [Message] section.

5.5 Starting Mirroring Controller

Mirroring Controller can be started either in master instance or in standby instance using WebAdmin.

Perform the following procedure to start Mirroring Controller.

1. In the [Instances] tab, select the instance where to start Mirroring Controller.
2. Click  .
3. In the confirmation dialog box, select the desired failover mode, and then click [Yes].

Mirroring Controller will be started on the selected instance. The Mirroring Controller status will be updated, and a confirmation message entry will be displayed in the [Message] section.

5.6 Disabling Failover Mode

Disabling failover mode in Mirroring Controller disables automatic switch/disconnection between master and standby instances.

Perform the following procedure to disable failover mode.

1. In the [Instances] tab, select the instance.

2. Click  .
3. In the confirmation dialog box, click [Yes].

Failover mode will be disabled in Mirroring Controller. The Mirroring Controller status will be updated and a confirmation message entry will be displayed in the [Message] section.

5.7 Enabling Failover Mode

Enabling failover mode in Mirroring Controller enables automatic switch/disconnection between master and standby instances.

Perform the following procedure to enable failover.

1. In the [Instances] tab, select the instance.
2. Click  .
3. In the confirmation dialog box, click [Yes].

Failover mode will be enabled in Mirroring Controller. The Mirroring Controller status will be updated and a confirmation message entry will be displayed in the [Message] section.

5.8 Deleting Mirroring Controller Setup

Deleting Mirroring Controller setup removes its setup from master and standby instances.

1. In the [Instances] tab, select the instance.
2. Click  .
3. In the confirmation dialog box, click [Yes].

Mirroring Controller setup will be removed from the cluster. The cluster status will be updated and a confirmation message entry will be displayed in the [Message] section.

For the instances in FUJITSU Enterprise Postgres 12 or later, WebAdmin does not delete the Mirroring Controller management directory and the configuration files.

5.9 Status Update after Failover

When Mirroring Controller performs a failover, standby instance will be promoted to standalone instance. The Mirroring Controller setup will be removed from both standby and master instances.

The following scenario describes one of the ways in which failover can be triggered, and the results achieved by the use of Mirroring Controller in WebAdmin.

1. In the [Instances] tab, select the master instance "inst1".
2. Click  .
3. In the confirmation dialog box, the warning "This instance is being monitored by Mirroring Controller. Stopping the instance may result in cluster failover." is displayed.
4. Choose the stop mode and click [Yes].

   In the server, the following takes place:

   a. The master instance is stopped.
   b. Failover is triggered in Mirroring Controller.
   c. The Mirroring Controller setup is removed from both master and standby instances
   d. Standby instance is promoted to standalone.
5. When the instance is refreshed in WebAdmin, the latest status of the instances will be displayed.

Note

When failover is performed, the Mirroring Controller setup is removed from both master and standby instances. Therefore, to manage the Mirroring Controller using WebAdmin again, create the standby instance and set up Mirroring Controller.

Refer to "Creating a Standby Instance" in the Operation Guide for details.

Refer to "5.1 Mirroring Controller Setup" for details.

5.10 Action Required when an Error Occurs in the Combined Admin Network and Log Transfer Network

Communication errors may temporarily occur in the network used as the admin network and log transfer network due to reasons such as high load on the server or insufficient system resources. Because of this, there is a risk of causing a split-brain situation by mistake even though the server has no issues.

Split brain is a phenomenon in which both servers temporarily operate as primary servers, causing data updates to be performed on both servers.

How to detect split brain using WebAdmin

If the conditions below are met, split brain may occur. Refer to "Split-brain detection method" and "How to recover from a split-brain" in "Appendix D Notes on Performing Automatic Degradation Immediately after a Heartbeat Abnormality" and take the actions described.

1. A standby instance is selected in the [Instances] tab, and
2. "Standalone" is displayed in [Instance type], and
3. A master instance is selected in the [Instances] tab, and
4. "Standalone" is displayed in [Instance type].

Note

The admin network is important because Mirroring Controllers use it to confirm the status of each server.

The log transfer network is also important to maintain the data freshness.

Therefore, use network configurations resistant to faults for these networks by using the network redundancy channel bonding feature provided by the operating system or network driver vendor.

5.11 Performing Automatic Degradation Using the Arbitration Server

If database multiplexing is performed using WebAdmin, it is also possible to perform automatic degradation using the arbitration server. In such cases, it is necessary to perform tasks on the database server and the arbitration server after setting up Mirroring Controller in WebAdmin.

Tasks on the arbitration server

Perform setup of the arbitration server using Mirroring Controller commands.

1. Set up the arbitration server.

   Refer to "2.3 Setting Up the Arbitration Server" in "Chapter 2 Setting Up Database Multiplexing Mode" for information on how to set up the arbitration server.

Tasks on the database server

Change some of the settings after setting up Mirroring Controller in WebAdmin.

1. Set up Mirroring Controller in WebAdmin.

   Refer to "5.1 Mirroring Controller Setup" for details.

2. Use WebAdmin to stop Mirroring Controller on the master and standby instances.

   Refer to "5.4 Stopping Mirroring Controller" for details.

3. Edit the network configuration file of the master and standby instances, and add the arbitration server information.

   The network configuration file is network.conf, which exists in the Mirroring Controller management directory specified during Mirroring Controller setup. Refer to "A.3 Network Configuration File" for details.

   A definition example of network.conf is shown below.

   ```
   Example:
   The port number of the database server to be used as the arbitration network is set to "27541".
   The ID of the server of the Mirroring Controller arbitration process is set to "arbiter", and its
   port number is set to "27541".
   ```

   ```
   dbsvm27500 192.0.2.100,192.0.3.100 27540,27541 server
   dbsvs27500 192.0.2.110,192.0.3.110 27540,27541 server
   arbiter 192.0.3.120 27541 arbiter
   ```

   Note

   - Ensure that the port numbers set for the database server and the arbitration server do not conflict with other software. In addition, do not configure the same segment for the admin network and the arbitration network.
   - If the server type is "server", two IP addresses or host names, and two port numbers need to be specified in the following order:
     - IP address or host name of the database server used as the admin network
     - IP address or host name of the database server used as the arbitration network
     - Port number of the database server used as the admin network
     - Port number of the database server used as the arbitration network
   - If the server type is "arbiter", specify the IP address or host name set for the my_address parameter and the port number set for the port parameter in arbitration.conf of the arbitration server.
   - WebAdmin also support editing mirroring controller configuration via Use WebAdmin to edit Mirroring Controller configurations.
     Refer to "5.2 Edit Mirroring Controller Setup" for detailes.

4. Edit the server configuration file of the master and standby instances, and add the parameters required for automatic degradation using the arbitration server.

   The server configuration file is *instanceName*.conf or *instancePort*.conf, which exists in the Mirroring Controller management directory specified during Mirroring Controller setup.

   To perform automatic degradation using the arbitration server, set the heartbeat_error_action parameter to "arbitration".

   Refer to "A.4.1 Server Configuration File for the Database Servers" for information on other parameters.

5. Use WebAdmin to start Mirroring Controller on the master and standby instances.

   Refer to "5.5 Starting Mirroring Controller" for details.

Common tasks

1. Use the Mirroring Controller command to check the connection status from the database server or the arbitration server.

   Refer to "2.8 Checking the Connection Status" for information on how to check the connection status.

Appendix A Parameters

This appendix describes the configuration files and parameters required by the database multiplexing mode.

See

Refer to "Server Configuration" in the PostgreSQL Documentation for information on the postgresql.conf file.

A.1 Parameters Set on the Primary Server

The content for the parameters set in the postgresql.conf file of the primary server is shown in the table below.

Table A.1 postgresql.conf file

Parameter	Value set	Explanation
wal_level	replica or logical	Specify the output level for the transaction log. Specify "logical" when logical decoding is also to be used.
max_wal_senders	2 or more	Specify "2" when building a Mirroring Controller cluster system. When additionally connecting asynchronous standby servers to the cluster system, add the number of simultaneous connections from these standby servers.
synchronous_standby_names	'*standbyServerName*'	Use single quotation marks (') to enclose the name that will identify the standby server. Any name can be specified. Do not change this parameter while Mirroring Controller is running. Do not specify multiple names to this parameter as the Mirroring Controller can manage only one standby server.
synchronized_standby_slots	'*physicalReplicationSlotName*'	Specify this parameter if the primary server will be a logical replication publication. Setting this parameter ensures that the subscriber is updated after WAL is sent to the standby server. This allows logical replication to continue if the primary server fails and the standby server is promoted. Do not change this parameter while the Mirroring Controller is running. Because the Mirroring Controller can manage only one standby server, do not specify multiple names for this parameter.
hot_standby	on	Specify whether queries can be run on the standby server. Specify "on".
wal_keep_size	WAL save size (megabytes)	If a delay exceeding the value set in this parameter occurs, the WAL segment required later by the primary server may be deleted. Additionally, if you stop a standby server (for maintenance, for example), consider the stop time and set a value that will not cause the WAL segment to be deleted. Refer to "Estimating Transaction Log Space Requirements" in the Installation and Setup Guide for Server for information on estimating the WAL save size.

Parameter	Value set	Explanation
wal_log_hints	on	When using the pg_rewind command to recover a standby server, specify this parameter or enable checksums when executing the initdb command.
wal_sender_timeout	Timeout (milliseconds)	Specify the time period after which it is determined that the receiver process (walreceiver) of the transaction log is in an abnormal state on the primary server. The specified value must be larger than the value set for the wal_receiver_status_interval parameter set in the postgresql.conf file of the standby server. By aligning this value with the value for the database process heartbeat monitoring time, you can unify the time after which it is determined that an error has occurred.
wal_receiver_timeout	Timeout (milliseconds)	Specify the time period after which it is determined that an error has occurred when the transaction log was received on the standby server. By aligning this value with the value for the database process heartbeat monitoring time, you can unify the time after which it is determined that an error has occurred.
archive_mode	on	Specify the archive log mode.
archive_command	'*installDir*/bin/pgx_walcopy.cmd "%p" "*backupDataStorageDestinationDirectory*/archived_wal/%f"'	Specify the command and storage destination to save the transaction log.
backup_destination	Backup data storage destination directory	Specify the name of directory where to store the backup data. Set the permissions so that only the instance administrator user can access the specified directory. Specify the same full path on all servers, so that the backup data of other servers can be used to perform recovery.
listen_addresses	Primary server IP address, host name, or "*"	Specify the IP address or host name of the primary server. Specify the IP address or corresponding host name that will be used to connect to the log transfer network. The content specified is also used to allow connections from client applications. To receive the connection and the transaction log from any client or standby server, specify "*". Refer to "Connections and Authentication" in the PostgreSQL Documentation for details.
max_connections	Number of simultaneous client connections to the instance + superuser_reserved_connections value	The value specified is also used to restrict the number of connections from client applications and the number of connections for the management of instances. Refer to "When an Instance was Created with the initdb Command" in the Installation and Setup Guide for Server, and "Connections and Authentication" in the PostgreSQL Documentation, for details.
superuser_reserved_connections	Add the number of simultaneous executions of mc_ctl status (*1) + 2	Specify the number of connections reserved for connections from database superusers.

Parameter	Value set	Explanation
		Add the number of connections from Mirroring Controller processes. Also reflect the added value in the max_connections parameter.
restart_after_crash	off	If "on" is specified, or the default value is used for this parameter, behavior equivalent to restarting Fujitsu Enterprise Postgres, including crash recovery, will be performed when some server processes end abnormally. However, when database multiplexing monitoring is used, a failover will occur after an error is detected when some server processes end abnormally, and the restart of those server processes is forcibly stopped. Specify "off" to prevent behavior such as this from occurring for no apparent reason.
synchronous_commit	on or remote_apply	Specify up to what position WAL send is to be performed before transaction commit processing returns a normal termination response to a client. Set "on" or "remote_apply" to prevent data loss caused by operating system or server down immediately after a switch or switch.
recovery_target_timeline	latest	Specify "latest" so that the new standby server (original primary server) will follow the new primary server when a switch occurs. This parameter is required when the original primary server is incorporated as a new standby server after the primary server is switched.

*1: Number of simultaneous executions of the mc_ctl command in the status mode.

A.2 Parameters Set on the Standby Server

This section explains the content of the file and parameters set on the standby server. After editing postgresql.conf file, start the instance.

The content for the parameters specified in postgresql.conf file is shown in the table below.

Table A.2 postgresql.conf file

Parameter	Value set	Explanation
wal_level	replica or logical	Specify the output level for the transaction log. Specify "logical" when logical decoding is also to be used.
max_wal_senders	2 or more	Specify "2" when building a Mirroring Controller cluster system. When additionally connecting asynchronous standby servers to the cluster system, add the number of simultaneous connections from these standby servers.
synchronous_standby_names	'*primaryServerName*'	Use single quotation marks (') to enclose the name that will identify the primary server. Any name can be specified. This name will be required to rebuild the original primary server as the new standby server after the primary server was switched.

Parameter	Value set	Explanation
		Do not change this parameter while Mirroring Controller is running. Do not specify multiple names to this parameter as the Mirroring Controller can manage only one standby server.
hot_standby	on	Specify whether queries can be run on the standby server. Specify "on".
wal_keep_size	WAL save size (megabytes)	If a delay exceeding the value set in this parameter occurs, the WAL segment required later by the standby server may be deleted by the primary server. Additionally, if you stop a standby server (for maintenance, for example), consider the stop time and set a value that will not cause the WAL segment to be deleted. Refer to "Estimating Transaction Log Space Requirements" in the Installation and Setup Guide for Server for information on estimating the WALsave size.
wal_log_hints	on	When using the pg_rewind command to recover a standby server, specify this parameter or enable checksums when executing the initdb command.
wal_sender_timeout	Timeout (milliseconds)	Specify the time period after which it is determined that the receiver process (walreceiver) of the transaction log is in an abnormal state on the primary server. The specified value must be larger than the value set for the wal_receiver_status_interval parameter set in the postgresql.conf file of the standby server. By aligning this value with the value for the database process heartbeat monitoring time, you can unify the time after which it is determined that an error has occurred.
wal_receiver_timeout	Timeout (milliseconds)	Specify the time period after which it is determined that an error has occurred when the transaction log was received on the standby server. By aligning this value with the value for the database process heartbeat monitoring time, you can unify the time after which it is determined that an error has occurred.
backup_destination	Backup data storage destination directory	Specify the name of the backup data storage directory. Set the permissions so that only the instance administrator user can access the specified directory. Specify the same full path on all servers so that the backup data of other servers can be used to perform recovery.
archive_mode	on	Specify the archive log mode.
archive_command	'*installDir*/bin/pgx_walcopy.cmd "%p" "*backupDataStorageDestinationDirectory*/archived_wal/%f"'	Specify the command and storage destination to save the transaction log.

Parameter	Value set	Explanation
listen_addresses	Standby server IP address, host name, or "*"	Specify the IP address or host name of the standby server. Specify the IP address or corresponding host name that will be used to connect to the log transfer network. The content specified is also used to allow connections from client applications. To receive the connection and the transaction log from any client or standby server, specify "*". Refer to "Connections and Authentication" in the PostgreSQL Documentation for details.
max_connections	Number of simultaneous client connections to the instance + superuser_reserved_connections value	The value specified is also used to restrict the number of connections from client applications and the number of connections for the management of instances. Refer to "When an Instance was Created with the initdb Command" in the Installation and Setup Guide for Server, and "Connections and Authentication" in the PostgreSQL Documentation, for details.
superuser_reserved_connections	Add the number of simultaneous executions of mc_ctl status (*1) + 2	Specify the number of connections reserved for connections from database superusers. Add the number of connections from Mirroring Controller processes. Also reflect the added value in the max_connections parameter.
restart_after_crash	off	If "on" is specified, or the default value is used for this parameter, behavior equivalent to restarting Fujitsu Enterprise Postgres, including crash recovery, will be performed when some server processes end abnormally. However, when database multiplexing monitoring is used, a failover will occur after an error is detected when some server processes end abnormally, and the restart of those server processes is forcibly stopped. Specify "off" to prevent behavior such as this from occurring for no apparent reason.
synchronous_commit	on or remote_apply	Specify up to what position WAL send is to be performed before transaction commit processing returns a normal termination response to a client. Set "on" or "remote_apply" to prevent data loss caused by operating system or server down immediately after a switch or switch.
primary_conninfo	'*streamingReplicationConnectionDestinationInfo*'	Use single quotation marks (') to enclose the connection destination information of the streaming replication. The default value of this parameter is automatically set to postgresql.auto.conf in the procedure to run pg_basebackup for instance setup.
recovery_target_timeline	latest	Specify "latest" so that the new standby server (original primary server) will follow the new primary server when a switch occurs. This parameter is required when the original primary server is incorporated as a new standby server after the primary server is switched.

Parameter	Value set	Explanation
streaming_wal_compression	Any number from -1 to 9	Specifies that you want to send a compressed WAL by streaming replication. -1: Compress at zlib's default compression level 0: Uncompressed 1-9: Compress at specified compression level, 9 is high compression Default is uncompressed.

A.3 Network Configuration File

This section explains the network configuration file (network.conf) to be defined individually for the database servers and the arbitration server. Define the same content on the primary server and standby server.

For database multiplexing mode, define the network configuration for the following in network.conf.

- Integration between Mirroring Controller processes
- Integration between a Mirroring Controller process and the Mirroring Controller arbitration process

Items to be defined in network.conf

Format:

```
serverIdentifier hostName[,hostName] portNum[,portNum] [serverType]
Or,
serverIdentifier ipAddr[,ipAddr] portNum[,portNum] [serverType]
```

Specify the server identifier, IP address or host name, port number, and server type, using a space as the delimiter.

The items are explained in the table below.

Table A.3 network.conf file

Item	Description
serverIdentifier	Specify any identifier for the server. The maximum length is 64 bytes. Use ASCII characters excluding spaces and number signs (#) to specify this parameter.
ipAddrOrHostName	Specify the IP address or its corresponding host name that will connect to the admin network that performs communication between the database servers, and to the arbitration network that performs communication between a database server and the arbitration server. When specifying two IP addresses or host names delimited by a comma, do not insert a space after the comma. Use ASCII characters excluding spaces to specify the host name.
portNum	A port number cannot be specified if it exceeds the range 0 to 65535. Ensure that the port number does not conflict with other software. Do not specify an ephemeral port that may temporarily be assigned by another program. Note that the value specified in this parameter must also be set in the services file. When specifying two port numbers delimited by a comma, do not insert a space after the comma.
serverType	Specify "server" for a database server ("server" can be omitted), or "arbiter" for the arbitration server.

Content to be defined on the database servers

This section explains the network.conf content to be defined on the database servers.

The content to be defined depends on the operation settings at the time a heartbeat abnormality is detected.

When automatic degradation by the arbitration server is selected

- Specify definitions related to the admin network and arbitration network.

- Specify the IP address or host name and port number according to the server type (database server or arbitration server) as shown in the table below.

Server type	IP address or host name		Port number	
	First	Second	First	Second
server	IP address or host name used as the admin network	IP address or host name used as the arbitration network (*1)	Port number used as the admin network	Port number used as the arbitration network (*1)
arbiter	IP address or host name of the arbitration server Specify the same value as that specified in the my_address parameter of arbitration.conf on the arbitration server.	Not required	Port number on the arbitration server Specify the same value as that specified in the port parameter of arbitration.conf on the arbitration server.	Not required

*1: This value can be omitted from definitions not related to the local server. If it is omitted, network.conf must be created on both the primary server and standby server.

Example)

IPv4

```
server1 192.0.2.100,192.0.3.100 27540,27541 server
server2 192.0.2.110,192.0.3.110 27540,27541 server
arbiter 192.0.3.120 27541 arbiter
```

IPv6

```
server1 2001:258:8404:1217:250:56ff:fea7:559f,2001:258:8404:1217:250:56ff:fea8:559f
27540,27541 server
server2 2001:258:8404:1217:250:56ff:fea7:55a0,2001:258:8404:1217:250:56ff:fea8:55a0
27540,27541 server
arbiter 2001:258:8404:1217:250:56ff:fea8:55a0 27541 arbiter
```

When operation other than automatic degradation by the arbitration server is selected

- Specify definitions related to the admin network.
- Define the same content on the primary server and standby server.
- Define lines for database servers only.
- Specify only one IP address or host name and port number.

IP address or host name		Port number	
First	Second	First	Second
IP address or host name to be used as the admin network	Not required	Port number used as the admin network	Not required

Example)

The literal space represents a space.

IPv4

```
server1 192.0.2.100 27540
server2 192.0.2.110 27540
```

IPv6

```
server1 2001:258:8404:1217:250:56ff:fea7:559f 27540
server2 2001:258:8404:1217:250:56ff:fea7:55a0 27540
```

Content to be defined on the arbitration server

This section explains the network.conf content to be defined on the arbitration server.

- Specify definitions related to the arbitration network.
- Define lines for database servers only.
- For the IP address or host name, specify the same value as the second IP address or host name specified in the database server line in network.conf of the database server.
- For the port number, specify the same value as the second port number specified in the database server line in network.conf of the database server.

Example)

The literal space represents a space.

IPv4

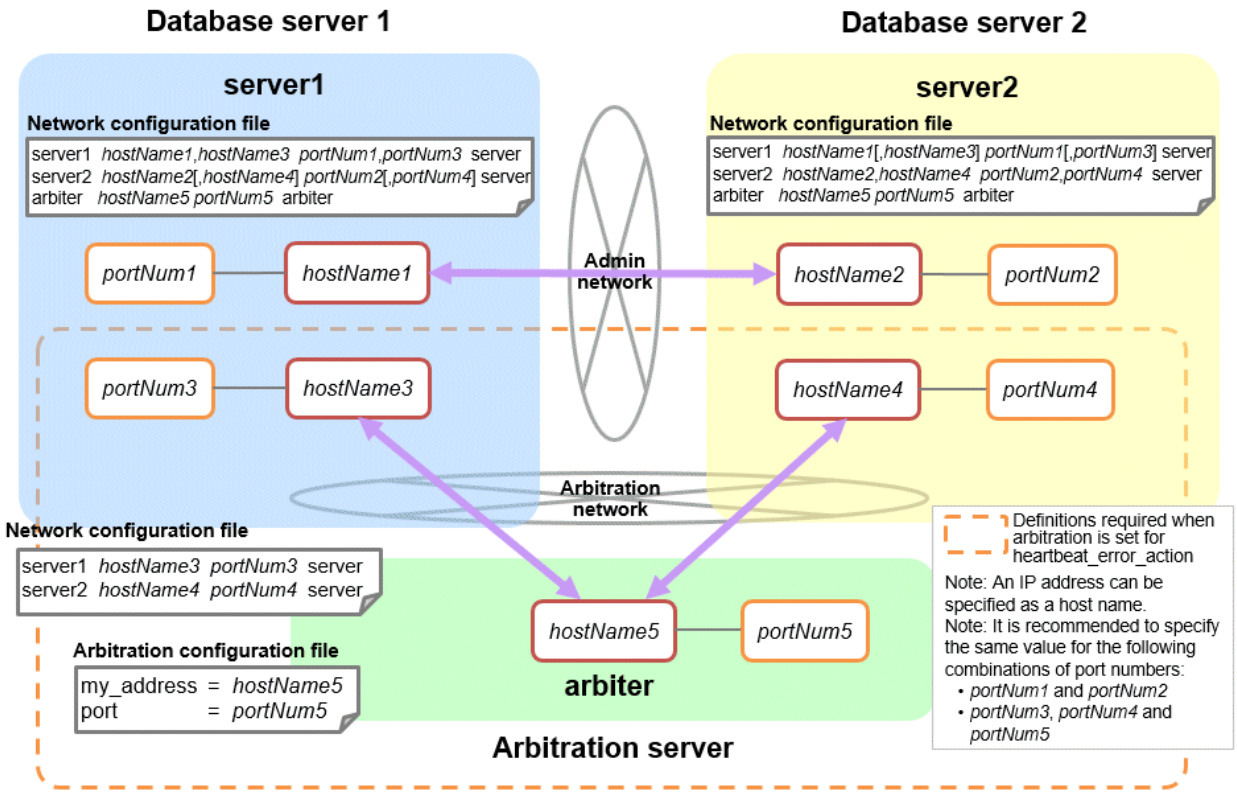
```
server1 192.0.3.100 27541
server2 192.0.3.110 27541
```

IPv6

```
server1 2001:258:8404:1217:250:56ff:fea8:559f 27541
server2 2001:258:8404:1217:250:56ff:fea8:55a0 27541
```

Relationship between network-related definitions

Refer to the diagram below for the relationship between the host names and IP addresses or port numbers specified in the network configuration file (network.conf) and arbitration configuration file (arbitration.conf).



A.4 Server Configuration File

A.4.1 Server Configuration File for the Database Servers

Define the information related to Mirroring Controller monitoring and control in the serverIdentifier.conf file. The maximum length of the server identifier is 64 bytes. Use ASCII characters excluding spaces to specify this parameter.

If the primary server and standby server environments are different, define content that is different, according to the environment.

Table A.4 serverIdentifier.conf file

Parameter	Value set	Explanation
db_instance	'*dataStorageDestinationDir*' [Example] db_instance = '/database1/inst1'	Specify using single quotation marks (') to enclose the data storage destination directory used to identify the monitoring target instance. Use ASCII characters to specify this parameter.
target_db	postgres or template1	Specify the name of the database to be connected to the database instance. The default is "postgres".
db_instance_username	'*usernameToConnectToDbInstance*'	Specify the username to connect to the database instance. Use ASCII characters to specify this parameter. Specify this parameter if the database administrator user is different from the operating system user who starts Mirroring Controller. Enclose the username of the database superuser in single quotation marks ('). The maximum length of the username is 63 bytes.

Parameter	Value set	Explanation
		The default is the operating system user who starts Mirroring Controller.
db_instance_password	'*passwordOfInstanceAdminUser*'	Specify the password used when Mirroring Controller connects to a database instance, enclosed in single quotation marks ('). Use ASCII characters to specify this parameter. If password authentication is performed, you must specify this parameter in the settings used when Mirroring Controller connects to a database instance. If you specify this parameter when password authentication is not performed, the parameter will be ignored. If the specified value of this parameter includes ' or \, write \' or \\, respectively.
enable_hash_in_password	on or off	Specify on to treat the # in the db_instance_password specification as a password character, or off to treat it as a comment. The default is "off".
core_file_path	*'coreFileOutputDir'*	Specify the directory to which the core file is to be output, enclosed in single quotation marks ('). Use ASCII characters to specify this parameter. If this parameter is omitted, it will be assumed that the Mirroring Controller management directory was specified.
syslog_facility	Specify LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, or LOCAL7.	When the import of logs to the syslog is enabled, the value of this parameter will be used for "facility" of the syslog. The default is "LOCAL0".
syslog_ident (*1)	'*programName*'	Specify using single quotation marks (') to enclose the program name used to identify the Mirroring Controller message in the system log. Use ASCII characters excluding spaces to specify this parameter. The default is 'MirroringControllerOpen'.
remote_call_timeout	Admin communication timeout	Specify the timeout value (milliseconds) of the Mirroring Controller agent process for communication between servers. Specify a value between 0 and 2147483647 to be less than the operation system TCP connection timeout (*2). In addition, when using the Mirroring Controller arbitrage process, fencing commands, and state transition commands, specify a value that is greater than the sum of the timeout values (*3). The value 0 indicates that there is no timeout limit. The default is 70000 milliseconds (70 seconds).
agent_alive_timeout	Timeout for Mirroring Controller process heartbeat monitoring (seconds)	If there is no response for at least the number of seconds specified, the Mirroring Controller process is restarted.

Parameter	Value set	Explanation
		Specify 0 or a value between 2 and 2147483647. The value 0 indicates that there is no timeout limit. The default is 0 seconds.
heartbeat_error_action	Operation when a heartbeat abnormality is detected using operating system or server heartbeat monitoring	arbitration: Perform automatic degradation using the arbitration server. command: Call a user command to determine degradation, and perform automatic degradation if required. message: Notify messages. fallback: Perform automatic degradation unconditionally. The default is "arbitration". Set the same value on the primary server and standby server.
heartbeat_interval	Interval time for abnormality monitoring during heartbeat monitoring of the operating system or server (milliseconds)	Abnormality monitoring of the operating system or server is performed at the interval specified in heartbeat_interval. If an error is detected, operation will conform to the value specified for heartbeat_error_action. If "arbitration" is specified in heartbeat_error_action, the error detection time during monitoring of the operating system or server becomes longer than when the arbitration server is not used, by up to the value specified for arbitration_timeout. Specify a value between 1 and 2147483647. The specified value is used as the default for db_instance_check_interval and disk_check_interval. The default is 800 milliseconds.
heartbeat_timeout	Timeout for abnormality monitoring during heartbeat monitoring of the operating system or server (seconds)	If there is no response for at least the number of seconds specified, it will be assumed that an error has occurred that requires the primary server to be switched, or the standby server to be disconnected. If an error is detected, operation will conform to the value specified for heartbeat_error_action. If "arbitration" is specified in heartbeat_error_action, the error detection time during monitoring of the operating system or server becomes longer than when the arbitration server is not used, by up to the value specified for arbitration_timeout. Specify a value between 1 and 2147483647. The specified value is used as the default for db_instance_check_timeout. The default is 1 second.
heartbeat_retry	Number of retries for abnormality monitoring during heartbeat monitoring of the operating system or server (number of times)	Specify the number of retries to be performed when an error has been detected that requires the primary server to be switched, or the standby server to be disconnected. If an error is detected in succession more than the specified number of times, switch or disconnection will be performed. If an error is detected, operation will conform to the value specified for heartbeat_error_action. If "arbitration" is specified in heartbeat_error_action, the error detection time during monitoring of the operating system or server

Parameter	Value set	Explanation
		becomes longer than when the arbitration server is not used, by up to the value specified for arbitration_timeout. Specify a value between 0 and 2147483647. The specified value is used as the default for db_instance_check_retry and disk_check_retry. The default is 2 times.
db_instance_check_interval	Database process heartbeat monitoring interval (milliseconds)	Heartbeat monitoring of the database process is performed at the interval specified in db_instance_check_interval. This parameter setting is also used for abnormality monitoring of streaming replication. Specify a value between 1 and 2147483647. The default is the value set for heartbeat_interval.
db_instance_check_timeout	Database process heartbeat monitoring timeout (seconds)	If there is no response for at least the number of seconds specified, it will be assumed that an error has occurred that requires the primary server to be switched, or the standby server to be disconnected. Specify a value between 1 and 2147483647. The default is the value set for heartbeat_timeout.
db_instance_check_retry	Number of retries for database process heartbeat monitoring (number of times)	Specify the number of retries to be performed when an error has been detected that requires the primary server to be switched, or the standby server to be disconnected. If an error is detected in succession more than the specified number of times, switch or disconnection will be performed. However, if it detects that the database process is down, it will immediately switch or disconnect regardless of the setting of this parameter. This parameter setting is also used for abnormality monitoring of streaming replication. Specify a value between 0 and 2147483647. The default number of retries is the value set for heartbeat_retry.
db_instance_timeout_action	none, message, or failover	Specify the behavior for no-response monitoring of the instance. none: Do not perform no-response monitoring. message: Notify messages if an error is detected during no-response monitoring. failover: Perform automatic degradation if an error is detected during no-response monitoring. The default is "failover".
disk_check_interval	Interval time for disk abnormality monitoring (milliseconds)	Abnormality monitoring of disk failure is performed at the interval specified in disk_check_interval. If the file cannot be created, it will be assumed that an error has occurred that requires the primary server to be switched, or the standby server to be disconnected. Specify a value between 1 and 2147483647. Set a value larger than the disk access time.

Parameter	Value set	Explanation
		The default is the value set for heartbeat_interval.
disk_check_retry	Number of retries for disk abnormality monitoring (number of times)	Specify the number of retries to be performed when an error has been detected that requires the primary server to be switched, or the standby server to be disconnected. If an error is detected in succession more than the specified number of times, switch or disconnection will be performed. Specify a value between 0 and 2147483647. The default number of retries is the value set for heartbeat_retry.
disk_check_timeout	Abnormality monitoring timeout time (seconds)	The time allowed from the start time of the next disk_check_interval after a disk error occurs until the error is determined to be due to timeout. To disconnect the standby server when a disk error due to this timeout is detected on the standby server, set shutdown_detached_synchronous_standby to on. The default is 2147483. Specify an integer between 0 and 2147483.
disk_check_max_threads	Upper limit on the number of threads used for abnormality monitoring	Upper limit on the number of threads for disk monitoring. The default is the number of processors available to the JVM) Specify an integer between 1 and 2147483647, but setting a value greater than the threads available on the machine may result in a system error. When you run the mc_ctl status command separately from the monitoring process, each mc_ctl status temporarily uses the same number of threads as the monitoring process. When setting disk_check_max_threads, consider the machine's thread limit, the number of table spaces you plan to use, and the number of mc_ctl status commands that may be executed at the same time.
tablespace_directory_error_action	message or failover	Specify the behavior to be implemented if an error is detected in the tablespace storage directory. message: Notify messages. failover: Perform automatic degradation. The default is "failover".
arbiter_alive_interval	Interval time for monitoring connection to the Mirroring Controller arbitration process (milliseconds)	A heartbeat is sent to the Mirroring Controller arbitration process at the specified interval. Specify a value between 1 and 2147483647. The default is 16000 milliseconds. This parameter does not need to be set for operation that does not use the arbitration server.
arbiter_alive_timeout	Timeout for monitoring connection to the Mirroring Controller arbitration process (seconds)	If the heartbeat does not respond within the specified number of seconds, the Mirroring Controller arbitration process is determined to have been disconnected, a message is output, and reconnection is attempted.

Parameter	Value set	Explanation
		Specify a value between 1 and 2147483647. The default is 20 seconds. This parameter does not need to be set for operation that does not use the arbitration server.
arbiter_alive_retry	Number of retries for monitoring connection to the Mirroring Controller arbitration process (number of times)	Specify the number of heartbeat retries to be performed if an error is detected in the heartbeat to the Mirroring Controller arbitration process. If the heartbeat does not respond within the specified number of retries, the Mirroring Controller arbitration process is determined to have been disconnected. Specify a value between 0 and 2147483647. The default is 0 times. This parameter does not need to be set for operation that does not use the arbitration server.
arbiter_connect_interval	Attempt interval for connection to the Mirroring Controller arbitration process (milliseconds)	Reconnection is attempted at the specified interval if connection fails at startup of the Mirroring Controller process or if the Mirroring Controller arbitration process is disconnected. Specify a value between 1 and 2147483647. The default is 16000 milliseconds. This parameter does not need to be set for operation that does not use the arbitration server.
arbiter_connect_timeout	Timeout for connection to the Mirroring Controller arbitration process (seconds)	If reconnection at startup of the Mirroring Controller process or after disconnection of the Mirroring Controller arbitration process does not succeed within the specified number of seconds, connection to the Mirroring Controller arbitration process is determined to have failed and reconnection is attempted. Specify a value between 1 and 2147483647. The default is 20 seconds. This parameter does not need to be set for operation that does not use the arbitration server.
fencing_command	'*fencingCmdFilePath*' [Setting example] fencing_command = '/mc/fencing_dir/execute_fencing.sh'	Specify the full path of the fencing command that fences a database server where an error is determined to have occurred. Enclose the path in single quotation marks ('). Specify the path using less than 1024 bytes. This parameter must be specified when "command" is set for heartbeat_error_action.
fencing_command_timeout	Fencing command timeout (seconds)	If the command does not respond within the specified number of seconds, fencing is determined to have failed and a signal (SIGTERM) is sent to the fencing command execution process. Specify a value between 1 and 2147483647. The default is 20 seconds.

Parameter	Value set	Explanation
arbitration_timeout	Arbitration processing timeout in the Mirroring Controller arbitration process (seconds)	The specified value must be at least equal to the value of fencing_command_timeout in the arbitration configuration file, which is the heartbeat monitoring time of the operating system or server. If there is no response for at least the number of seconds specified, the primary server will not be switched and the standby server will not be disconnected. Therefore, perform degradation manually. If the heartbeat_interval, heartbeat_timeout, and heartbeat_retry values are specified in arbitration.conf for the arbitration server, use the arbitration server values to design arbitration_timeout. Specify a value between 1 and 2147483647. The default is 30 seconds. This parameter does not need to be set for operation that does not use the arbitration server.
arbitration_command	'*arbitrationCmdFilePath*' [Setting example] arbitration_command = '/mc/arbitration_dir/execute_arbitration_command.sh'	Specify the full path of the arbitration command to be executed when an abnormality is detected during heartbeat monitoring of the operating system or server. Enclose the path in single quotation marks ('). Specify the path using less than 1024 bytes. This parameter must be specified when "command" is set for heartbeat_error_action.
arbitration_command_timeout	Arbitration command timeout (seconds)	If the arbitration command does not respond within the specified number of seconds, it is determined that execution of the arbitration command has failed and a signal (SIGTERM) is sent to the arbitration command execution process. Specify a value between 1 and 2147483647. The default is 30 seconds. This parameter can be specified only when "command" is set for heartbeat_error_action.
shutdown_detached_synchronous_standby	on or off	Specify whether to forcibly stop the instance on the standby server when the standby server is disconnected. on: Stop the instance. off: Do not stop the instance. If "on" is specified and the pre-detach command was created, the pre-detach command is executed and then the instance is stopped. The default is "off".
post_switch_command	'*postSwitchCmdFilePath*' [Setting example] post_switch_command = '/mc/status_change/execute_post_switch.sh'	Specify the full path of the command to be called by Mirroring Controller after a new primary server is promoted during a failover of the primary server. Enclose the path in single quotation marks ('). Specify the path using less than 1024 bytes.

Parameter	Value set	Explanation
post_attach_command	'*postAttachCmdFilePath*' [Setting example] post_attach_command = '/mc/status_change/execute_post_attach.sh'	Specify the full path of the command to be called by Mirroring Controller after the standby server is attached to the cluster system. Enclose the path in single quotation marks ('). Specify the path using less than 1024 bytes.
pre_detach_command	'*preDetachCmdFilePath*' [Setting example] pre_detach_command = '/mc/status_change/execute_pre_detach.sh'	Specify the full path of the command to be called by Mirroring Controller before the standby server is disconnected from the cluster system. Enclose the path in single quotation marks ('). Specify the path using less than 1024 bytes.
status_change_command_timeout	State transition command timeout (seconds)	Specify the timeout value of the post-switch command, post-attach command, and pre-detach command. If the command does not respond within the specified number of seconds, a signal (SIGTERM) is sent to the execution process of the status change command. Specify a timeout between 1 and 2147483647. The default is 20 seconds.
enable_promote_on_os_and_admin_network_error	on or off	If on is specified, if a admin network error occurs and replication is further lost, the system will ask the arbitration server to verify that the primary server and databases are running, and if they are down, promote the standby server. The default is "off".
check_synchronous_standby_names_validation	on or off	Specify whether Mirroring Controller is to periodically check during operations whether the synchronous_standby_names parameter in postgresql.conf was changed by an incorrect user operation. However, it is not recommended to enable this parameter, because performing this check causes Mirroring Controller to use the CPU of the database server redundantly and execute SQL statements at high frequency. The default is "off".
db_instance_ext_pq_conninfo	*'libpqConnectionSSLParmToConnectToDbinstance'*	Specify, in key-value form, the connection parameter for libpq that Mirroring Controller adds when connecting to a database. The connection parameters you can specify are those related to SSL. Use ASCII characters to specify this parameter. If you want to validate the server certificate using the destination host name, such as specifying sslmode=verify-full as the connection parameter, specify the host name in the Common Name of the server certificate in the sslservercertcn connection parameter. For information about the sslservercertcn connection parameter, refer to "Using the C Library (libpq)" in "Application Connection Switch Feature" in the Application Development Guide. The connection parameter specified in this parameter must also be specified in the db_instance_ext_jdbc_conninfo.
db_instance_ext_jdbc_conninfo	*'JDBCConnectionSSLParmToConnectToDbinstance'*	Specify, in URI form, the connection parameter for JDBC that Mirroring Controller adds when connecting to a database. The connection parameters you can specify are

Parameter	Value set	Explanation
		those related to SSL. Use ASCII characters to specify this parameter. If you want to validate the server certificate using the destination host name, such as specifying sslmode=verify-full as the connection parameter, specify the host name in the Common Name of the server certificate in the sslservercertcn connection parameter. For information about the sslservercertcn connection parameter, refer to "Using the JDBC Driver" in "Application Connection Switch Feature" in the Application Development Guide. The connection parameter specified in this parameter must also be specified in the db_instance_ext_pq_conninfo.

*1: By specifying the syslog_ident parameter of the postgresql.conf file, the Mirroring Controller output content can be referenced transparently, so log reference is easy.

*2: The operating system TCP connection timeout period is determined by the kernel parameter tcp_syn_retries. The remote_call_timeout parameter must be set to a value that is shorter than the timeout period for the operating system TCP connection timeout, so change either parameter as necessary.

*3: In management communications, arbitrage processing, fencing commands, and state transition commands may be executed in succession by the Mirroring Controller arbitrage process. Therefore, the value specified for the remote_call_timeout parameter must be greater than the sum of these timeout values. Depending on the value specified for the heartbeat_error_action parameter, set the remote_call_timeout parameter using the following formula:

- arbitration: (arbitration_timeout + fencing_command_timeout + status_change_command_timeout) * 1000
- command: (fencing_command_timeout + status_change_command_timeout) * 1000
- message: (fencing_command_timeout + status_change_command_timeout) * 1000
- fallback: (status_change_command_timeout) * 1000

Because other internal processing may be performed in the management communication, set the value obtained by multiplying the calculation result of the above equation by the safety factor (about 1.2).

Also, for the fencing_command_timeout parameter, use the value of the parameter in the database server's server definition file, not in the arbitration definition file.

The availability of some parameters depends on the value set for the heartbeat_error_action parameter that sets the operation to be performed if heartbeat monitoring of the operating system or server detects a heartbeat abnormality.

Table A.5 Parameter availability depending on the value set for the heartbeat_error_action parameter

Parameter	Value set			
	arbitration	command	message	fallback
arbiter_alive_interval	Y	N	N	N
arbiter_alive_timeout	Y	N	N	N
arbiter_alive_retry	Y	N	N	N
arbiter_connect_interval	Y	N	N	N
arbiter_connect_timeout	Y	N	N	N
arbitration_timeout	Y	N	N	N
arbitration_command	N	R	N	N
arbitration_command_timeout	N	Y	N	N
fencing_command	Y	R	Y	N
fencing_command_timeout	Y	Y	Y	N
shutdown_detached_synchronous_standby	Y	Y	N	N

R: Required

Y: Can be specified

N: Cannot be specified

A.4.2 Arbitration Configuration File

In arbitration.conf, define the information related to arbitration and control of the Mirroring Controller arbitration process.

Table A.6 arbitration.conf file

Parameter	Value set	Description
port	Port number of the Mirroring Controller arbitration process	The specified value must not exceed the range 0 to 65535. Ensure that the port number does not conflict with other software. Do not specify an ephemeral port that may temporarily be assigned by another program. For the port number of the arbitration server to be specified in network.conf on the database server, specify the same value as the port number specified in this parameter.
my_address	'*ipAddrOrHostNameThatAcceptsConnectionFromMirroringControllerProcessOnDbServer*' [Setting example] my_address = '192.0.3.120'	For the IP address or host name of the arbitration server to be specified in network.conf on the database server, specify the same value as the IP address or host name specified in this parameter. IPv4 and IPv6 addresses can be specified. Specify the IP address or host name, enclosed in single quotation marks (').
core_file_path	'*coreFileOutputDir*'	Specify the directory to which the core file is to be output, enclosed in single quotation marks ('). Use ASCII characters to specify this parameter. If this parameter is omitted, it will be assumed that the Mirroring Controller arbitration process management directory was specified.
syslog_facility	Specify LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, or LOCAL7.	When the import of logs to the syslog is enabled, the value of this parameter will be used for "facility" of the syslog. The default is "LOCAL0".
syslog_ident	'*programName*'	Specify using single quotation marks (') to enclose the program name used to identify the Mirroring Controller arbitration process message in the system log. Use ASCII characters excluding spaces to specify this parameter. The default is 'MirroringControllerArbiter'.
fencing_command	'*fencingCmdFilePath*' [Setting example] fencing_command = '/arbiter/fencing_dir/execute_fencing.sh'	Specify the full path of the fencing command that fences a database server where an error is determined to have occurred. Enclose the path in single quotation marks ('). Specify the path using less than 1024 bytes.
fencing_command_timeout	Fencing command timeout (seconds)	If the command does not respond within the specified number of seconds, fencing is

Parameter	Value set	Description
		determined to have failed and a signal (SIGTERM) is sent to the fencing command execution process. Specify a value between 1 and 2147483647. The default is 20 seconds.
heartbeat_interval(*1)	Interval time for heartbeat monitoring of the operating system or server (milliseconds)	The heartbeat monitoring of the database server is checked at the specified interval and arbitration is performed. Specify a value between 1 and 2147483647. The default is the value specified in *serverIdentifier*.conf of the database server. Specify this parameter to perform optimization taking into account differences in the line load to the admin network and the reduction in the time it takes to degrade.
heartbeat_timeout	Timeout for heartbeat monitoring of the operating system or server (seconds)	If there is no response for at least the number of seconds specified, it will be assumed that an error has occurred that requires the primary server or standby server to be fenced. Specify a value between 1 and 2147483647. The default is the value specified in *serverIdentifier*.conf of the database server. Specify this parameter to perform optimization taking into account differences in the line load to the admin network and the reduction in the time it takes to degrade.
heartbeat_retry	Number of retries for heartbeat monitoring of the operating system or server (number of times)	Specify the number of retries to be performed when an error has been detected that requires the primary server or standby server to be fenced. If an error is detected in succession more than the specified number of times, fencing will be performed. Specify a value between 0 and 2147483647. The default is the value specified in *serverIdentifier*.conf of the database server. Specify this parameter to perform optimization taking into account differences in the line load to the admin network and the reduction in the time it takes to degrade.

*1:Refer to "2.11.4 Tuning for Optimization of Degradation Using Abnormality Monitoring" for information on the tuning parameters for operating system or server abnormality monitoring when using an arbitration server.

Appendix B Supplementary Information on Building the Primary Server and Standby Server on the Same Server

The primary server and standby server can be pseudo-configured on the same server for system testing, for example. Out of consideration for performance and reliability, do not use this type of configuration for any other purposes. For this reason, do not use this type of configuration in a production environment.

Note that the setup and operations is the same as if the primary and standby servers are built on different servers.

This appendix provides supplementary information explaining how to configure the primary server and standby server on the same server.

Note

Even if automatic degradation by an arbitration server is set when the primary server and standby server are configured on the same server, there will be no effect of it.

B.1 Backup Data Storage Destination Directory

It is not a problem if the same backup data storage destination directory is used on the primary server and standby server.

B.2 How to Execute the mc_ctl Command

When executing the mc_ctl command, specify the server identifier in the --local-server option in order to identify the operation destination server.

Below is an example of starting Mirroring Controller of the server "server1" defined in the network.conf file. For mc_ctl command operations using another mode, also specify the --local-server option.

Define two server identifiers for the same IP address with different port numbers in the network.conf file.

```
Example)
```

```
server1 192.0.2.100 27540
server2 192.0.2.100 27541
```

Ensure that the port numbers of both primary server and standby server do not conflict with any other software.

Enabling automatic switch/disconnection

Start Mirroring Controller of the server "server1":

```
Example)
```

```
$ mc_ctl start -M /mcdir/inst1 --local-server server1
```

Stop Mirroring Controller of the server "server1":

```
Example)
```

```
$ mc_ctl stop -M /mcdir/inst1 --local-server server1
```

Disabling automatic switch/disconnection

Start Mirroring Controller of the server "server1":

```
Example)
```

```
$ mc_ctl start -M /mcdir/inst1 -F --local-server server1
```

Stop Mirroring Controller of the server "server1":

Example)

```
$ mc_ctl stop -M /mcdir/inst1 --local-server server1
```

Note

Add the --local-server option to the mc_ctl option specification for ExecStart and ExecStop of the unit file for systemd.

Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" for details.

Appendix C User Commands

This appendix describes three categories of commands:

- Fencing command
- Arbitration command
- State transition commands

This appendix describes each category of user command.

C.1 Fencing Command

Format

The syntax for calling the fencing command from the Mirroring Controller process or the Mirroring Controller arbitration process is described below.

Fencing command of the database server

```
fencingCmd executionMode mcDegradationOper cmdServerId targetServerId primarycenter
```

Fencing command of the arbitration server

```
fencingCmd executionMode mcDegradationOper targetServerId
```

Input

Fencing command of the database server

Execution mode

monitor: Detect issues via automatic monitoring of the Mirroring Controller process

command: Mirroring Controller command execution (switch mode or detach mode of the mc_ctl command)

Degradation operation to be performed by Mirroring Controller

switch: Switch

detach: Disconnect

cmdServerId

ID of the database server that called the command

targetServerId

ID of the database server to be fenced

primarycenter

Fixed value

Fencing command of the arbitration server

Execution mode

monitor: Detect issues via automatic monitoring of the Mirroring Controller process

command: Mirroring Controller command execution (switch mode or detach mode of the mc_ctl command)

Degradation operation to be performed by Mirroring Controller

switch: Switch

detach: Disconnect

targetServerId

ID of the database server to be fenced

Output

Return value

0: Mirroring Controller will continue the degradation process.

Other than 0: Mirroring Controller will cancel the degradation process.

Description

Identifies the database server targeted for fencing based on the input server identifier, and implements the process that isolates it from the cluster system.

Notes

- The command is executed by the operating system user who started Mirroring Controller or the Mirroring Controller arbitration process. Therefore, if the command is to be executed by a specific operating system user, change the executing user of the command accordingly.
- The operating system user who started Mirroring Controller or the Mirroring Controller arbitration process must have execution privileges to the command. Otherwise, the degradation process will be canceled.
- From a security point of view, set the access privileges as necessary so that the fencing command is not overwritten and unauthorized operations are not performed by unintended operating system users.
- If the fencing command returns a value other than 0, Mirroring Controller will cancel the degradation process, so it is necessary for the user to check the status of the server, and switch or disconnect it manually.
- Before executing the fencing command, check if the server is already fenced, to avoid the command terminating abnormally.
- If the command times out, Mirroring Controller will stop the command, output an error message, and cancel the degradation process.

Information

The fencing command can be implemented by simply stopping the operating system or server. For example, if stopping the power for the database server, it is possible to use a utility to control the hardware control board in environments equipped with boards compatible with IPMI hardware standard.

Below is a sample script of a fencing command that powers off the database server using the IPMI tool.

Sample shell script

```
/installDir/share/mcarb_execute_fencing.sh.sample
```

C.2 Arbitration Command

Format

The syntax for calling the arbitration command from the Mirroring Controller process is described below.

```
arbitrationCmd cmdServerId targetServerId primarycenter
```

Input

cmdServerId

ID of the database server that called the command

targetServerId

ID of the database server to arbitrate

primarycenter

Fixed value

Output

Return value

0: The database server to arbitrate has an issue, and Mirroring Controller will continue the degradation process.

Other than 0: The database server to arbitrate is normal, and Mirroring Controller will cancel the degradation process.

Description

Identifies the database server to arbitrate based on the input server identifier, and checks the status of the server.

Notes

- The command is executed by the operating system user who started Mirroring Controller.
- The operating system user who started Mirroring Controller must have execution privileges to the command. Otherwise, the command will not be called, and the degradation process will be canceled.
- If the command times out, Mirroring Controller will stop the command, output an error message, and cancel the degradation process.

C.3 State Transition Commands

State transition commands include the three types of user commands below. Any of the commands can be implemented by Mirroring Controller in conjunction with database server status transitions.

- Post-switch command
- Pre-detach command
- Post-attach command

C.3.1 Post-switch Command

Format

The syntax for calling the post-switch command from the Mirroring Controller process is described below.

```
postswitchCmd serverIdentifier primarycenter
```

Input

serverIdentifier

ID of the database server (new primary server) that was switched

primarycenter

Fixed value

Output

Return value

None

Notes

- The command is executed by the operating system user who started Mirroring Controller.
- The operating system user who started Mirroring Controller must have execution privileges to the command. Otherwise, the command will not be called.
- If the command times out, Mirroring Controller will stop the command, output an error message, and cancel the process.

C.3.2 Pre-detach Command

Format

The syntax for calling the pre-detach command from the Mirroring Controller process is described below.

```
predetachCmd cmdServerId serverRole targetServerId primarycenter
```

Input

cmdServerId

ID of the database server that called the command

Server role

Role of the database server that called the command

primary: Primary

standby: Standby

targetServerId

ID of the standby server to be disconnected from the cluster system

primarycenter

Fixed value

Output

Return value

None

Notes

- The command is executed by the operating system user who started Mirroring Controller.
- The operating system user who started Mirroring Controller must have execution privileges to the command. Otherwise, the command will not be called, however, Mirroring Controller will output an error message and continue the process.
- If the command times out, Mirroring Controller will stop the command, output an error message, and cancel the process.

C.3.3 Post-attach Command

Format

The syntax for calling the post-attach command from the Mirroring Controller process is described below.

```
postattachCmd cmdServerId serverRole targetServerId primarycenter
```

Input

cmdServerId

ID of the database server that called the command

Server role

Role of the database server that called the command

primary: Primary

standby: Standby

targetServerId

ID of the standby server to be attached to the cluster system

primarycenter

Fixed value

Output

Return value

None

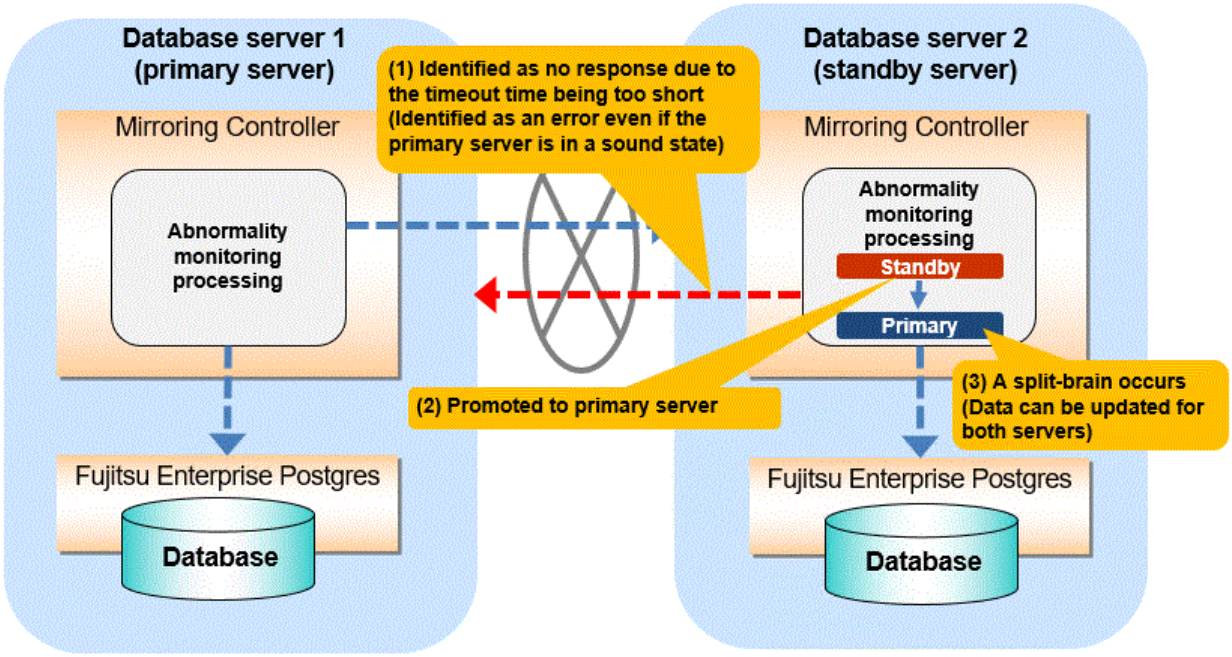
Notes

- The command is executed by the operating system user who started Mirroring Controller.
- The operating system user who started Mirroring Controller must have execution privileges to the command. Otherwise, the command will not be called.
- If the command times out, Mirroring Controller will stop the command, output an error message, and cancel the process.

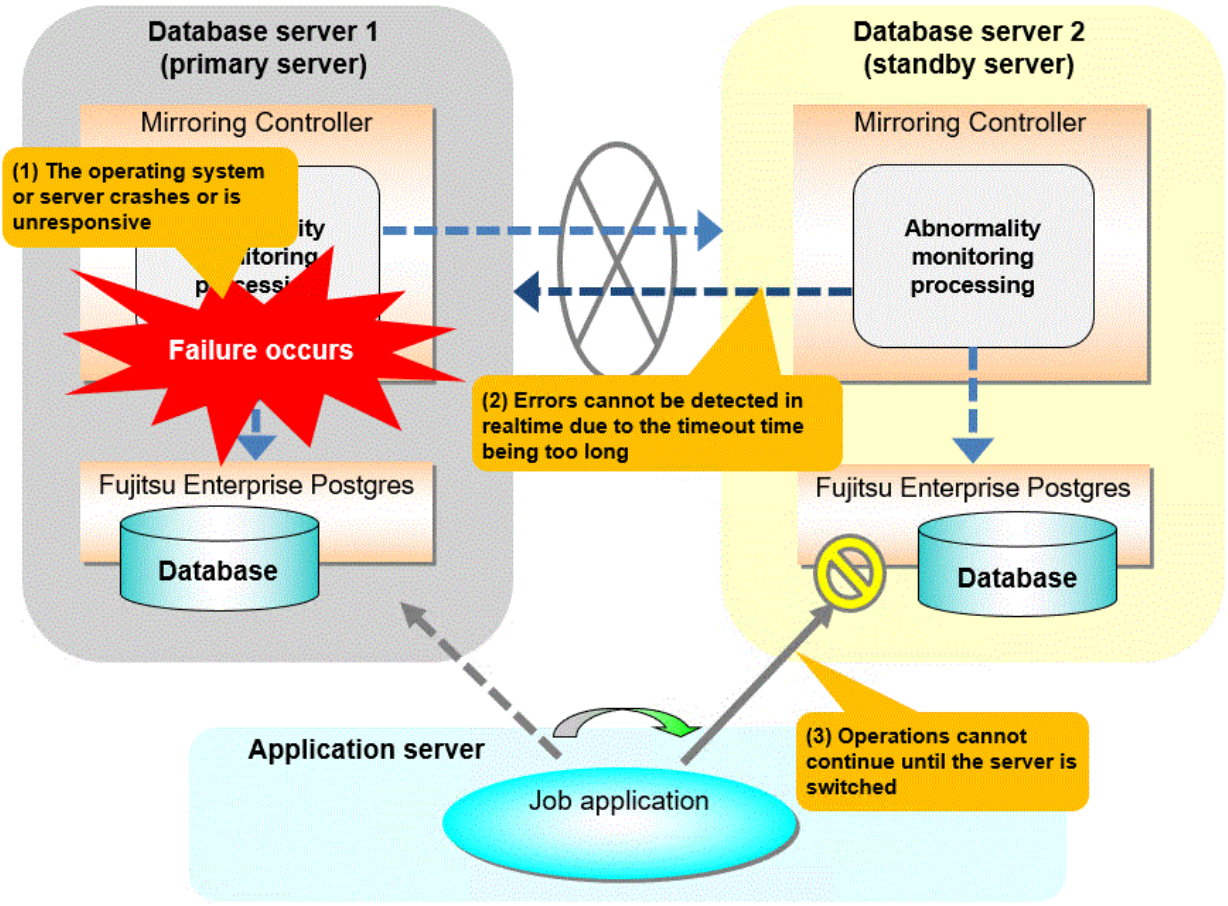
Appendix D Notes on Performing Automatic Degradation Immediately after a Heartbeat Abnormality

The type of issue below occurs if automatic degradation is performed unconditionally after an issue is detected during heartbeat monitoring of an operating system or server, and heartbeat monitoring was not properly tuned.

- **If the timeout time is too short**



● If the timeout time is too long



Notes on monitoring when the operating system or server crashes or is unresponsive

As illustrated in the diagram above, timeout is used to monitor whether the operating system or server crashes or is unresponsive. Therefore, if tuning has not been performed correctly, there is a risk of a split-brain mistakenly occurring even if the server is in a sound state.

Split-brain is a phenomenon in which both servers temporarily operate as primary servers, causing data updates to be performed on both servers.

Split-brain detection method

It can be confirmed that split-brain occurs under the following conditions:

1. When the mc_ctl command is executed in status mode on both servers, the "host_role" of both servers is output as "primary", and
2. The following message is output to the system log of one of the servers:

```
promotion processing completed (MCA00062)
```

How to recover from a split-brain

Use the procedure described below. Note that the new primary server is the server that was confirmed in step 2 of the aforementioned detection method.

1. Stop all applications that are running on the old and new primary servers.
2. Investigate and recover the database.
   Investigate the update results that have not been reflected to the new primary server from the database of the old primary server, and apply to the new primary server as necessary.
3. Stop the old primary server instance and the Mirroring Controller.
4. Resume the applications that were stopped in step 1.

5. Recover the old primary server.
   While referring to "2.5 Setting Up the Standby Server", build (set up) the old primary server as the new standby server, from the new primary server.

Notes on monitoring by restarting the OS

The heartbeat monitoring of the database server uses the OS ping command. Therefore, if the timeout period is too long and the OS restarts, an error might not be detected. This can result in a business shutdown without automatic switchover even though the primary server is down. There are several ways to avoid this situation:

- Refer to "Parameters for the abnormality monitoring of the operating system or server in the server configuration file of the database server" and perform tuning so that the timeout time is shorter than the time required to restart the OS.
- Refer to "2.12 Setting Automatic Start and Stop of Mirroring Controller and Multiplexed Instances" to set automatic startup of Mirroring Controller.

Appendix E WebAdmin Disallow User Inputs Containing Hazardous Characters

WebAdmin considers the following as hazardous characters, which are not allowed in user inputs.

```
| (pipe sign)

& (ampersand sign)

; (semicolon sign)

$ (dollar sign)

% (percent sign)

@ (at sign)

' (single apostrophe)

" (quotation mark)

\' (backslash-escaped apostrophe)

\" (backslash-escaped quotation mark)

<> (triangular parenthesis)

() (parenthesis)

+ (plus sign)

CR (Carriage return, ASCII 0x0d)

LF (Line feed, ASCII 0x0a)

, (comma sign)

\ (backslash)
```

Appendix F Collecting Failure Investigation Data

If the cause of an error that occurs while building the environment or during operations is unclear, data must be collected for initial investigation.

This appendix describes how to collect data for initial investigation.

Use the pgx_fjqssinf command to collect data for initial investigation.

See

Refer to "pgx_fjqssinf" in the Reference for informations about the pgx_fjqssinf command.

Index

[S]

[T]

[U]

[W]