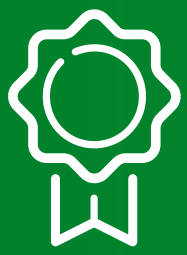
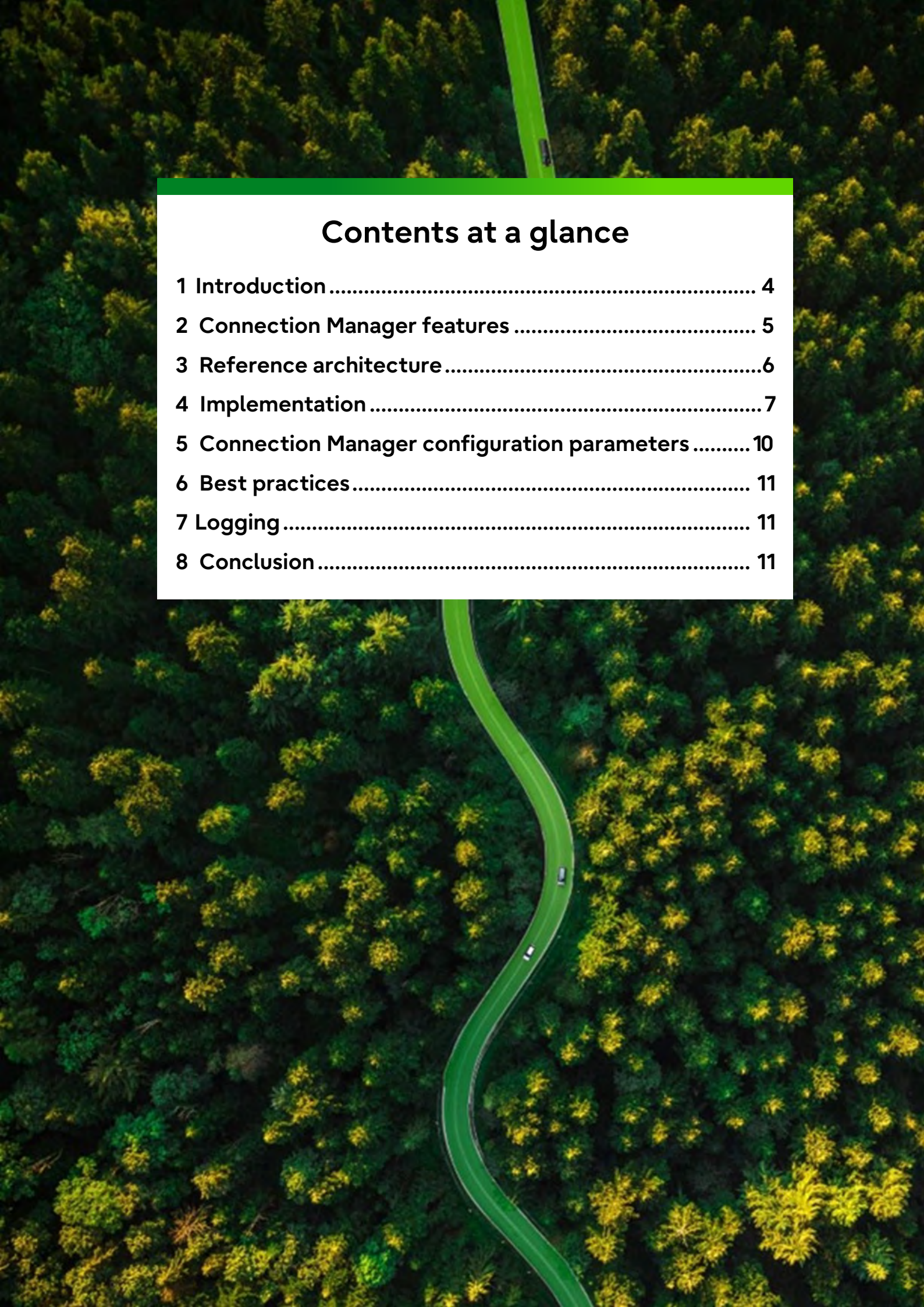
Best practices for Fujitsu Enterprise Postgres Connection Manager

Best practices





Contents at a glance

Table of contents

1. Introduction

The Connection Manager (CM) component of Fujitsu Enterprise Postgres is designed to enhance high-availability and seamless connectivity for client applications in clustered database environments. Connection Manager provides mechanisms that allow applications to connect to whichever database instance is appropriate (for example, after a fail-over) without needing explicit logic in the application layer.

Fujitsu Enterprise Postgres Connection Manager provides heartbeat monitoring and transparent connection support features. The Connection Manager monitors client/server and Fujitsu Enterprise Postgres instances running on the database server. If there is a physical server failure and the inter-server network link goes down, the Connection Manager notifies the client and the database instance. The Connection Manager transparent connection feature allows the application to connect to an instance from a set of instances that is configured for streaming replication. The connection to an underlying database instance is transparent to the application.

Benefits of using Connection Manager:

- Automatic reaping of connection happens when an abnormality is observed on the application server.
- The connection to the database instance is transparent to the application, and the application is notified at the time of the database instance failure.
- The application connection switches to the available database instance when database failover happens.
- After a connection is established, the application requests directly go to the database instance, so there is no performance degradation.

2. Connection Manager features

This section describes the Connection Manager features.

2.1. Heartbeat monitoring

The heartbeat monitoring feature of Connection Manager allows the system to detect unreachable or failed database instances more proactively than typical operating-system TCP keep-alive. While the TCP keepalives can fail to detect certain failure modes (e.g., where the TCP layer retransmits indefinitely or where the remote host is unreachable but still at TCP level "alive"), the Connection Manager mechanism uses an application-layer heartbeat between client-side conmgr and server-side "watchdog" process. On the client side, a single conmgr process is started (via cm_ctl) for a given set of instances. On the server side, a PostgreSQL extension "watchdog" is installed; this spawns two background workers at instance startup: one for sending/receiving heartbeat packets to/from conmgr, and one (the terminator) for forcibly terminating SQL connections from clients whose heartbeat monitoring fails (by IP address) even if those connections themselves don't explicitly go down.

2.2. Transparent connection support

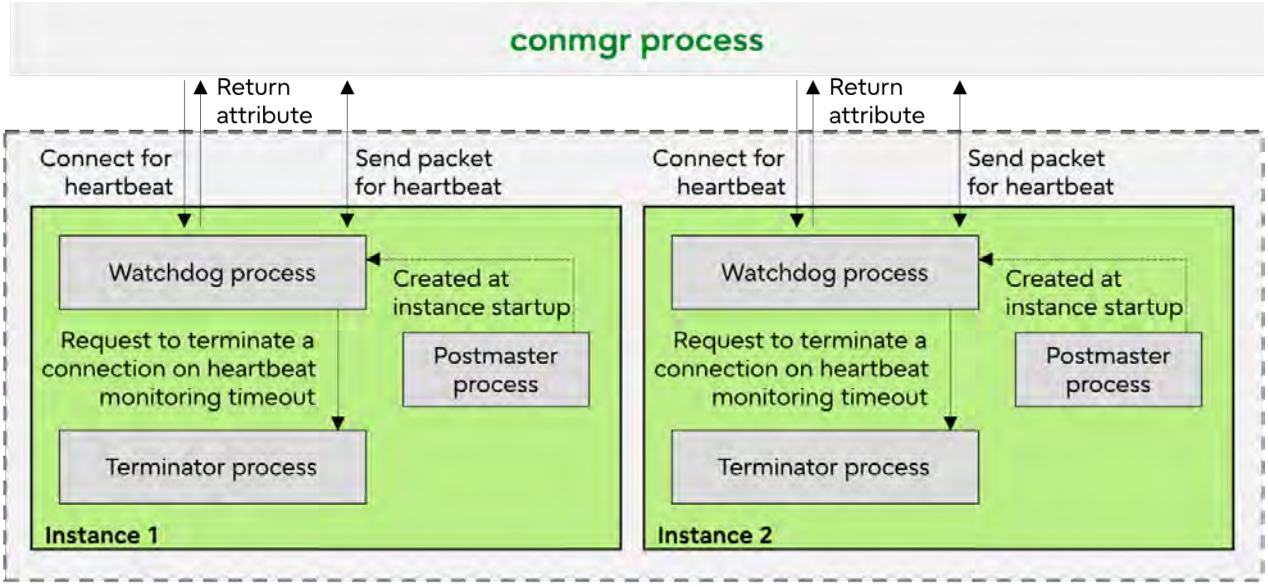
Transparent Connection Support allows an application to connect to the database without needing to know which host is currently the primary or standby (or which instance is appropriate). With Connection Manager, the client driver connects initially to the conmgr listening endpoint; conmgr then routes the connection to the correct database instance (based on heartbeat/monitoring state) so the application sees essentially a single endpoint.
Internally this works in two phases: the client driver connects to the local conmgr (or Connection Manager endpoint) which receives the request and then forwards (or instructs the driver to connect) to the target database instance. Once the actual SQL session is established, the driver and database communicate directly (so there is minimal extra latency from Connection Manager beyond the initial resolution). Transparent connection support thus enables faster failover and less application-impact, because the application doesn't need to implement custom logic such as "try host A, then host B" or "query which host is primary". Connection Manager handles that logic and hides the complexity.

3. Reference architecture

Connection Manager is configured on the client/server and the database server. On the client side, a monitoring process is started that is called the 'conmgr' process to monitor the set of database instances.

On the server side, the PostgreSQL extension Watchdog is installed. Due to the Watchdog extension, Postmaster starts two processes as background workers at database instance startup.

One process that is called Watchdog is for heartbeat monitoring, and the second process that is called Terminator is to forcibly terminate client SQL connections when the Watchdog process detects a failure on heartbeat monitoring.



Set of instances configured for replication

The diagram shows that primary and standby instance configured with streaming replication, with watchdog process sharing heartbeat and returning status of the instance with conmgr process running on the application side.

When Connection Manager is configured on Fujitsu Enterprise Postgres instances configured with Mirroring Controller to provide automatic failover, the Connection Manager at the front-end provides transparent connectivity to the instance, based on attributes in the client connection string.

When a standby database is promoted as primary or an automatic failover happens, the watchdog process updates the state of the instance with conmgr process, so that the connections to that server can be initiated as soon as the application requests it.

If the instance set in the conmgr.conf includes multiple standby servers configured for streaming replication, read requests are automatically distributed among those standby servers to achieve load-balancing.

4. Implementation

4.1. On database server

On the database server side, implementation requires installing the server-side extension 'Watchdog' for Connection Manager, enabling it in the postgresql.conf file. After the extension is created, the watchdog and terminator background workers are started. On the database server, the terminator process may forcibly disconnect client connections based on IP address if the heartbeat fails. The server must also accept heartbeat traffic from conmgr processes (on defined ports).

To set up Connection Manager on the database server,

1. Add below parameters to postgresql.conf

```
## Add to $PGDATA/postgresql.conf
shared_preload_libraries = 'watchdog'
watchdog.port = 27545
```

2. Restart the Fujitsu Enterprise Postgres cluster

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

3. Create database extension

```
CREATE EXTENSION watchdog;
```

4. Repeat the above steps on the standby servers as well.

4.2. On application server

On the application server, implementation involves installing the Fujitsu Enterprise Postgres Client software and configuring the conmgr process for the set of database instances. The conmgr.conf file defines the instance set (primary and one or more standby servers in streaming replication), host/port of each database and monitoring parameters (heartbeat interval, timeout, etc).

The Connection Manager must be installed on the server where application is hosted and if applications running on multiple servers share the same database cluster, then install and configure Connection Manager on all the application servers.

Steps to configure Connection Manager on Application server:

1. On application server, as appuser, create a conmgr.conf file for connection manager.

```
## Create a directory
$ mkdir /home/appuser/conn_mgr
## Create conmgr.conf file
$ vi /home/appuser/conn_mgr/conmgr.conf
port = 27546
backend_host0 = '10.0.0.10'
backend_host1 = '10.0.0.20'
backend_port0 = 27500
backend_port1 = 27500
watchdog_port0 = 27545
watchdog_port1 = 27545
log_destination = 'syslog'
```

2. Start the Connection Manager process.

```
$ cm_ctl -D /home/appuser/conn_mgr start
cm_ctl: conmgr process is ready (25077)
cm_ctl: waiting conmgr process to connect to watchdog (25070)
cm_ctl: started conmgr process successfully (25078)
```

3. Check the status of the Connection Manager

```
$ cm_ctl -D /home/appuser/conn_mgr status -i instance
conmgr_status:
status           pid
ready            30579

instance_information:
addr                                port  database_attr
10.0.0.10                           27500 primary
10.0.0.20                           27500 standby
```

The application connection string is pointed at the conmgr endpoint (e.g., host=localhost port=<CM_PORT>), and you typically set driver attributes like target_session_attrs (in libpq/JDBC) so the driver works with Connection Manager seamlessly.
On Linux, ensure that conmgr is started as a daemon or systemd service, and that it remains running for the lifecycle of the application processes. Also, ensure that the client side has network visibility and route connectivity to all target database hosts defined in conmgr.conf. It is strongly recommended *not* to run conmgr process from the database host.

4.3. Supported client drivers

Connection Manager supports following client drivers,

- libpq (C language library) – supported from Fujitsu Enterprise Postgres 12 onwards
- ECPG (embedded SQL in C) – supported from Fujitsu Enterprise Postgres 12 onwards
- ECOBPG (embedded SQL in COBOL) – supported from Fujitsu Enterprise Postgres 14 onwards
- JDBC driver – supported from Fujitsu Enterprise Postgres 14 onwards
- ODBC driver connection – supported from Fujitsu Enterprise Postgres 14 onwards
- Python language package (psycopg) – supported from Fujitsu Enterprise Postgres 17 SP1 onwards

Note: Golang is not supported.

4.4. User and privileges

- Fujitsu Enterprise Postgres Client installation on application server requires root or sudo access.
- Run Connection Manager's conmgr process from the application user account (appuser).
- Root or sudo privileges must not be used for daily operations.

5. Connection Manager configuration parameters

Configuration parameters for “conmgr.conf” on the application servers. For example, conmgr.conf:

Parameter	Description	Remarks
port	Specify the port number on which the conmgr process listens for connections from the applications.	
backend_host*	Specify the host name or IP address of the instance.	To distinguish multiple instances, append a zero-based number immediately after the parameter name, such as backend_host0, backend_host1,etc.
backend_port*	Specify the port number the postmaster of the database instance will listen on.	
watchdog_port*	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.
heartbeat_interval	The interval at which heartbeat packets are sent to and from this conmgr process. The unit is seconds.	The default is 10 seconds.
heartbeat_timeout	The timeout value for the heartbeat to and from this conmgr process. The unit is seconds.	The default is 20 seconds.
log_destination	Specify the destination to log the messages. "stderr" and "syslog" can be specified. The default is to print only to stderr.	
syslog_ident	Specify the program name used to identify the output from the conmgr process. The default is "conmgr".	

6. Best practices

Some of the best practices to follow when installing and configuring Connection Manager.

- **Configure driver and connection string properly**: Use the driver parameter `target_session_attrs` for libpq (or `targetServerType` for JDBC) combined with Connection Manager endpoint so that client applications don't need custom logic. Let Connection Manager handle redirection to primary/standby.
- **Ensure failover readiness:** Because transparent connection support allows rapid switch, test the failover path (simulate primary failure, ensure standby becomes primary, Connection Manager redirects connections) and confirm that session interruption is within acceptable bounds.
- **Avoid co-locating `conmgr` and Database instance:** Particularly in replication setups, don't deploy the `conmgr` process on the same host as the upstream database instance to avoid unintended termination of replication connections.
- **Tune heartbeat parameters:** In low-latency, high-reliability networks you can afford lower `heartbeat_interval` values (for faster detection), but in higher latency or cloud-network setups you may want to increase `heartbeat_timeout` to avoid false detection. Review the default values and adjust based on your environment. The system view `pgx_stat_watchdog` can be queried from database instance, to list the heartbeat values set on the conmgr.conf file.
- **Monitor Connection Manager health:** On both client and server side, monitor the health of `conmgr`, watchdog/terminator processes, and look for logs indicating heartbeat failures or forced terminations. Set alerts for such events.

7. Logging

These are the log locations that are helpful when troubleshooting.

Component	Location	Notes
PostgreSQL	$PGDATA/log/postgresql.log	Database and instance logs
Connection Manager log	/var/log/messages	Specify the 'log_destination' parameter to 'syslog' in the conmgr.conf.

8. Conclusion

The Connection Manager feature in Fujitsu Enterprise Postgres provides a robust mechanism to enhance connectivity resiliency and availability for client applications in clustered database environments. By combining heartbeat monitoring and transparent connection support, it enables near-seamless client failover and reduces the burden on application logic. When implemented on Linux, careful attention to client-side conmgr configuration, server-side watchdog/terminator deployment, heartbeat parameter tuning, network isolation, and operational readiness testing are key to maximizing the benefit of Connection Manager. Coupled with application-level timeout safeguards, Connection Manager serves as a powerful component in a high-availability architecture.

FUJITSU

Fujitsu Enterprise Postgres is the enhanced version of PostgreSQL, for enterprises seeking a more robust, secure, and fully supported edition for business-critical applications

Contact
Fujitsu Limited
Email: enterprisepostgresql@fujitsu.com
Website: fast.fujitsu.com

2025-10-29 WW EN