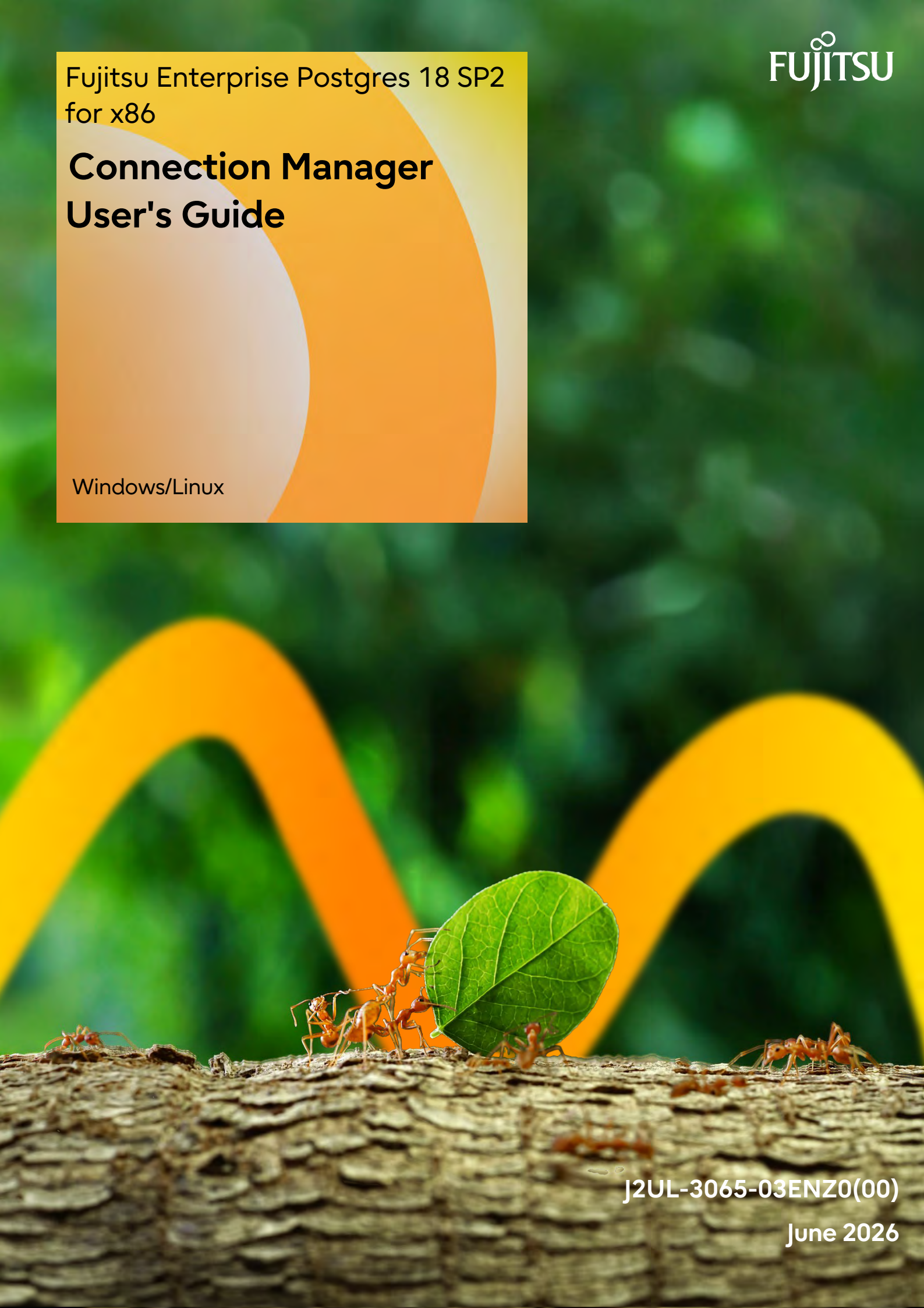
FUJITSU

Fujitsu Enterprise Postgres 18 SP2 for x86

Connection Manager User's Guide

Windows/Linux



J2UL-3065-03ENZ0(00)

June 2026

Preface

Purpose of this document

This document describes the Connection Manager features of Fujitsu Enterprise Postgres.

Intended readers

This document is aimed at people who use the Connection Manager features.

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

- Fujitsu Enterprise Postgres
- PostgreSQL
- Linux
- Windows
- Network

Structure of this document

This document is structured as follows:

Chapter 1 Connection Manager Features

Explains the features and Mechanisms of the Connection Manager.

Chapter 2 Setting Up

Explains setting up the Connection Manager.

Chapter 3 Using from an Application

Explains how to use the Connection Manager from an application.

Appendix A System Views

Explains the system view of Connection Manager.

Export restrictions

If this document is to be exported or provided overseas, confirm legal requirements for the Foreign Exchange and Foreign Trade Act as well as other laws and regulations, including U.S. Export Administration Regulations, and follow the required procedures.

Issue date and version

```
Edition 3.0: June 2026
Edition 2.0: March 2026
Edition 1.0: December 2025
```

Copyright

Contents

Chapter 1 Connection Manager Features

The Connection Manager provides the following features:

Heartbeat monitoring feature

Detects kernel panics between the server running the client and the server running the PostgreSQL instance(hereinafter referred to as instance), physical server failures, and inter-server network link downs, and notifies the client or instance. The client is notified as an error event through the SQL connection, and the instance will be notified in the form of a force collection of SQL connections with clients that are out of service.

Transparent connection support feature

When an application wants to connect to an instance of an attribute in a set of instances configured for replication, it can connect to that instance without being aware of which server it is running on.

Load balancing and read connection redistribution feature

With load balancing, if selection is possible, connections can be made to the server accepting the fewest connections. To use this feature, specify "leastconn" for the loadbalance parameter in conmgr.conf.

Using read connection redistribution, when a standby server is promoted, Connection Manager disconnects existing connections, allowing the promoted server to focus on the workload of the newly added primary server. To use this feature, specify "auto_disconnect" for the on_promote_action parameter in conmgr.conf. Only connections using Connection Manager are disconnected, and the disconnection occurs when a new query is executed.

Additionally, when the number of connections between standby servers becomes uneven, existing connections can be disconnected. This gives applications an opportunity to reconnect, thereby balancing the number of connections. For example, this can handle changes in the number of connections due to the addition or removal of standby servers, or the suspension of specific operations. To use this feature, set standby_reconnect_threshold parameter and standby_reconnect_ratio_target parameter. Only non-update connections using Connection Manager are disconnected, and the disconnection occurs when a new query is executed.

Load balancing and read connection redistribution feature is available in cluster systems using Patroni.

Initial data synchronization waiting feature for standby servers

You can exclude standby servers that are not synchronized with the primary server on a transaction-by-transaction basis from the connection destination candidates. This prevents applications from connecting to unsynchronized standby servers, ensuring data consistency. To use this feature, set the wait_streaming_replica_state parameter in conmgr.conf to "on".

Initial data synchronization waiting feature for standby servers is available in cluster systems using Patroni.

Information

The available client drivers for Connection Manager are libpq (C language library), ECPG (embedded SQL in C), ECOBPG (embedded SQL in COBOL), JDBC driver, ODBC driver, Python language package (psycopg) and Go language.

Each function is described below.

1.1 Heartbeat Monitoring Feature

Describes the Connection Manager's heartbeat monitoring feature.

Information

The Connection Manager does not monitor for delays, such as CPU busy occurring in the postmaster process or in the backend processes to which the application connects directly, or for no response, such as due to a software bugs. It also does not

monitor application downtime or unresponsiveness. To detect these, use various timeout features provided by PostgreSQL or the client drivers.

1.1.1 Difference from TCP keepalive

A peer of TCP connections cannot automatically detect a link down or server down.

There are two main methods to detect it. One is the operating system (Not all operating systems support it) TCP keepalive feature, and the other is the keepalive-equivalent timeout function implemented at the application layer. Connection Manager's heartbeat monitoring capabilities are categorized as the latter.

The operating system TCP keepalive feature has the following disadvantages, but the Connection Manager's heartbeat monitoring feature does not:

- The keepalive does not work when the TCP layer cannot receive an acknowledgement (ACK) and retransmits the packet repeatedly. This means that it is not possible to detect a down (For example, if a network goes down,) before sending some data and receiving ACK from the other side. There is also a parameter to interrupt retransmissions, which is not supported by some operating systems. The Connection Manager's heartbeat monitoring feature does not have this disadvantage because it is timeout monitoring at the application layer.
- The periodic packets for keepalive are sent per-TCP socket. If a instance accepts too many (For example, a few thousand clients) SQL connections, the load on the instance side cannot be ignored. The Connection Manager's heartbeat monitoring feature greatly reduces the load by allowing packets to be sent to the instance on a per-server basis on which the client runs.

1.1.2 Mechanism of Heartbeat Monitoring Feature

On the client side, the user must start one monitoring process using the cm_ctl command for the set of the instances to be monitored. This process, called the "conmgr process", can only be started by a user who is not an administrator (e.g. superuser(root) on Linux), or by a user who does not have Windows administrator privileges. An instance set is a collection of one or more instances that make up replication. One configuration file (conmgr.conf) for each conmgr process is used to set the information about the set of the instances being monitored and the parameters for monitoring.

On the server side, by installing PostgreSQL's EXTENSION that is called "watchdog", the postmaster will start two processes as background workers at instance startup.

One is the process for sending and receiving packets to and from the conmgr process for heartbeat monitoring. It is called "watchdog process". The other is the process for forcibly terminating SQL connections of the clients for which the watchdog process detects a failure on heartbeat monitorting. It is called "terminator process". SQL connections that do not use Connection Manager is also terminated, because the terminator process terminates them by IP address as key.

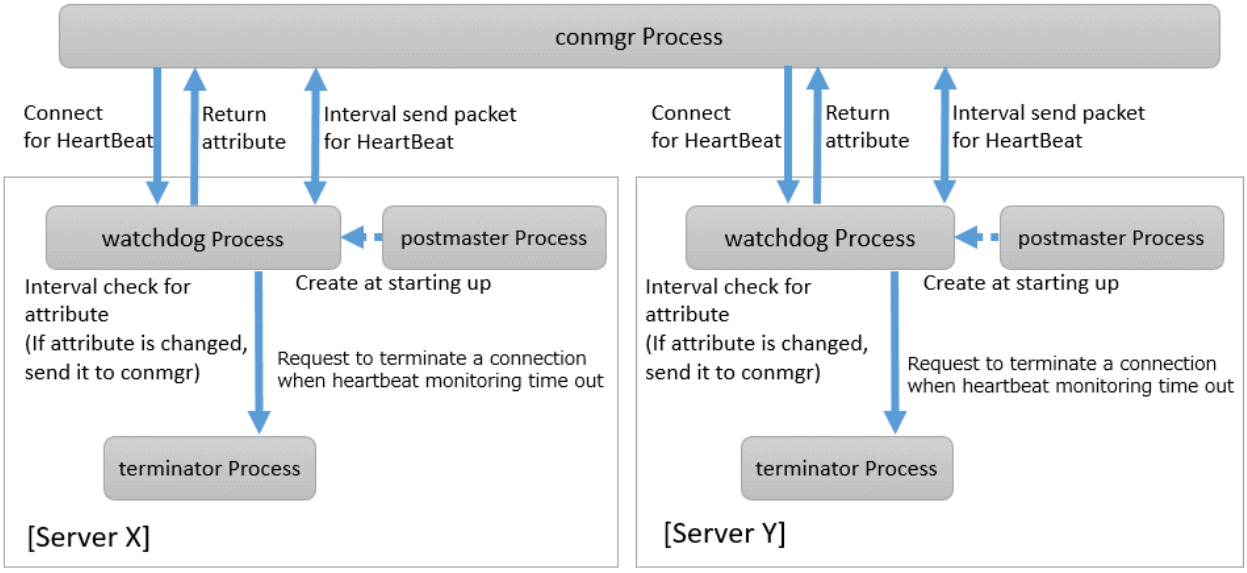
Information

Considerations for System Configuration

For replication, it is recommended that the instance that connects to the upstream instance of replication and the conmgr process that regards the upstream instance as an instance to be monitored for heartbeart (specified in backend_host parameter or backend_hostaddr parameter that is a configuration parameter of conmgr process) be not placed on the same server. This is because if the conmgr process stops normally or abnormally, the terminator process in the upstream instance will also kill the replication connection. The replication connection will reconnect automatically even if it is forcibly disconnected, so replication will continue without any problems. However, this can be a problem when the replication load is high or on systems that are sensitive to replication delays.

Note that the replication connection have different monitoring feature than the Connection Manager, so there is no need to monitor the Connection Manager for heartbeat. Refer to PostgreSQL documentation for details.

The process relationship is as follows:



See

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

1.2 Transparent Connection Support Feature

The features similar to Connection Manager's transparent connection support feature can be found in PostgreSQL's libpq and other client drivers.

Using libpq as an example, the connection parameter to use that feature is target_session_attrs parameter. If this parameter is used not through Connection Manager, libpq will attempt to find the required instance by connecting sequentially to all instances of the set of instance requested by the host parameter or hostaddr parameter. In the worst case, libpq may find the promoted primary at the connection to the last instance of instance set. This means that you cannot predict how long it will take to complete the switch.

However, when conmbined with the Connection Manager, the conmgr process obtains its attributes via the watchdog process from all servers in a set of servers in advance, so that the connections to that server can be initiated as soon as the application requests it.

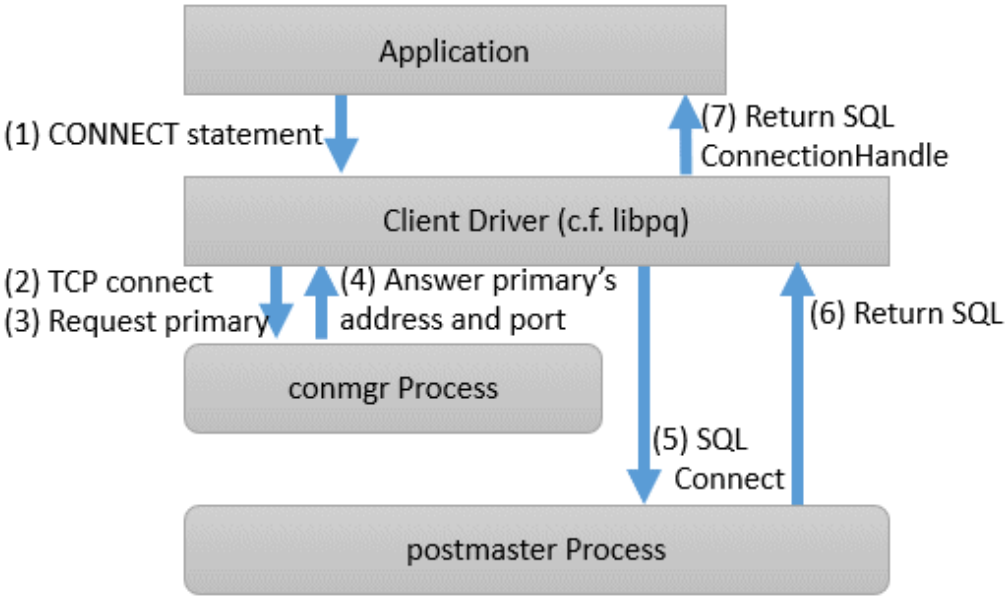
1.2.1 Mechanism of Connections using Transparent Connection Support Feature

A connection using this mechanism actually consists of two steps, but from the perspective of the application, it looks like a single SQL connection. In the application's connection string, specify the IP address or host name (In most cases it is "localhost") and port number where the conmgr process listens, and target_session_attrs parameter. You do not need to explicitly state that the connection is to the conmgr process. This is because the client driver can automatically determine whether the connection is to a instance or a conmgr process.

In the first phase of the connection, the client driver receives a connection request from the application and connects to the location specified in the connection string. Initially, it uses the protocol PostgreSQL requests, and if it learns in the middle that the connection is to a conmgr process, it asks the conmgr process for the IP address and port number that the instance with the attributes specified in the connection parameter target_session_attrs is listening for. If the destination is a backend process rather than a conmgr process, the connection process completes immediately and continues to send and receive data for normal SQL execution. The first stage of processing falls within the scope of timeout monitoring for SQL connection processing by each client driver. For example, the connection_timeout parameter of libpq.

In the second phase of the connection, the client driver connects to the instance using the IP address and port number from the conmgr process.Thereafter, the client driver and the instance directly send and receive the data for SQL execution. This ensures that the Connection Manager does not affect the performance of the SQL execution.

When the client driver is waiting to receive data after the second stage is completed, it monitors the reception of data to the two sockets obtained at each stage of the connection. This allows the client driver to know when, for example, the conmgr process notifies the client of a network link down.



Chapter 2 Setting Up

Describes setting up the Connection Manager.

2.1 Setting Up the Client Side

On the client side, configure settings for the conmgr process.

2.1.1 Setup Preparation Tasks

Describes the preparatory tasks to perform before you set up the conmgr process.

2.1.1.1 Security Policy Settings

In order for the Windows service to start and stop the conmgr process, Connection Manager requires a security setting that allows the OS user account that is to start the conmgr process to log on as a service.

Describes how to set up security so that you can log on as a service.

1. Displaying the [Local Security Policy] window

   In Windows, select [Administrative Tools], and then click [Local Security Policy].

2. Setting up security

   1. In the [Local Security Policy] window, select [Security Settings], select [Local Policies], and then click [User Rights Assignment].
   2. Under [Policy] in the [User Rights Assignment] window, double-click [Log on as a service].
   3. In the [Log on as a service Properties] window, set the following:

      a. Select the [Local Security Setting] tab.

      b. On the [Local Security Setting] tab, click [Add User or Group].

      c. In the [Select Users or Groups] window, enter the operating system user account of the instance administrator user in [Enter the object names to select].

      d. Click [OK].

   4. In the [Log on as a service Properties] window, click [OK].

2.1.1.2 Preparing for Output to the Event Log

Describes how to prepare for outputting error logs to the event log.

Information

If you have not registered the event source name, the content of the message output to the event log might be incomplete.

Register this default event source name before using the cm_ctl command because the event log may be output with a default event source name of "conmgr".

The following is an example of registering a DLL for a 64 bit product as the event source name "conmgr".

```
> regsvr32 /n /i:"conmgr" "c:\Program Files\Fujitsu\fsepv<x>client64\lib\pgevent.dll"
```

For a multi-version installation

If Fujitsu Enterprise Postgres is already installed on the same machine, note the path to the registered DLLs by searching for the following key in the Registry Editor. Then, register a new DLL with the default event source name.

The DLL path is used to re-register the default event source name during removing setup.

```
conmgr
```

2.1.2 Creating a Directory for the conmgr Process

You need one conmgr process for each set of instances that you want to configure for replication. Assign a dedicated directory to each conmgr process. This directory must assign read, execute, and write permissions for the user who starts the conmgr process.

This directory is specified when you run the cm_ctl command, which starts and stops the conmgr process. To specify a directory in the cm_ctl command, set it in the environment variable CMDATA or specify it in the -D option.

See

Refer to "cm_ctl" in the Reference for information on cm_ctl.

2.1.3 Configuring conmgr.conf

Place the configuration file conmgr.conf in the directory for the conmgr process.

Syntax for conmgr.conf

- In conmgr.conf, after the symbol(#) are considered comments.
- The parameter name = value" is a set of settings and must be written on one line.
- Set the value in a format that matches the type of each parameter. The types and formats are:
  - integer: Numeric type. Express as a sequence of numbers in decimal number.
  - string: String type. You can also include spaces by enclosing them in quotation marks('). If you include quotation marks, escape them.
  - enum: Enumeration type. Possible values are determined.

Parameters to Set

port (integer)

Specify the port number on which the conmgr process listens for connections from the applications.

The value must be greater than or equal to 1 and less than or equal to 65535. The default is 27546. You must restart conmgr process for this parameter change to take effect.

backend_host* (string)

Specify the host name or IP address of the instance.

You can also use IPv6 address. If you specify the IP address directly, you can save time by using backend_hostaddr parameter. If backend_host parameter and backend_hostaddr parameter are both specified, backend_hostaddr parameter is used. You must restart conmgr process for this parameter change to take effect.

To distinguish multiple instances, append a zero-based number immediately after the parameter name, such as backend_host0, backend_host1,... This number is called the instance number. A parameter identified by the same instance number configures the settings of a single instance. If you want to exclude some instances from your replication configuration, you can simply remove the settings for that instance.

Information

- Refer to "Considerations for System Configuration" in "1.1.2 Mechanism of Heartbeat Monitoring Feature" for details.

- If the primary is not included in the instances configured in conmgr.conf, use the -W option when starting Connection Manager with the cm_ctl command. Without the -W option, the cm_ctl command will not return until the primary connection is complete.

For example, if two instances are listening on "host name:host0, port number:5432" and "host name:host1, port number: 2345", write as follows.

backend_host0='host0'
backend_port0=5432
backend_host1='host1'
backend_port1=2345

You can also mix different instance number settings:

backend_host0='host0'
backend_host1='host1'
backend_port0=5432
backend_port1=2345

It does not matter if the instance number is missing as in the following (instance number 1):

backend_host0='host0'
backend_host2='host2'
backend_port0=5432
backend_port2=2345

If the host name is omitted, as in instance number 1 below, an error will occur when loading the configuration file.

backend_host0='host0'
backend_host2='host2'
backend_port0=5432
backend_port1=5555
backend_port2=2345

backend_hostaddr*(string)

Same as backend_host parameter except no name resolution is used.

backend_port* (integer)

Specify the port number the postmaster of the instance will listen on.

The value must be greater than or equal to 1 and less than or equal to 65535. The default is 27500. Append the instance number as you would for backend_host parameter. You must restart conmgr process for this parameter change to take effect.

watchdog_port* (integer)

Specify the port number on which the watchdog process listens.

The conmgr process connects to this port, but the user application does not. you must set it to the same value as watchdog.port parameter in postgresql.conf. The value must be greater than or equal to 1 and less than or equal to 65535. The default is 27545. Append the instance number as you would for backend_host parameter. You must restart conmgr process for this parameter change to take effect.

heartbeat_interval (integer)

Specify the interval at which heartbeat packets are sent for heartbeat monitoring.

Used in conjunction with heartbeat_timeout parameter. Connection Manager heartbeat monitoring always continues to send packets periodically from both ends of the connection. If a packet is not received from the other side within a certain period of time, the link is considered down.

Note that this method is different from TCP keepalive. TCP keepalive send a keepalive packet only when there is a certain amount of inactivity (idle), and expects to receive an ACK for that packet. If TCP keepalive does not receive an ACK, it repeats this a specified number of times and then assumes that the link is down.

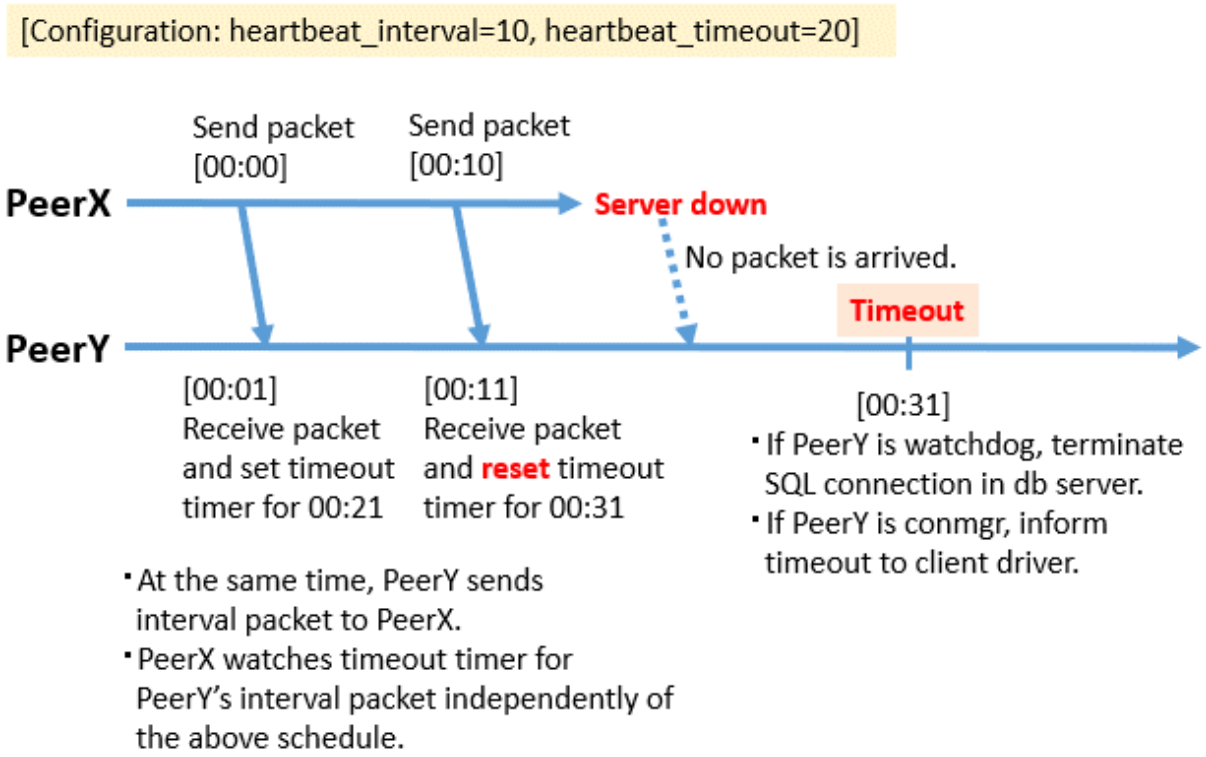
The heartbeat_interval parameter and heartbeat_timeout parameter are propagated from the conmgr process to the watchdog process, and also apply to the interval between the transmissions of heartbeat packets from the watchdog process. If a watchdog process is connected from both a conmgr process with a heartbeat_interval parameter of 3 seconds and a conmgr process with a heartbeat_interval of 5 seconds, it sends heartbeat packets every 3 seconds to the former process and every 5 seconds to the latter process. The unit is seconds. Specify a value equal to or more than 1 second. The default is 10 seconds. You must restart conmgr process for this parameter change to take effect.

heartbeat_timeout (integer)

If a heartbeat packet for heartbeat monitoring cannot be received for more than the time specified by this parameter, an error is assumed to have occurred and the application is notified of the error.

This parameter should be decide of heartbeat_interval parameter as the basis. No error is occured when the configuration file is loaded, but is always considered abnormal by heartbeat monitoring if it is at least not greater than heartbeat_interval parameter. The unit is seconds. Specify a value equal to or more than 1 second. The default is 20 seconds. You must restart conmgr process for this parameter change to take effect.

Refer to the following figure for the relationship between the heartbeat_interval parameter and heartbeat_timeout parameter settings and the heartbeat timeout.



heartbeat_connect_interval (integer)

Specify the interval between attempts to establish heartbeat monitoring again after detecting an abnormality.

This parameter is useful when only the database server is started, but not the instance. In such a situation, the TCP connection fails immediately, and retries cannot be attempted without an interval. If you specify an excessively long value, you may delay noticing the start of the instance. If a connection attempt fails for a long time, it will attempt the next connection after the time specified by heartbeat_connect_interval parameter has elapsed. The unit is seconds. Specify a value equal to or more than 1 second. The default is 1 second. You must restart conmgr for this parameter change to take effect.

heartbeat_connect_timeout (integer)

Specify the connection timeout for establishing heartbeat monitoring.

The connection includes the time it takes to send the TCP connection and the first heartbeat packet to the watchdog process and receive a reply from the watchdog process. This parameter is particularly needed when the other server is down or the network is disconnected. This is because TCP connections are attempted over a long period of time,

depending on the operating system configuration, and the connection takes a long time to fail. The unit is seconds. Specify a value equal to or more than 1 second. The default is 10 seconds. You must restart conmgr process for this parameter change to take effect.

log_destination (string)

Specify the destination of the message.

You can specify multiple destinations. Use commas to separate multiple entries and enclose all in single quotation marks.

"stderr" and "syslog" can be specified. The default is to print only to stderr. You must restart conmgr process for this parameter change to take effect.

"stderr" and "eventlog" can be specified. The default is to print only to stderr. You must restart conmgr process for this parameter change to take effect.

syslog_facility (enum)

Specify the syslog facility.

Valid only if log_destination parameter includes "syslog".

LOCAL0, LOCAL1, LOCAL2, LOCAL3, LOCAL4, LOCAL5, LOCAL6, or LOCAL7 can be specified. The default is "LOCAL0". You must restart conmgr process for this parameter change to take effect.

syslog_ident (string)

Specify the program name used to identify the output from the conmgr process.

The default is "conmgr". You must restart conmgr process for this parameter change to take effect.

event_source (string)

Specify the event source name used to identify the output from the conmgr process.

The default is "conmgr". You must restart conmgr process for this parameter change to take effect.

log_min_messages (enum)

Specifies the level of messages to output.

It can be DEBUG, INFO, NOTICE, WARNING, ERROR, LOG, FATAL, or PANIC. Messages below the specified level are not output. The default is "WARNING". You must restart conmgr process for this parameter change to take effect.

max_connections (integer)

Specifies the maximum number of simultaneous connections to the conmgr process.

If there are more than this maximum number of client connections, it forces the connection to be closed without sending an error message to the client.

The conmgr process also outputs this fact at level “LOG” to the destination specified by log_destination. Specify a value equal to or more than 0.

If 0 is specified, there is no limit. The default is 0. You must restart conmgr process for this parameter change to take effect.

Note

The maximum number of file descriptors that can be opened simultaneously (You can check it with -n of the ulimit command.) imposed on a conmgr process by the OS user limit should be greater than the value derived from the following equation. Otherwise, the conmgr process will abort if the user limit is violated.

```
9 + Number of database instances specified in conmgr.conf x 2  + max_connections
specified in conmgr.conf
```

loadbalance (string)

Specify the algorithm used for the load balancing feature as "roundrobin" or "leastconn".

When a connection request is received, the server to connect to is determined as follows:

roundrobin: It simply determines the connection destination according to the order of the server list defined in conmgr.conf.

leastconn: When a choice is available, it connects to the server that accepts the fewest connections.

The default is "roundrobin". You must restart conmgr process for this parameter change to take effect.

wait_streaming_replica_state (boolean)

Specify whether to exclude from connection candidates standby servers that are not synchronized with the primary server on a transaction-by-transaction basis.

If on is specified, it will be excluded from the connection destination. The default is off. You must restart conmgr process for this parameter change to take effect.

on_promote_action (string)

The following specifies the notification behavior to applications from the conmgr process when a standby server is promoted, using the read connection redistribution feature.

auto_disconnect: Notifying the message and disconnecting the connection.

none: Nothing works.

The default is "none". You must restart conmgr process for this parameter change to take effect.

standby_reconnect_threshold (integer)

With the read connection redistribution feature, you can specify a threshold for the number of connections to balance the total number of connections across all standby servers, including active ones, when a standby server recovers. Specify the average number of connections expected on a standby server.

When a server recovers, if the number of connections on an active server exceeds the specified value, a message may be sent to the application and the connection may be disconnected. In such cases, the acceptable range for imbalance will follow the value specified in the standby_reconnect_balance_ratio_target parameter.

If this parameter is omitted or set to 0, the read connection redistribution feature will not be enabled, and the application's connections will be maintained as they are.

The default is 0. You must restart conmgr process for this parameter change to take effect.

standby_reconnect_balance_ratio_target (floating point)

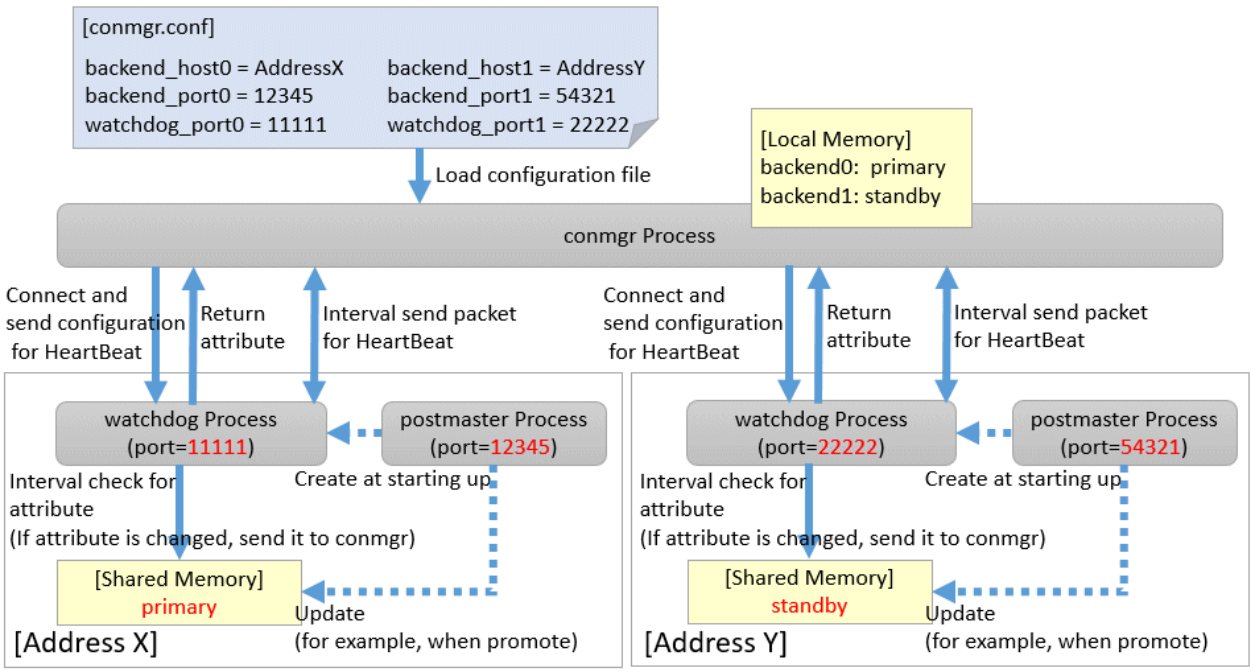
With the read connection redistribution feature, you can specify a target ratio as a decimal value between 0.2 and 0.8 to resolve the imbalance in the number of connections between nodes when a standby server recovers. This applies when standby_reconnect_threshold is enabled.

For example, if you specify 0.3, message notifications and disconnections will be performed for applications (clients) connected to the server with the most connections until the maximum number of connections across all standby servers is 1.3 times or less than the minimum number of connections, aiming to converge to the set ratio.

The default is 0.3. You must restart conmgr process for this parameter change to take effect.

Overview of connections definitions

The following figure shows the relationship between the IP address or host name and the port number set in conmgr.conf and the processes.



2.1.4 Registering the conmgr Process with a Windows Service

To start or stop the conmgr process using a Windows service, register the conmgr process with the Windows service. Describes how to register the conmgr process with a Windows service.

Register the conmgr process with a Windows service by using the cm_ctl command with the service name to register, the user name, the password, and the path to the directory where the conmgr.conf is located.

Execute the command as a Connection Manager administrator user with "Administrator" privileges. Execute the command from the [Administrator: Command Prompt] window. Right-click [Command Prompt], and then select [Run as administrator] from the menu to display the [Administrator: Command Prompt] window.

Example)

The following example is executed when the registration service name is "conmgr", the user name is "fsepuser", and the directory where confgr.conf is located is "D: \conmgr".

```
> cm_ctl register -N "conmgr" -U fsepuser -P ******** -D D:\conmgr
```

You must specify a user name and password for the following reasons:

- Because Windows services are started under the Network Service account, the conmgr process is created for that account. As a result, the conmgr process may fail to start on another user. For security reasons, if you do not specify a user name and password, specify the account from the list of Windows services immediately after you register the conmgr process with the Windows service.

Information

Do not specify the --complete option when registering Windows services in register mode of the cm_ctl command. This is because the conmgr process may be running in the background if it fails to start using Windows services.

Note

When entering the password for the cm_ctl command, be careful that it is not referenced by other users for security reasons.

2.2 Setting Up the Server Side

On the server side, configure settings for the watchdog process.

2.2.1 Configuring postgresql.conf

Describes the postgresql.conf parameters that must be set when using the Connection Manager.

Parameters to Set

max_connections

An existing PostgreSQL parameter. Add 2 to the value already set.

Connection to the instance is maintained from the time the instance is started to do the following:

- The watchdog process checks the state of the instance.
- The terminator process forces the client to terminate the SQL connection.

shared_preload_libraries

An existing PostgreSQL parameter. Add a watchdog.

The watchdog process and terminator process start when you add watchdog and restart the instance.

watchdog.port (integer)

Specify the port number on which the watchdog process accepts connections for heartbeat monitoring from the conmgr process.

The value must be greater than or equal to 1 and less than or equal to 65535. The default is 27545. The instance must be restarted for this parameter change to take effect.

watchdog.check_attr_interval (integer)

Specify the interval between checking the attributes of a instance.

watchdog process immediately notifies the conmgr process if the attribute changes.

Also, if you are using the standby initial data synchronization waiting feature, it is also used for the interval to check the status of the standby server.

The unit is milliseconds. Specify a value equal to or more than 1 millisecond. The default is 1000 milliseconds. The instance must be restarted for this parameter change to take effect.

watchdog.max_heartbeat_connections (integer)

Specify the maximum number of conmgr processes that connect to watchdog process.

The default is the value specified in max_connections of postgresql.conf.

There is no upper limit, but about 200 bytes of memory are consumed for 1 connection when PostgreSQL is started.

L

Information

Normally you do not need to consider, but if you have a heartbeat connection with a very large number of conmgr processes, it may violate on the maximum number of file descriptors (You can check it with -n of the ulimit command.) of the OS user limit. This is because the socket for the heartbeat connection consumes the file descriptor. Set the maximum number of file descriptors of the OS user limit to a value larger than the value calculated below from the max_files_per_process parameter value and watchdog.max_heartbeat_connections parameter value in postgresql.conf.

```
max_files_per_process + watchdog.max_heartbeat_connections x 2
```

2.2.2 Introducing the watchdog extension

Execute the CREATE EXTENSION statement with watchdog.

Example)

```
postgres=# CREATE EXTENSION watchdog;
CREATE EXTENSION
```

This allows you to see the pgx_stat_watchdog view for information about the watchdog process.

2.2.3 Windows Firewall Settings

Describes how to enable the port number used by the watchdog process when you enable the Windows Firewall feature.

Windows Server(R) 2016:

1. In the [Windows Firewall] window, click [Advanced settings] on the left side of the window.
2. In the [Windows Firewall with Advanced Security] window, click [Inbound Rules] on the left side of the window.
3. Click [New Rule] on the right side of the window.
4. In the [New Inbound Rule Wizard] window, select [Port], and then click [Next].
5. Select [TCP] and [Specific local ports], then specify the "port number on which the watchdog process accepts connections for life-and-death monitoring from the conmgr process (watchdog.port)" defined in postgresql.conf in the text box, and then click [Next].
6. Select [Allow the connection], and then click [Next].
7. Select the profiles for which this rule applies, and then click [Next].
8. In [Name], specify the desired name(ex: Connection Manager), and then click [Finish].
9. In the [Windows Firewall with Advanced Security] window, check if the added rule is enabled under [Inbound Rules] in the center of the window.

Other than above:

1. In the [Windows Defender Firewall] window, click [Advanced settings] on the left side of the window.
2. In the [Windows Defender Firewall with Advanced Security] window, click [Inbound Rules] on the left side of the window.
3. Click [New Rule] on the right side of the window.
4. In the [New Inbound Rule Wizard] window, select [Port], and then click [Next].
5. Select [TCP] and [Specific local ports], then specify the "port number on which the watchdog process accepts connections for life-and-death monitoring from the conmgr process (watchdog.port)" defined in postgresql.conf in the text box, and then click [Next].
6. Select [Allow the connection], and then click [Next].
7. Select the profiles for which this rule applies, and then click [Next].
8. In [Name], specify the desired name(ex: Connection Manager), and then click [Finish].
9. In the [Windows Defender Firewall with Advanced Security] window, check if the added rule is enabled under [Inbound Rules] in the center of the window.

2.3 Starting the conmgr Process

2.3.1 Starting and Stopping the conmgr Process

Describes starting and stopping the conmgr process.

You can use commands related to Windows services to start, stop, and check the health of the conmgr process.

To use a Windows service, register the confmgr process with the Windows service.

Information

Although it is possible to start and stop the conmgr process by running the cm_ctl command without registering the conmgr process with the Windows service, we recommend that you use the Windows service to start and stop the conmgr process for the following reasons.

The cm_ctl command starts the conmgr process as a user process. Therefore, closing the command prompt window from which the command was executed will force the conmgr process to stop.

Starting the conmgr process

Start with the net start or sc start command and the service name.

You can also start it from the Windows Services window by following the steps below.

1. Display the [Services] window

   In [Administrative Tools], click [Services].

2. Start a service

   From the list of service names, select the conmgr process and click the [Start Service] button.

Stopping the conmgr process

Start with the net stop or sc stop command and the service name.

You can also stop it from the Windows Services window by following the steps below.

1. Display the [Services] window

   In [Administrative Tools], click [Services].

2. Stop a service

   From the list of service names, select the conmgr process and click the [Stop Service] button.

Checking the operating status of the conmgr process

To check whether a service is running immediately after the start operation of the conmgr process, check the status of the service as follows.

1. Display the [Services] window

   In [Administrative Tools], click [Services].

2. Check the status of the service

   Check the service status of the conmgr process from the list of service names.

Use the cm_ctl command to check the operating status of the conmgr process during operation.

See

Refer to "cm_ctl" in the Reference for information on cm_ctl.

2.3.2 Setting of Automatic Start / Stop of conmgr Process

The conmgr process can be started/stopped automatically according to the start/stop of the database server OS.

If the service startup type is "manual", change it to "automatic". By setting it to "automatic", it will start automatically when the database server OS is started, and will automatically stop when the database server OS is shut down.

The automatic start/stop settings for the conmgr process should be performed by a user with "Administrator" privileges.

Follow the procedure below to switch services.

1. Display the [Services] window

   In [Administrative Tools], click [Services].

2. Switch startup

   Select the service name of the conmgr process, display the [Properties] dialog, and switch the startup type from Manual to Automatic.

You can also change the above settings with the sc config command.

2.4 Removing Setup

Describes how to removing setup the Connection Manager.

Linux

No work is required on the client side.

On the server side, drop watchdog extension by DROP EXTENSION statement and remove it from shared_preload_libraries.

Example)

```
postgres=# DROP EXTENSION watchdog;
DROP EXTENSION
```

Windows

The following work is required on the client side.

1. Unregister Windows service

   Cancel the conmgr process registered in the Windows service.

   In unregister mode of the cm_ctl command, specify the registered service name to cancel the conmgr process of the Windows service.

   Execute the command as a Connection Manager administrator user with "Administrator" privileges. Execute the command from the [Administrator: Command Prompt] window. Right-click [Command Prompt], and then select [Run as administrator] from the menu to display the [Administrator: Command Prompt] window.

   Example)

   An execution example is shown when the registered service name is "conmgr".

   ```
   > cm_ctl unregister -N "conmgr"
   ```

2. Deletion of registration for event log

   If you are outputting to the event log in "2.1.1.2 Preparing for Output to the Event Log", delete the registered event source name.

   Example)

```
> regsvr32 /u /i:"conmgr" "c:\Program Files\Fujitsu\fsepv<x>client64\lib\pgevent.dll"
```

For a multi-version installation

If the conmgr process created using this package is configured to output the error log to the event log, re-register the default event source name using the DLL pathname that was saved in "2.1.1.2 Preparing for Output to the Event Log".

On the server side, use the DROP EXTENSION statement to remove the watchdog extension and remove the watchdog from shared_preload_libraries.

Example)

```
postgres=# DROP EXTENSION watchdog;
DROP EXTENSION
```

Chapter 3 Using from an Application

Describes how to use the Connection Manager from an application.

3.1 Connection Method

When connecting to the instance using ConnectionManager, specify the following values in the connection parameters of the application. Application connection parameters are parameters that specify the database IP address, host name, port number, etc., which are originally specified when connecting to the database from the application. For example, when using libpq, specify "localhost" for the host parameter and specify the port number on which the conmgr process listens for the port parameter.

Connection parameters not shown here are used directly by the instance in the second stage of the connection, connecting to the instance (connecting to an instance without the Connection Manager), and the conmgr process does not check or use it.

Connection destination address

Specify "localhost". Unix domain sockets are not allowed.

It is possible to connect to a remote conmgr process, but it should not be used for other purposes expect such as testing. This is because there is no mechanism between the application and the conmgr process to detect the remote server down or the network link down, making the Connection Manager meaningless.

Port number

Specify the value specified for the port parameter in conmgr.conf.

Connection destination instance attributes

Follow the "Target server" in the application connection switch feature. Refer to "Taget server" in "Connection Information for the Application Connection Switch Feature" in the "Application Development Guide" for information on the target server in the application connection switch feature.

Information

- If you specify 'direct' for the sslnegotiation connection parameter, you cannot use Connection Manager.
- Connection Manager cannot be used if preferPrimary is specified for targetServerType when using the JDBC driver.

3.2 How to Detect Instance Errors

Only special if you are using libpq's asynchronous communication method. For additional discovery methods, refer to "Errors when an Application Connection Switch Occurs and Corresponding Actions" of the for each client driver in the "Application Development Guide" . If the conmgr process goes down while accessing it, or if the conmgr process tries to establish a SQL connection while it is down, the same error is returned as if the instance went down.

3.3 How to Use in libpq

libpq provides very detailed communication control. Therefore, to detect a heartbeat error through the conmgr process, you may need to modify the existing application logic.

See

Refer to "libpq - C Library" in the PostgreSQL Documentation on functions described below.

3.3.1 Specifying a Connection Destination

The host or hostaddr parameter of the connection string should specify a connection destination for only one conmgr process.

Information

The host or hostaddr parameter of the connection string can be a mixture of multiple conmgr process destinations and the postmaster destination. The first connection from an application process is attempted in the listed order, and the second and subsequent connections are attempted in the order in which the previous connection was made. However, if a connection to the conmgr process is confirmed, subsequent connections are always attempted in the listed order.

For example, if conmgr1, conmgr2 are specified, and if conmgr1 does not initially know a server with the attributes specified in the target_session_attrs parameter, it will query conmgr2 for a destination. In the second and subsequent connections, since there was a connection to the conmgr1 process, the connection destinations are queried in the order of the listed conmgr1 and conmgr2.

Also, for example, if postmaster1, conmgr1 is specified, it will try to connect directly to the database instance initially named postmaster1. If this fails, it queries conmgr1 for a destination. On subsequent connections, since there was a connection to the conmgr1 process, it queries the listed postmaster1, then conmgr1, for connections.

3.3.2 Using the Asynchronous Connection Method

An asynchronous connection method is to use a function like PQconnectStart() instead of a function like PQconnectdb (). PQconnectStart() returns without synchronizing the completion of the connection to the database. The user application must then monitor the sockets returned by PQsocket() to be readable or writable, for example by using the poll() system call, according to the values required by the return value of PQconnectPoll().

With the Connection Manager, the socket returned by PQsocket() may change after a call to PQconnectPoll(), so be sure to reacquire the socket that you want to give to the poll() system call using PQsocket(). This behavior is similar to simply specifying multiple hosts in the connection string without using the Connection Manager.

3.3.3 Using an Asynchronous Communication Method

An asynchronous communication method is one in which the application returns control without waiting for a response from the database, and PQsetnonblocking() is used to asynchronize completion of transmission or completion of receipt of all results. Instead of using a function like PQexec(), use a function like PQsendQuery(). In this method, the user application monitors the socket that connects to the database returned by PQsocket(), for example, by using the poll() system call.

For example, if the link to the database goes down, simply monitoring the socket returned by PQsocket() with the poll() system call will not detect it.

However, it is possible to detect the reception of database anomaly detection packets sent from the conmgr process, for example, by monitoring the reception of data on the socket (POLLIN) connecting to the conmgr process returned by PQcmSocket(). Once a reception is detected, the user application need not directly manipulate the packet. By calling something like PQgetResult() or PQcosumeInput() according to the existing application logic, it behaves as if the connection were disconnected. Refer to "Errors when an Application Connection Switch Occurs and Corresponding Actions" in the Application Development Guide on SQLSTATE returned,etc. If you are not using the Connection Manager, PQcmSocket() returns -1.

3.3.4 Behavior of PQhost() or PQhostaddr() or PQport()

PQhost(), PQhostaddr() or PQport() typically return a host parameter or hostaddr parameter or port parameter specified in the connection string by the user application. However, if you specify a connection destination for the conmgr process, the destination for the conmgr process you specify will be changed to the database connection destination information provided

by the conmgr process before the connection is completed. This behavior is similar to simply specifying multiple hosts in the connection string without using the Connection Manager.

3.3.5 Behavior of PQstatus()

If you are using an asynchronous connection method, you can monitor the intermediate state of the connection to the database with PQstatus(). If you are using the Connection Manager, the enum value returned by PQstatus() is appended with the following:

```
CONNECTION_AWAITING_CMRESPONSE
/ * Waiting for a response from the conmgr process * /
```

3.3.6 PQcmSocket()

You can get a socket that leads to the conmgr process. It returns a value equal to or more than 0 for a valid socket, or -1 if you are not connected to the conmgr process.

```
int PQcmSocket(const PGconn *conn);
```

3.4 How to Use in ODBC Driver

Describes points to note when using the Connection Manager using the ODBC driver.

3.4.1 Behavior of SQLGetInfo()

When SQL_SERVER_NAME is specified in the argument InfoType, SQLGetInfo () normally returns the contents set in Servername or Server of the data source. However, if you specify a connection destination for the conmgr process, the destination for the conmgr process you specify will be changed to the database connection destination information provided by the conmgr process before the connection is completed. This behavior is similar to simply specifying multiple hosts in the connection string without using the Connection Manager.

3.5 How to Use in JDBC Driver

Describes points to note when using the Connection Manager using the JDBC driver.

3.5.1 Behavior of loadBalanceHosts Parameter

The loadBalanceHosts parameter is a connection parameter for the JDBC driver to use the load balancing feature. You can specify whether to use the load balancing feature by setting this parameter. However, Connection Manager provides a unique load balancing feature that users cannot specify whether to use or not. Therefore, even if the user sets the loadBalanceHosts parameter to disable the JDBC driver load balancing feature, the Connection Manager load balancing feature is always enabled when connecting to the database via the Connection Manager.

3.6 How to Use in Python Language Package (psycopg)

Describes points to note when using the Connection Manager using the Python language package (psycopg).

In the Python language package (psycopg), you can use the psycopg.Connection class and the psycopg.AsyncConnection class to work with the Connection Manager. If you want to use methods that are not defined in these two classes, also refer to "3.3 How to Use in libpq".

3.6.1 Specifying a Connection Destination

Refer to "3.3 How to Use in libpq".

3.6.2 Behavior of psycopg.ConnectionInfo.host() or psycopg.ConnectionInfo.hostaddr() or psycopg.ConnectionInfo.port()

psycopg.ConnectionInfo.host(), psycopg.ConnectionInfo.hostaddr() or psycopg.ConnectionInfo.port() typically return a host parameter or hostaddr parameter or port parameter specified in the connection string by the user application. However, if you specify a connection destination for the conmgr process, the destination for the conmgr process you specify will be changed to the database connection destination information provided by the conmgr process before the connection is completed. This behavior is similar to simply specifying multiple hosts in the connection string without using the Connection Manager.

3.6.3 Behavior of psycopg.ConnectionInfo.status()

If you are using an asynchronous connection method, you can monitor the intermediate state of the connection to the database with psycopg.ConnectionInfo.status(). If you are using the Connection Manager, the enum value returned by psycopg.ConnectionInfo.status() is appended with the following:

```
CONNECTION_AWAITING_CMRESPONSE
/ * Waiting for a response from the conmgr process * /
```

3.7 How to Use in Go Language

Describes points to note when using the Connection Manager using the Go language.

3.7.1 Specifying a Connection Destination

The host parameter of the connection string should specify a connection destination for only one conmgr process.

Appendix A System Views

A.1 pgx_stat_watchdog

A row in this view corresponds to conmgr process, which is connected to watchdog process. Additional columns may be added in future versions versions.

Column	Type	Description
conmgr_addr	inet	IP address of conmgr process.
conmgr_port	integer	The conmgr (ephemeral) port number that conmgr process is using to communicate with watchdog process. This is not the port number to be set in conmgr.conf.
heartbeat_interval	integer	The interval at which heartbeat packets are sent to and from this conmgr process. The unit is seconds.
heartbeat_timeout	integer	The timeout value for the heartbeat to and from this conmgr process. The unit is seconds.

Index