# CLUSTERPRO® X *for Linux*

PPガイド (SQL Server)

2021.03.09
第2版

CLUSTERPRO

改版履歴

| 版数 | 改版日付 | 内 容 |
|---|---|---|
| 1 | 2018/07/05 | PPガイド（データベース）より分冊し、新規作成 |
| 2 | 2021/03/09 | SQL Server 2019 関連内容を追記 |
| | | |
| | | |
| | | |
| | | |
| | | |

# 免責事項

本書の内容は、予告なしに変更されることがあります。
日本電気株式会社は、本書の技術的もしくは編集上の間違い、欠落について、一切責任をおいません。
また、お客様が期待される効果を得るために、本書に従った導入、使用および使用効果につきましては、お客様の責任とさせていただきます。
本書に記載されている内容の著作権は、日本電気株式会社に帰属します。本書の内容の一部または全部を日本電気株式会社の許諾なしに複製、改変、および翻訳することは禁止されています。

# 商標情報

CLUSTERPRO® X は日本電気株式会社の登録商標です。
Microsoft、SQL Server は、米国 Microsoft Corporation の米国およびその他の国における登録商標です。
本書に記載されたその他の製品名および標語は、各社の商標または登録商標です。
その他のシステム名、社名、製品名等はそれぞれの会社の商標及び登録商標です。

# 目次

# はじめに

## 対象読者と目的

『CLUSTERPRO® PPガイド』は、クラスタシステムに関して、システムを構築する管理者、およびユーザサポートを行うシステムエンジニア、保守員を対象にしています。
本書では、CLUSTERPRO 環境下での動作確認が取れたソフトウェアをご紹介しています。ここでご紹介するソフトウェアや設定例は、あくまで参考情報としてご提供するものであり、各ソフトウェアの 動作保証をするものではありません。

## 適用範囲

本書は、 以下の製品を対象としています。
CLUSTERPRO X 4.2 for Linux
CLUSTERPRO X 4.1 for Linux
CLUSTERPRO X 4.0 for Linux

SQL Server 2019 Enterprise / Standard
SQL Server 2017 Enterprise / Standard

※ マイクロソフト社より無償提供される以下のエディションは、SQL Server の PP・サポートサービス対象外であるため、本書の適用対象外となりますのでご留意ください。
・Express Edition
・Developer Edition

## お問い合わせについて

本書の SQL Server 製品に関する記載内容のお問い合わせには、原則として CLUSTERPRO の保守契約と SQL Server の弊社での保守契約が必要です。

SQL Server 製品の障害発生時には、保守契約に則り、以下の NEC サポートポータルから NEC SQL Server Response Center までお問い合わせください。
・ NEC サポートポータル (https://www.support.nec.co.jp/)

# CLUSTERPRO マニュアル体系

CLUSTERPRO のマニュアルは、以下の 6 つに分類されます。各ガイドのタイトルと役割を以下に示します。

**『CLUSTERPRO X スタートアップガイド』**(Getting Started Guide)
CLUSTERPRO を使用するユーザを対象読者とし、製品概要、動作環境、アップデート情報、既知の問題などについて記載します。

**『CLUSTERPRO X インストール & 設定ガイド』**(Install and Configuration Guide)
CLUSTERPRO を使用したクラスタ システムの導入を行うシステム エンジニアと、クラスタシステム導入後の保守・運用を行うシステム管理者を対象読者とし、CLUSTERPRO を使用したクラスタ システム導入から運用開始前までに必須の事項について説明します。実際にクラスタ システムを導入する際の順番に則して、CLUSTERPRO を使用したクラスタ システムの設計方法、CLUSTERPRO のインストールと設定手順、設定後の確認、運用開始前の評価方法について説明します。

**『CLUSTERPRO X リファレンス ガイド』**(Reference Guide)
管理者、およびCLUSTERPRO を使用したクラスタ システムの導入を行うシステム エンジニアを対象とし、CLUSTERPRO の運用手順、各モジュールの機能説明およびトラブルシューティング情報等を記載します。『インストール & 設定ガイド』を補完する役割を持ちます。

**『CLUSTERPRO X メンテナンスガイド』** (Maintenance Guide)
管理者、および CLUSTERPRO を使用したクラスタシステム導入後の保守・運用を行うシステム管理者を対象読者とし、CLUSTERPRO のメンテナンス関連情報を記載します。

**『CLUSTERPRO X ハードウェア連携ガイド』** (Hardware Feature Guide)
管理者、および CLUSTERPRO を使用したクラスタシステムの導入を行うシステムエンジニアを対象読者とし、特定ハードウェアと連携する機能について記載します。『CLUSTERPRO X インストール&設定ガイド』を補完する役割を持ちます。

**『CLUSTERPRO X 互換機能ガイド』** (Legacy Feature Guide)
管理者、および CLUSTERPRO を使用したクラスタシステムの導入を行うシステムエンジニアを対象読者とし、CLUSTERPRO X 4.0 WebManager および Builder に関する情報について記載します。

## 最新情報の入手先

最新の製品情報については、以下のWebサイトを参照してください。

https://jpn.nec.com/clusterpro

# 第 1 章　　SQL Server

## 機能概要

SQL Server を CLUSTERPRO 環境下で利用する際の機能概要について以下に記述します。
CLUSTERPRO 環境下での SQL Server の運用は、片方向スタンバイ型と双方向スタンバイ型があります。
クライアントは、通常、ODBC などを使用して現用系にアクセスします。現用系に障害が発生した場合、クライアントは待機系に接続し、運用することになります。（双方向スタンバイ型ではそれぞれが現用系、待機系となります。）

### 1. 片方向スタンバイ型

右図は、サーバ 1 を現用系、サーバ 2 を待機系とした片方向スタンバイ型の CLUSTERPRO 環境を構成して動作させるときのイメージ図です。

クライアントからは、フローティング IP アドレスを使用して、ODBC などにより接続します。

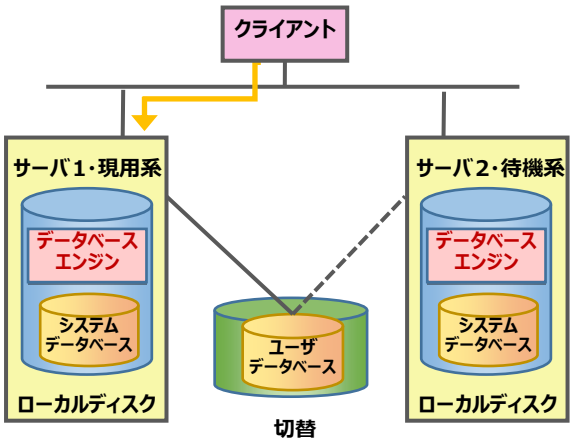


図 1.1

---

サーバ 1 に障害が発生すると右図のようになります。

フェイルオーバが完了すると、サーバ 2 上で SQL Server のサービスが立ち上がり、切替パーティションのリソースがサーバ 2 へ移行するため、クライアントはサーバ 2 へ接続し、運用することになります。
フローティング IP アドレスにてサーバへ接続をしている場合は、フェイルオーバにてフローティング IP アドレスがサーバ 2 へ移行するため、クライアントはサーバが切り替わったことを意識せずに、同一の IP アドレスで再接続することが可能です。

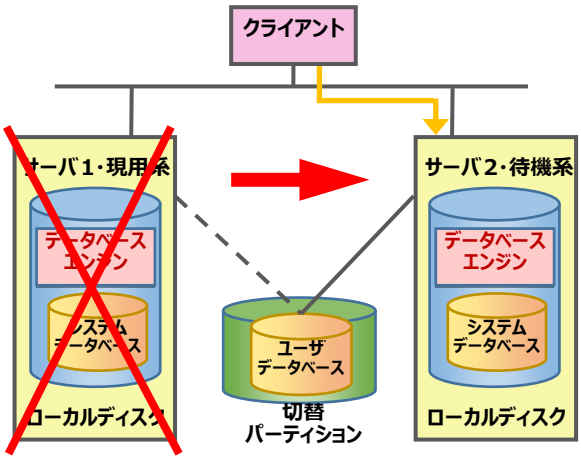


図 1.2

## 2. 双方向スタンバイ型

右図は、双方向スタンバイ型をCLUSTERPRO 環境下で動作させるときのイメージ図です。

双方向スタンバイ型の場合は以下のように構成します。

- サーバ 1 を現用系、サーバ 2 を待機系とするクラスタ グループを作成する。(右図の場合、切替パーティション 1 を使用します。)
- サーバ 2 を現用系、サーバ1を待機系とするクラスタ グループを作成します。(右図の場合、切替パーティション 2 を使用します。)

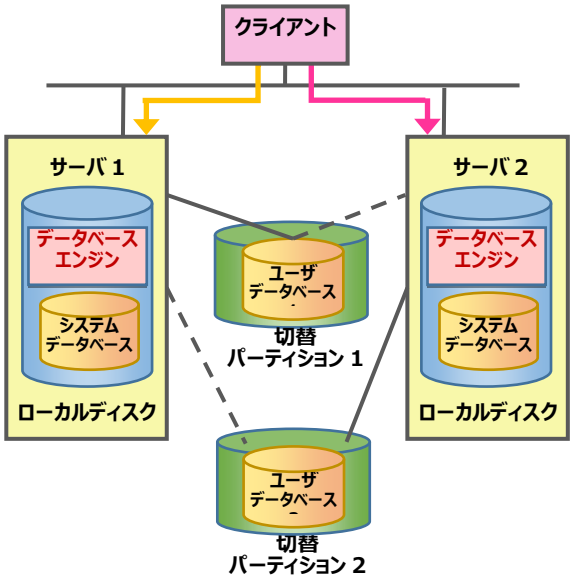


図 2.1

---

サーバ 1 に障害が発生すると、右図のようになります。

フェイルオーバが発生すると、サーバ 1 の切替パーティションのリソースがサーバ 2 に移行します。この時、サーバ2のSQL Server は 2 つのクラスタグループのユーザデータベースを持つことになります。サーバ 1（ユーザデータベース 1 ）にアクセスしていたクライアントは、サーバ 2 へ接続し、運用することになります。
フローティング IP アドレスにてサーバへ接続している場合は、フェイルオーバにてフローティング IP アドレスがサーバ 2 へ移行する為、クライアントはサーバが切り替わったことを意識せずに、同一の IP アドレスで再接続することが可能です。

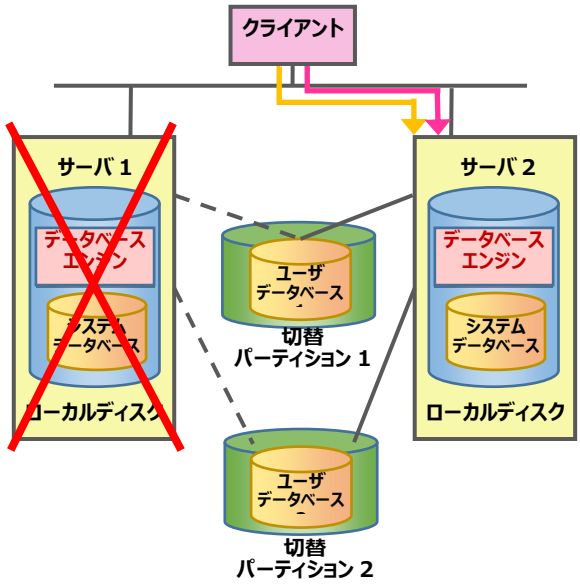


図 2.2

# 機能範囲

- CLUSTERPRO 環境で SQL Server を利用する場合、システムデータベース（master、msdb など）は、それぞれのノードのローカルディスク上に格納する必要があります。切替パーティション上にシステムデータベースを配置することはできません。
- システムデータベースで管理される情報（ログインやジョブ情報等）はフェイルオーバにより待機系サーバへ引き継がれません。

# 構築手順

SQL Server の CLUSTERPRO 環境構築は以下の流れで行います。

1. フェイルオーバグループの作成
2. SQL Server のインストール
3. ユーザデータベースの作成
4. SQL Server のスクリプト作成
5. CLUSTERPRO への SQL Server サービスの組み込み
6. 監視リソースの設定
7. ログイン情報の引継ぎ
8. 暗号化設定の引継ぎ

## 1. フェイルオーバグループの作成

CLUSTERPRO でフェイルオーバグループを作成します。フェイルオーバグループには、以下のリソースが必要です。

- フローティング IP アドレス
- 切替パーティション（ユーザデータベースファイルを格納する十分な容量をもったもの）

## 2. SQL Server のインストール

各サーバのローカルディスク上に SQL Server をインストールします。
SQL Server セットアップ時にデータディレクトリに指定するフォルダは、必ずローカルディスクを指定するようにしてください。
また、SQL Server サービスの開始モードは「DISABLE」に設定（自動起動を行わないように構成）します。

# 3. ユーザデータベースの作成

## [1] 現用系での作業

フェイルオーバ対象となるユーザデータベースの作成は、現用系から行います。
ユーザデータベースは、切替パーティション上に作成します。
以下の例では、切替パーティション上 （ここではマウントポイントを「/mnt/sdb2」に設定） に、TESTDB という名前のデータベース （データファイル初期サイズ 10MB、ログファイル初期サイズ 10MB） を作成しています。

**【データベース作成例】**

以下のクエリを sqlcmd から実行します。

```
/* TESTDB_Data、TESTDB_Log の2つのファイルから TESTDB という DBを作成 */
create database TESTDB on PRIMARY (
   name = 'TESTDB_Data',
   filename = '/mnt/sdb2/mssql/data/TESTDB_Data.mdf',
   size = 10
)
LOG ON (
   name = 'TESTDB_Log',
   filename = '/mnt/sdb2/mssql/data/TESTDB_Log.ldf',
   size = 10
)
go
CHECKPOINT
go
```

※「透過的データ暗号化」機能を使用して対象のデータベースの暗号化を行いたい場合、ここではまだ暗号化設定を行わないようにします。暗号化設定を行う手順については、後述の「8. 暗号化設定の引き継ぎ」にて記載しております。

データベースは、Windows 端末から Linux サーバに SQL Server Management Studio（以降、SSMS と表記）で接続して GUI 操作から作成することもできます。データファイルとログファイルを切替パーティション上に作成する以外は、通常のデータベース作成と違いはありません。

なお、双方向スタンバイ型の構成の場合、2 台のサーバでそれぞれユーザデータベースを作成する必要がありますが、データベース ID （dbid） を現用系と待機系で一致させる運用とすることを推奨しています。（注1）
たとえば、サーバ 1 を現用系とするフェイルオーバ グループのユーザデータベースとして db1、サーバ 2 を現用系とするフェイルオーバ グループのユーザデータベースとして db2 を作成する状況を考えます。以下は、サーバ 1 で db1 を作成した際の dbid が 7 となる場合の作成例となります。（注2）

1. サーバ 1 で db1 を作成 (dbid = 7)
2. サーバ 2 でダミーのデータベースを作成 (dbid = 7)
3. サーバ 2 で db2 を作成 (dbid = 8)
4. サーバ 2 でダミーのデータベースを削除

（注1）フェイルオーバにより待機系へ切り替わった際にも、現用系と同じ dbid となるよう構成することを目的としています。
データベースの dbid は、以下のクエリを実行することで確認できます。以下のクエリの実行結果から、対象データベースの [dbid] 列の値を確認します。

```
exec sp_helpdb
go
```

（注2）dbid 1 ～ 6 に割り当てられているデータベースがフェイルオーバ対象のデータベースではない（デタッチ/アタッチが行われない）データベースであることが前提となります。

## [2] 待機系での作業

待機系では、データベースの作成を行う必要はありません。

# 4. SQL Server のスクリプト作成

CLUSTERPRO によるフェイルオーバ、およびフェイルバックが行われる際には、対象となるユーザデータベース(フェイルオーバ データベース)のデタッチ／アタッチが必要となります。
以下は、アタッチを行うスクリプト（ACT.SQL）とデタッチを行うスクリプト（DEACT.SQL）の記述例となります。作成した各スクリプトを、各ノードの任意のディレクトリに格納します。(注3)

(注3) ローカルディスクに格納します。切替パーティション上(共有ディスク、ミラーディスク)には格納しないでください。

**[A] 片方向スタンバイ型**

- フェイルオーバデータベースが複数存在している場合は、そのそれぞれについて "create database for attach"／"sp_detach_db" を実行する必要があります。
- ACT.SQL

```
create database [<サーバ1上フェイルオーバデータベース名>] on
    (filename = '<物理ファイル名>'),
    (filename = '<物理ファイル名>')
for attach
```

例)『ユーザデータベースの作成』で作成した TESTDB を使用する場合

```
create database TESTDB on
        (filename = '/mnt/sdb2/mssql/data/TESTDB_Data.mdf'),
        (filename = '/mnt/sdb2/mssql/data/TESTDB_Log.ldf')
for attach
```

- DEACT.SQL

```
alter database [<サーバ1上フェイルオーバデータベース名>] set offline
with ROLLBACK IMMEDIATE
exec sp_detach_db '<サーバ 1 上フェイルオーバデータベース名>',TRUE
```

例)『ユーザデータベースの作成』で作成した TESTDB を使用する場合

```
alter database [TESTDB] set offline with ROLLBACK IMMEDIATE
exec sp_detach_db 'TESTDB',TRUE
```

**[B] 双方向スタンバイ型**

- フェイルオーバグループごとに ACT.SQL と DEACT.SQL を作成する必要があります。
- フェイルオーバデータベースが複数存在している場合は、そのそれぞれについて “create database for attach”／"sp_detach_db" を実行する必要があります。
- ACT1.SQL

```
create database [<サーバ1上フェイルオーバデータベース名>] on
    (filename = '<物理ファイル名>'),
    (filename = '<物理ファイル名>')
for attach
```

- DEACT1.SQL

```
alter database [<サーバ1上のフェイルオーバデータベース名>] set offline
with ROLLBACK IMMEDIATE
exec sp_detach_db '<サーバ 1 上のフェイルオーバデータベース名>',TRUE
```

- ACT2.SQL

```
create database [<サーバ 2 上フェイルオーバデータベース名>] on
    (filename = '<物理ファイル名>'),
    (filename = '<物理ファイル名>')
for attach
```

- DEACT2.SQL

```
alter database [<サーバ 2 上のフェイルオーバデータベース名>] set offline
with ROLLBACK IMMEDIATE
exec sp_detach_db '<サーバ 2 上のフェイルオーバデータベース名>',TRUE
```

# 5. CLUSTERPRO への SQL Server サービスの組み込み

SQL Server サービスの起動制御を CLUSTERPRO から行う様に設定します。

SQL Server サービスの起動をスクリプトリソースで行う方法を記載します。

※『4. SQL Server のスクリプト作成』で作成したスクリプトファイル（ACT.SQL／DEACT.SQL または ACT1.SQL／DEACT1.SQL／ACT2.SQL／DEACT2.SQL）を、ローカルディスク上の任意のディレクトリへ格納します。
以下の CLUSTERPRO スクリプトの記述例では、スクリプトファイルの格納先を /mssql としています。

## 【方法】SQL Server サービスの起動をスクリプトリソースで制御する

スクリプトリソースでは、実行時の状況に応じて処理を変更できるように、環境変数に実行状況を示す値が設定される構成となっています。デフォルトで作成されるテンプレートに、これらの環境変数の値による条件分岐が用意されています。
以下の開始／終了処理がそれぞれの環境変数の値に応じて実行されるように構成します。

以下に、スクリプトリソースの記載例を示します。

### [A] 片方向スタンバイ型

- 開始スクリプト例
  - $CLP_EVENT が「START」または「FAILOVER」
    $CLP_SERVER が「HOME」（「OTHER」ではない）

```
systemctl start mssql-server
sleep 60
su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/ACT.SQL -o /mssql/ACT.LOG
-S .'
```

  - $CLP_EVENT が「START」または「FAILOVER」
    $CLP_SERVER が「OTHER」

```
systemctl start mssql-server
sleep 60
su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/ACT.SQL -o /mssql/ACT.LOG
-S .'
```

  - sleep コマンドは、SQL Server サービスの起動後におこなわれる自動復旧処理を考慮したものです。自動復旧処理では、コミットされていないトランザクションのロールバック処理や、コミットされているがディスクへの書き込みが行われていなかったデータを反映するためのロールフォワード処理等が行われます。基本的には 60 秒程度で問題ありませんが、検証の上、判断する必要があります。

※ 次の例は、上記の記述を行った状態の開始スクリプト（start.sh）です。環境に応じて適宜修正してください。また、必要に応じて、異常発生時の中断／リカバリ処理をスクリプト内に追記してください。

start.shの編集例

```
#! /bin/sh
#****************************************
#*                  start.sh            *
#****************************************

#ulimit -s unlimited

if [ "$CLP_EVENT" = "START" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
        echo "NORMAL1"
        if [ "$CLP_SERVER" = "HOME" ]
        then
            echo "NORMAL2"
                systemctl start mssql-server
                sleep 60
                su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/ACT.SQL -o
/mssql/ACT.LOG -S .'
        else
            echo "ON_OTHER1"
                systemctl start mssql-server
                sleep 60
                su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/ACT.SQL -o
/mssql/ACT.LOG -S .'
        fi
    else
        echo "ERROR_DISK from START"
        exit 1
    fi
elif [ "$CLP_EVENT" = "FAILOVER" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
        echo "FAILOVER1"
        if [ "$CLP_SERVER" = "HOME" ]
        then
            echo "FAILOVER2"
                systemctl start mssql-server
                sleep 60
                su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/ACT.SQL -o
/mssql/ACT.LOG -S .'
        else
            echo "ON_OTHER2"
                systemctl start mssql-server
                sleep 60
                su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/ACT.SQL -o
/mssql/ACT.LOG -S .'
        fi
    else
        echo "ERROR_DISK from FAILOVER"
        exit 1
    fi
else
    echo "NO_CLP"
    exit 1
fi
echo "EXIT"
exit 0
```

◆ 終了スクリプト例

◇ $CLP_EVENT が「START」または「FAILOVER」
$CLP_SERVER が「HOME」（「OTHER」ではない）

```
su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/DEACT.SQL -o
/mssql/DEACT.LOG -S .'
systemctl stop mssql-server
sleep 10
```

◇ $CLP_EVENT が「START」または「FAILOVER」
$CLP_SERVER が「OTHER」

```
su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/DEACT.SQL -o
/mssql/DEACT.LOG -S .'
systemctl stop mssql-server
sleep 10
```

◇ sleep コマンドは、SQL Server サービスの停止時にキャッシュ上の情報がディスクに書き込まれるのを待ち合わせるためのものとなります。
引数に指定する待ち合わせ時間はサービス停止時にディスクにフラッシュされていない情報量に依存します。基本的には 10 秒程度で問題ありませんが、検証の上、判断する必要があります。

※ 次の例は、上記の記述を行った状態の終了スクリプト（stop.sh）です。環境に応じて適宜修正してください。また、必要に応じて、異常発生時の中断／リカバリ処理をスクリプト内に追記してください。

stop.shの編集例

```
#! /bin/sh
#****************************************
#*                  stop.sh                  *
#****************************************

#ulimit -s unlimited

if [ "$CLP_EVENT" = "START" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
        echo "NORMAL1"
        if [ "$CLP_SERVER" = "HOME" ]
        then
            echo "NORMAL2"
                su -l mssql -c 'sqlcmd -U sa -P <パスワード>  -i /mssql/DEACT.SQL
-o /mssql/DEACT.LOG -S .'
                systemctl stop mssql-server
                sleep 10
        else
            echo "ON_OTHER1"
                su -l mssql -c 'sqlcmd -U sa -P <パスワード>  -i /mssql/DEACT.SQL
-o /mssql/DEACT.LOG -S .'
                sleep 10
        fi
    else
        echo "ERROR_DISK from START"
        exit 1
    fi
elif [ "$CLP_EVENT" = "FAILOVER" ]
then
    if [ "$CLP_DISK" = "SUCCESS" ]
    then
        echo "FAILOVER1"
        if [ "$CLP_SERVER" = "HOME" ]
        then
            echo "FAILOVER2"
                su -l mssql -c 'sqlcmd -U sa -P <パスワード>  -i /mssql/DEACT.SQL
-o /mssql/DEACT.LOG -S .'
                systemctl stop mssql-server
                sleep 10
        else
            echo "ON_OTHER2"
                su -l mssql -c 'sqlcmd -U sa -P <パスワード>  -i /mssql/DEACT.SQL
-o /mssql/DEACT.LOG -S .'
                sleep 10
        fi
    else
        echo "ERROR_DISK from FAILOVER"
        exit 1
    fi
else
    echo "NO_CLP"
    exit 1
fi
echo "EXIT"
exit 0
```

**[B] 双方向スタンバイ型**

◆ 開始スクリプト例

◇ $CLP_EVENT が「START」または「FAILOVER」
$CLP_SERVER が「HOME」（「OTHER」ではない）

```
systemctl start mssql-server
sleep 60
su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/ACT1.SQL -o /mssql/ACT1.LOG
-S .'
```

◇ $CLP_EVENT が「START」または「FAILOVER」
$CLP_SERVER が「OTHER」

```
su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/ACT1.SQL -o /mssql/ACT1.LOG
-S .'
```

◆ 終了スクリプト例

◇ $CLP_EVENT が「START」または「FAILOVER」
$CLP_SERVER が「HOME」（「OTHER」ではない）

```
su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/DEACT1.SQL -o
/mssql/DEACT1.LOG -S .'
systemctl stop mssql-server
sleep 10
```

◇ $CLP_EVENT が「START」または「FAILOVER」
$CLP_SERVER が「OTHER」

```
su -l mssql -c 'sqlcmd -U sa -P <パスワード> -i /mssql/DEACT1.SQL -o
/mssql/DEACT1.LOG -S .'
```

◆ サーバ 2 を現用系とするフェイルオーバグループのスクリプトにも、同様に上記の処理を追加してください。
なお、サーバ 2 のスクリプトを作成する際は、上記の例を以下のように読み替えてください。

- ACT1.SQL → ACT2.SQL
- ACT1.LOG → ACT2.LOG
- DEACT1.SQL → DEACT2.SQL
- DEACT1.LOG → DEACT2.LOG

## 6. 監視リソースの設定

フェイルオーバ対象とするデータベースを監視するため、CLUSTERPRO 上で SQL Serverモニタリソースを設定します。SQL Server モニタリソースの詳細や設定方法は、「CLUSTERPRO X リファレンスガイド」の SQL Server モニタリソースの箇所を参照してください。

# 7. ログイン情報の引き継ぎ

現用系側で SQL Server 認証ログインを作成した場合、待機系側へのフェイルオーバ後に当該ログインを有効にするには、以下のいずれかの方法を実施する必要があります。
sa については SID が固定のため、本対処は不要です。

【方法1】現用系と待機系で同じログインを作成する
【方法2】包含データベースを使用する

【方法1】では稼動系と待機系それぞれで、同名かつ同一 SID のログインを作成する必要があります。【方法2】ではフェイルオーバ対象の包含データベースにログインを作成するため、各データベースにログインを作成する必要があります。このため運用にあわせて対処方法を選択してください。

各方法の詳細手順を以下に記載しています。

**【方法 1】現用系と待機系で同じログインを作成する**

手順1）
現用系側でログインを作成します。
ここでは、ログイン名を 「TestLogin」、パスワードを 「PassWord」、既定のデータベースを「TESTDB」としてログインを作成する例を示します。

```
create login TestLogin with password = 'PassWord', default_database = TESTDB
```

手順2）
手順1） で作成したログインの SID を記録します。この SID は、待機系側で同一のログインを作成するために必要となります。
ログインの SID は以下のクエリを実行することで確認することができます。

```
select SUSER_SID('TestLogin')
```

手順3）
現用系側にて、フェイルオーバ対象のデータベース上にユーザを作成します。
ここでは、手順1） で作成したログイン「TestLogin」に対するユーザ「TestUser」をデータベース「TESTDB」に作成する例を示します。

```
use TESTDB
go
create user TestUser for login TestLogin
go
```

手順4）
対象のデータベースが存在するフェイルオーバグループを待機系側へフェイルオーバします。
フェイルオーバ完了後、待機系側で対象のデータベースへアクセスできることを確認してください。

手順5）

待機系側にて、現用系側と同一のログインを作成します。

ここでは、手順1） で作成したログイン「TestLogin」と同一のログインを作成する例を示します。

```
create login TestLogin with password = 'PassWord',
    SID = 0x16EABE7E1CD9D3119FE90000C019B6FD, default_database = TESTDB
```

※ 上記 create login ステートメントの第 2 引数は、ログインの SID を示します。「0x16EABE7E1CD9D3119FE90000C019B6FD」と記載している箇所については、手順2） で確認した SID に置き換えて実行してください。

上記の通り、同一のログインを作成するには、ログインの SID を一致させてログインを作成する必要があります。SID 以外の項目の設定が同じであっても、SID が一致していない場合は異なるログインと認識されます。

**【方法 2】包含データベースを使用する**

手順1）

現用系側で包含データベースを有効化します。

```
sp_configure 'contained database authentication', 1
reconfigure
go
```

手順2）

対象の SQL Server 用のフェイルオーバグループを待機系側へフェイルオーバします。

手順3）

待機系側で包含データベースを有効化します。

```
sp_configure 'contained database authentication', 1
reconfigure
go
```

手順4）

対象の SQL Server 用のフェイルオーバグループを現用系側へフェイルバックします。

手順5）

現用系側で、フェイルオーバ対象のデータベースを部分的包含に設定します。
ここでは、データベース「TESTDB」 に対して設定する例を示します。

```
use master
go
alter database TESTDB set containment = partial
go
```

手順6）

現用系側でフェイルオーバ対象のデータベース上に包含データベース ユーザを作成します。
ここでは、ユーザ名 「TestUser」、パスワード「PassWord」としてデータベース「TESTDB」上にユーザを作成する例を示します。

```
use TESTDB
go
create user TestUser with password = 'PassWord'
go
```

手順7）

対象の SQL Server 用のフェイルオーバグループを待機系側へフェイルオーバします。
正常にデータベースがアタッチされ、手順6） で作成したユーザがログインできることを確認します。

# 8. 暗号化設定の引き継ぎ

「透過的なデータ暗号化」 機能を使用する場合、現用系と待機系で同じサーバ証明書が作成されている必要があります。
待機系側に現用系と同じサーバ証明書が存在していない状態でフェイルオーバが発生すると、エラー 33111 が発生してデータベースのアタッチに失敗します。

現用系と待機系で同じサーバ証明書を作成し、正しくフェイルオーバが行われるようにデータベースを構成するには、以下の手順を実行します。

※ 「透過的なデータ暗号化」 機能を使用しない場合、本設定は不要です。

手順1）

現用系側でマスターキーを作成します。
ここでは、パスワードを「PassWord」に設定してマスターキーを作成する例を示します。

```
use master
go
create master key encryption by password = 'PassWord'
go
```

手順2）

現用系側でサーバ証明書を作成します。
ここでは、証明書名を「TestCert」、サブジェクトを「Server Certificate Test」としてサーバ証明書を作成する例を示します。

```
create certificate TestCert with subject = 'Server Certificate Test'
go
```

手順3）

現用系側でサーバ証明書をバックアップします。
ここでは、手順2） で作成したサーバ証明の秘密キーをパスワード「##pa$sS$」で暗号化して「/var/opt/mssql/data/TestCertKey」に保存し、サーバ証明書を「/var/opt/mssql/data/TestCert」へバックアップする例を示します。

```
backup certificate TestCert to file = '/var/opt/mssql/data/TestCert'
with private key (file = '/var/opt/mssql/data/TestCertKey', encryption by password =
  '##pa$sS$')
```

手順4）

対象の SQL Server 用フェイルオーバグループを、現用系から待機系へフェイルオーバします。
また、併せて手順3） でバックアップしたサーバ証明書（バックアップファイル、秘密キーファイル）を待機系側へコピーします。
ここでは、現用系側と同じディレクトリ （/var/opt/mssql/data/） 配下にサーバ証明書をコピーし、以降の手順を実施するものとします。

手順5）

待機系側でマスターキーを作成します。
マスターキーの作成時に指定するパスワードは現用系側と同じパスワードとする必要があります。
ここでは、手順1） で指定したパスワード「PassWord」を指定してマスターキーを作成する例を示します。

```
use master
go
create master key encryption by password = 'PassWord'
go
```

手順6）
待機系側で手順4） でコピーしたサーバ証明書のリストアを行います。

```
create certificate TestCert from file = '/var/opt/mssql/data/TestCert'
with private key (file = '/var/opt/mssql/data/TestCertKey',
decryption by password = '##pa$sS$')
```

（※）「decryption by password」に指定するパスワードは、手順3） で指定したパスワードと同じパスワードを指定します。

手順7）
対象の SQL Server 用フェイルオーバグループを、待機系から現用系へフェイルバックします。

手順8）
フェイルオーバ対象のデータベース上に暗号化キーを作成します。
ここでは、データベース「TESTDB」に、手順2） で作成したサーバ証明書「TestCert」と、暗号化アルゴリズム「AES_256」を使用して暗号化キーを作成する例を示します。

```
use TESTDB
go
create database encryption key with algorithm = AES_256
encryption by server certificate TestCert
go
```

（※） 暗号化アルゴリズムに指定可能な値は、以下の 4 つです。推奨値はないため、環境に応じて選択してください。
- AES_128
- AES_192
- AES_256
- TRIPLE_DES_3KEY

手順9）
フェイルオーバ対象のデータベースに対して、暗号化設定を有効化します。
ここでは、手順8） で暗号化キーを作成したデータベース「TESTDB」の暗号化設定を有効化する例を示します。

```
alter database TESTDB set encryption on
go
```

手順10）
現用系から待機系へフェイルオーバし、待機系側で正しくアタッチが行われることを確認します。

# 注意事項

## 1. CLUSTERPRO によるフェイルオーバが利用できない機能について

- システム データベース（master、msdb 等）を使用する機能は、フェイルオーバすることはできません。フェイルオーバが利用できない主な機能は以下の通りです。

  - **SQL Server 2019**
    SQL Server 2017 に記載の各機能、Machine Learning Services 等

  - **SQL Server 2017**
    自動チューニング、インメモリ OLTP、データベース スナップショット、ログ配布、レプリケーション、SQL Server Audit 監査機能等

    これらの機能はフェイルオーバ後に情報が引き継がれないため、CLUSTERPRO 環境上で正常な動作を保証することができません。このため、NEC SQL Server Response Center では、CLUSTERPRO 環境における上記の機能はサポートしていません。

## 2. SQL Server のクラスタ構成の注意事項について

- 現用系、待機系の SQL Server サービス は「DISABLE」に設定してください。（サーバ起動時に自動起動しないよう構成する必要があります。）
- SQL Server のフェイルオーバグループに登録する切替パーティション上には、フェイルオーバ対象となるデータベースのデータファイル（*.mdf、*.ndf）とトランザクションログファイル（*.ldf）のみを格納してください。これら以外のファイルを切替パーティション上に格納し、現用系と待機系で同じファイルを使用する構成はサポートされません。
- CLUSTERPRO 開始、終了スクリプトファイル（start.sh、stop.sh）内に記述する sleep コマンドのパラメータ（スリープ時間）は、システムの状態や SQL Server の状態により異なるため、実機での評価後、調整する必要があります。
- 「4. SQL Server のスクリプト作成」に記載しているクエリで使用するフェイルオーバデータベース名は、SQL Server 上で認識されているデータベース名と大文字／小文字を一致させて記述してください。
  SQL Server インスタンスレベルの照合順序の設定によっては、大文字／小文字が区別され、クエリの実行に失敗する可能性があります。
  SQL Server 上で認識されているデータベース名は、以下のクエリを実行することで確認できます。以下のクエリの実行結果から、[name] 列の値を確認します。

```
exec sp_helpdb
go
```

◆ フェイルオーバデータベースに 3 つ以上のファイルを使用する場合は、「4. SQL Server のスクリプト作成」に記載している ACT.SQL を以下のように修正してください。

```
create database '<現用系サーバ上フェイルオーバデータベース名>' on
    (filename = '<物理ファイル名>'),
    (filename = '<物理ファイル名>'),
    (filename = '<物理ファイル名>'),
    ・・・
    (filename = '<物理ファイル名>')
for attach
```

例）TESTDB データベースに、TESTDB.mdf、TESTDB_1.ndf、
TESTDB_2.ndf、TESTDB_log.ldf の 4 つのファイルが存在する場合

```
create database TESTDB on
    (filename = '/mnt/sdb2/mssql/data/TESTDB.mdf'),
    (filename = '/mnt/sdb2/mssql/data/TESTDB_1.ndf'),
    (filename = '/mnt/sdb2/mssql/data/TESTDB_2.ndf'),
    (filename = '/mnt/sdb2/mssql/data/TESTDB_log.ldf')
for attach
```

◆ フェイルオーバデータベースの物理ファイルの構成変更（*.ndf や *.ldf の追加や削除）を行う場合は、もう一方のサーバに当該データベースのエントリが残存していない状態で実施してください。たとえば、現用系側にて OS ダウンなどの障害発生により、データベースのデタッチが行われなかった場合、当該データベースのエントリが残存します。この状態で待機系側で物理ファイルの構成変更を行い、フェイルバックを行うと、現用系側で当該データベースのオープン処理が失敗します。

## 3. 片方向スタンバイ構成における注意事項について

◆ フェイルオーバ対象のデータベースが複数存在する場合、現用系と待機系で同一の dbid で登録するために、現用系の dbid 順に待機系に create database for attach を実行してください。データベースの dbid は、以下のクエリを実行することで確認できます。以下のクエリの実行結果から、対象データベースの [dbid] 列の値を確認します。

```
exec sp_helpdb
go
```

## 4. 双方向スタンバイ構成における注意事項について

◆ 双方向スタンバイ型構成において 1 つのサーバでフェイルオーバが発生した場合、まずフェイルオーバされたデータベースをフェイルバックしてください。その際に、データベース ID（dbid）を現用系と待機系で一致させる運用とすることを推奨しています。

◆ サーバ 1 のフェイルオーバデータベースの dbid が 7 の場合、サーバ 2 にて一度ダミーデータベースを dbid 7 で登録し、その状態でサーバ 2 のフェイルオーバデータベースを dbid 8 で登録してください。dbid 8 で登録後、ダミーデータベースを削除し、dbid 7 がサーバ 2 上に存在しない状態とします。また、サーバ 1 上にはdbid 8 が登録されていない状態としてください。データベースの dbid は、以下のクエリを実行することで確認できます。以下のクエリの実行結果から、対象データベースの [dbid] 列の値を確認します。

```
exec sp_helpdb
go
```

## 5. データファイル格納ディスク破損時のログ末尾のバックアップに関する注意事項について

- データベースを構成する物理ファイルのうち、データファイル（*.mdf、*ndf）を格納しているディスクが破損してデータファイルへアクセス不可となった場合、その時点でトランザクションログ末尾のバックアップを取得できれば、障害発生直前の状態まで復旧することが可能です。
- 障害発生によりフェイルオーバが発生すると、フェイルオーバ処理によって障害が発生したサーバからデータベースがデタッチされます。デタッチされた後にトランザクションログ末尾のバックアップを取得することはできませんので、フェイルオーバ発生前にトランザクションログ末尾のバックアップを取得するよう構成する必要があります。

  以下のいずれかの方法でフェイルオーバ発生前にトランザクションログ末尾のバックアップ取得を行うよう構成することが可能です。（どちらの方法でもトランザクションログバックアップを行う操作であることは同じとなりますので環境に応じてご選択ください。）
  - ◇ デタッチを行うスクリプト（DEACT.SQL）の 1 行目（alter database ステートメント）の前に、backup log ステートメント（with CONTINUE_AFTER_ERROR オプション付き）を記述する。
  - ◇ データファイル格納ディスクが破損したことを検知した際に実行可能な CLUSTERPRO スクリプトに backup log ステートメント（with CONTINUE_AFTER_ERROR オプション付き）を実行する sqlcmd コマンドを記述する。
- backup log ステートメントに CONTINUE_AFTER_ERROR オプションを指定するのは、破損により万一バックアップ時にエラーが検出された場合でも、バックアップを継続して実行することを想定しているためです。エラーが発生しない場合は、当該オプションを指定しない場合と同等の動きとなります。
- CONTINUE_AFTER_ERROR オプションを指定すれば、必ずログ末尾のバックアップが取得できるということを保証するものではありません。当該オプションを指定した状態においてもバックアップが行えない可能性もあります。
  当該オプションを指定してもバックアップが取得できない場合には、ログ末尾のバックアップを行う方法はありません。

## 6. SQL Server Agent の機能を使用する場合の注意事項について

- SQL Server Agent ジョブ、警告等は、現用系のみの設定ではフェイルオーバ後、継続して利用することができません。
  フェイルオーバ後も待機系で SQL Server Agent ジョブ、警告等を使用する場合は、現用系と待機系でそれぞれ同じ SQL Server Agent ジョブ、警告等を作成する必要があります。

### 7. ポリシーベースの管理機能を使用する場合の注意事項について

- 「ポリシーベースの管理」機能を CLUSTERPRO 環境で使用する場合、現用系で作成したポリシーを待機系へ移行することで、待機系でも問題なくポリシーを適用することが可能となります。また、現用系でポリシーの評価を実行するジョブを作成している場合においても、ポリシーの移行を行うことで併せて作成されます。(ジョブを個別に移行する必要はありません。)
  SSMS を使用した ポリシーの移行方法は以下の通りです。
  (1) SSMS から現用系に接続します。
  (2) SSMS 上で、作成したポリシーを右クリックして、[ポリシーのエクスポート] を選択し、表示されたダイアログで任意の名前と保存場所を指定して保存します。
  (3) SSMS から待機系に接続します。
  (4) SSMS 上で、[管理] － [ポリシー管理] － [ポリシー] を右クリックして、[ポリシーのインポート] を選択し、(3) でコピーしたファイルを指定します。
  (5) [OK] をクリックし、正常にポリシーが作成されたことを確認します。

### 8. 変更データキャプチャ(CDC)機能を使用する場合の注意事項について

- SQL Server 2019 で使用可能な「変更データキャプチャ(CDC)」機能を CLUSTERPRO 環境で使用する場合、切替パーティション上のユーザデータベースに対して設定することで、待機系へフェイルオーバ後も継続して使用することが可能となります。

### 9. その他の機能を使用する場合の注意事項について

- SQL Server 2017 Standard / Enterprise から SQL Server 2019 Standard / Enterprise までで追加された新機能における CLUSTERPRO 上での利用可否については、利用可否が判明次第、情報を公開いたします。

# その他

- SQL Server の各スクリプトの詳細については、SQL Server Books Online に記載されていますのでご参照願います。
- 以下の URL にマイクロソフトサポート技術情報が公開されていますので、併せてご参照ください。
  マイクロソフトサポートオンライン
  http://support.microsoft.com/default.aspx?LN=JA