CLUSTERPRO® X *for Windows*

PPガイド (SQL Server)

2018.02.02
第5版

改版履歴

版数	改版日付	内 容
1	2012/07/02	PPガイド（データベース）より分冊し、新規作成
2	2015/02/28	全面改訂（SQL Server 2014対応等）
3	2015/06/24	構築手順 ・1. 記載漏れを修正 ・2. 2点目に対象となる具体的なサービス名を追記 ・4. 記載ミスを修正・5. 記載内容の順序入れ替え、ARMLOAD／ARMKILLコマンドの記載を削除（サービスリソースによる制御を推奨のため）注意事項 ・6. 監視リソースの設定を追記
4	2016/10/14	SQL Server 2016 関連内容を追記 SQL Server 2005 対応に関する記載を変更「適用範囲」にサポート対象外のエディションに関する記載を追記 「注意事項」2. に（※2）の記載を追記
5	2018/02/02	SQL Server 2017 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、Windows、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

目次 iii
はじめに v
対象読者と目的 v
適用範囲 v
CLUSTERPRO マニュアル体系 vi
最新情報の入手先 vii

第 1 章 SQL Server 1

機能概要 1
1. 片方向スタンバイ型 1
2. 双方向スタンバイ型 2
機能範囲 3
構築手順 3
1. フェイルオーバグループの作成 4
2. SQL Server のインストール 4
3. ユーザデータベースの作成 5
4. SQL Server のスクリプト作成 7
5. CLUSTERPRO への SQL Server サービスの組み込み 9
6. 監視リソースの設定 21
7. ログイン情報の引き継ぎ 22
8. 暗号化設定の引き継ぎ 24
注意事項 26
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 26
2. SQL Server のクラスタ構成の留意点について 26
3. 片方向スタンバイ構成における留意点について 28
4. 双方向スタンバイ構成における留意点について 28
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留意点について 31
6. SQL Server Agent の機能を使用する場合の留意点について 32
7. ポリシーベースの管理機能を使用する場合の留意点について 32
8. FILESTREAM 機能を使用する場合の留意点について 32
9. 変更データキャプチャ(CDC)機能を使用する場合の留意点について 33
10. その他の機能を使用する場合の留意点について 33
その他 33

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 3.3 for Windows
CLUSTERPRO X 3.2 for Windows
CLUSTERPRO X 3.1 for Windows
CLUSTERPRO X 3.0 for Windows
CLUSTERPRO X 2.1 for Windows
CLUSTERPRO X 2.0 for Windows
CLUSTERPRO X 1.0 for Windows

SQL Server 2017 Enterprise / Standard
SQL Server 2016 Enterprise / Standard
SQL Server 2014 Enterprise / Standard
SQL Server 2012 Enterprise / Standard
SQL Server 2008 R2 Datacenter / Enterprise / Standard
SQL Server 2008 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
  2016/04以前に製品版で購入済みの場合を除く。2016/04 以降、マイクロソフト社からの無償提供に変更となったため。

CLUSTERPRO マニュアル体系

CLUSTERPRO のマニュアルは、以下の 4 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、メンテナンス関連情報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 統合WebManager 管理者ガイド』(Integrated WebManager Administrator's Guide)
CLUSTERPRO を使用したクラスタシステムを CLUSTERPRO 統合WebManager で管理するシステム管理者、および統合WebManager の導入を行うシステムエンジニアを対象読者とし、統合WebManager を使用したクラスタシステム導入時に必須の事項について、実際の手順に則して詳細を説明します。

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。

http://jpn.nec.com/clusterpro

第 1 章　　SQL Server

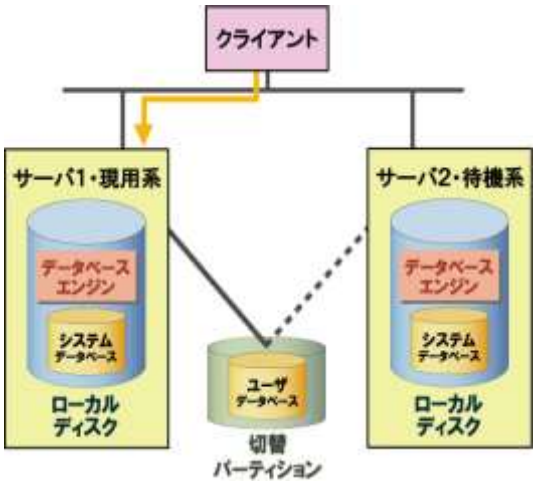
機能概要

Microsoft SQL Server 2008 以降（以下 SQL Server）を、CLUSTERPRO X 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。
なお、SQL Server のバージョンにより異なる箇所については各々説明します。
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

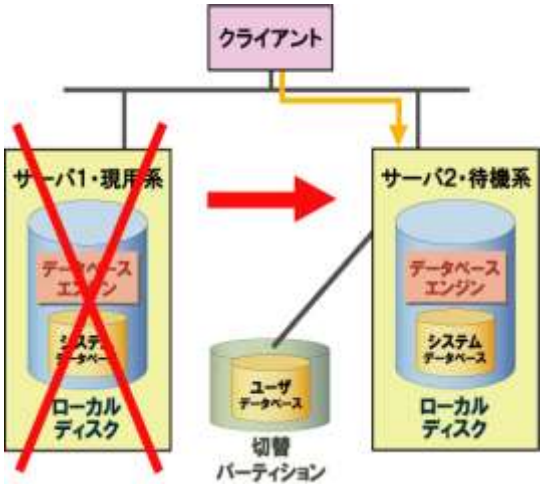
右図は、サーバ1を現用系、サーバ2を待機系とした片方向スタンバイ型のCLUSTERPRO環境を構成して動作させるときのイメージ図です。

クライアントからは、フローティングIPアドレスや仮想コンピュータ名を使用して、ODBCなどにより接続します。



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。

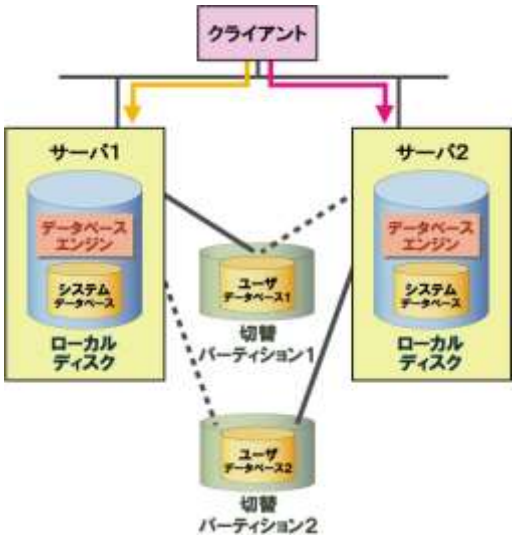


2. 双方向スタンバイ型

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。

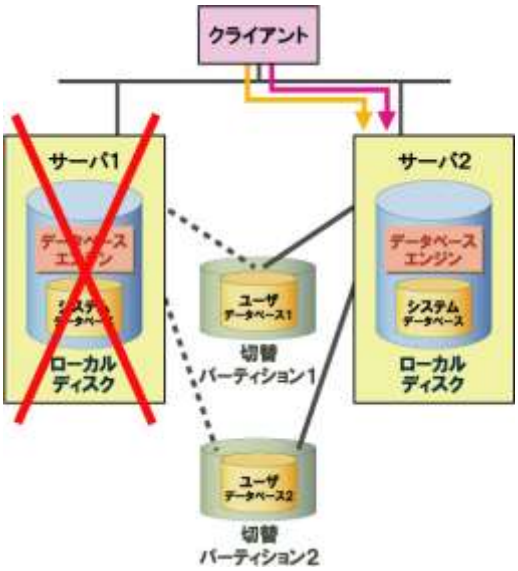
双方向スタンバイ型の場合は以下のように構成します。

- 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。)



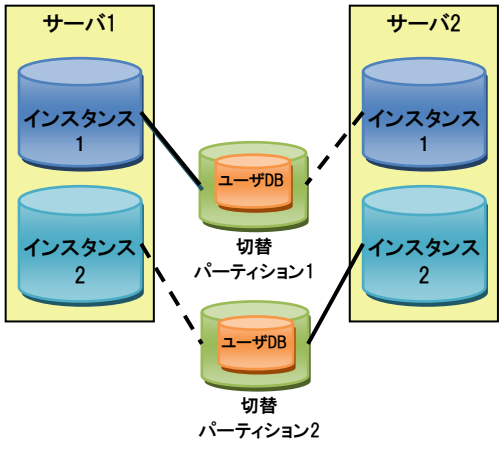
サーバ1に障害が発生すると、右図のようになります。

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。



なお、右図のようにマルチインスタンス構成において片方向スタンバイ型を組み合わせ、双方向スタンバイ型として運用することも可能です。
右図の場合は、サーバ1を現用系とする片方向スタンバイ型のインスタンスと、サーバ2を現用系とする片方向スタンバイ型のインスタンスをそれぞれ作成し、双方向スタンバイ型を実現しています。

そのため、右図のようなマルチインスタンス構成における双方向スタンバイ型を構築する場合は、各インスタンスごとに片方向スタンバイ型の作成手順に沿って構築する必要があります。



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。
- SQL Server 2008／2008 R2／2012／2014／2016／2017 は、既定インスタンス、および名前付きインスタンスでの動作を確認しております。
  （SQL Server 2000／2005 に関しては、SQL Server のPP・サポートサービス契約を締結の上、当該窓口へお問い合わせください。）

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／仮想コンピュータ名
- 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本体、およびシステム データベース ファイルを格納するフォルダは、必ずローカルディスクを指定するようにしてください。
CLUSTERPRO環境では、ユーザ データベース ファイルのみを切替パーティションに作成します。
また、SQL Server 関連のサービスの開始モードは全て「手動」に設定（自動起動を行わないように構成）します。
さらに、SQL Server サービスに関してのみ起動アカウントをCLUSTERPROサービスの起動アカウント（「LOCAL SYSTEM」）と同じアカウントに設定し、データベースのオープン処理に失敗する事象を回避します。

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（ここではドライブ文字を「Y」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SQL Server Management Studio（以降、SSMS と表記）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 = 'Y:¥sql¥data¥TESTDB_Data.mdf',
   size = 10
)
LOG ON (
   name = 'TESTDB_Log',
   filename = 'Y:¥sql¥data¥TESTDB_Log.ldf',
   size = 10
)
go
CHECKPOINT
go
```

※ SQL Server 2008 以降で使用可能な「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、SSMSによる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、各ノードの任意のフォルダに格納します。(注3)
なお、SQL Server のスクリプトは、ご利用の SQL Server バージョンによって使用するスクリプトが異なっておりますので、ご注意ください。

(注3) ローカルドライブに格納します。切替パーティション上(共有ディスク、ミラーディスク)には格納しないよう注意してください。

[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 = 'Y:¥sql¥data¥TESTDB_Data.mdf'),
    (filename = 'Y:¥sql¥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。

なお、SQL Server サービスの起動をスクリプトリソースで行う方法と、サービスリソースで行う方法があり、構築手順が一部異なりますので、注意してください。

スクリプトリソースで行う方法では CLUSTERPRO の互換コマンドを使用するため、サービスリソースで行う方法を推奨しております。

※ 『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、ローカルドライブ上の任意のフォルダへ格納します。
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 C:¥mssql としています。

【方法 1】SQL Server サービスの起動をサービスリソースで制御する場合

サービスリソースは、CLUSTERPRO でサービスの管理を行う機能です。
サービスリソース機能を使用する際にサービス監視を行う場合には、別途サービス監視リソースの設定を行う必要がありますので、ご留意ください。サービス監視リソースの設定を行う手順等の詳細については、「CLUSTERPRO X リファレンスガイド」のサービス監視リソースの箇所を参照してください。

サービスリソースでSQL Server サービスの起動を管理するには、以下2つのリソースを登録します。

① サービスリソース
「MSSQLSERVER」を登録します。（名前付きインスタンスの場合は、「MSSQL$<対象のインスタンス名>」を登録します。）

② スクリプトリソース
データベースのデタッチとアタッチを実行するためのスクリプトリソースを作成し、サービスリソースとの依存関係を設定します。

※ 各リソースの依存関係として、SQL Server では以下のように設定する必要があります。そのため、サービスリソース、スクリプトリソースでは「既存の依存関係に従う」のチェックをオフにして個別に設定を行います。
[1] サービスリソースの依存関係に、ディスクリソースを追加します
[2] スクリプトリソースの依存関係に、サービスリソースおよびディスクリソースを追加します

サービスリソースを利用する場合、スクリプトリソースでは SQL Server サービスの制御を行わないため、開始スクリプト、および終了スクリプト内に起動（net start）／停止（net stop）のコマンドを記述する必要はありません。
なお、サービスリソースを設定する際の各設定項目の詳細につきましては、「CLUSTERPRO X リファレンスガイド」のサービスリソースに関する箇所を参照してください。

また、データベースのデタッチとアタッチを実行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。

次ページに、スクリプトリソースの記載例を示します。
※以降、sqlcmd コマンド例では SQL Server 認証を利用しています。

[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）

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG
-S .¥<インスタンス名>
```

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

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG
-S .¥<インスタンス名>
```

※ 次の例は、上記の記述を行った状態の開始スクリプト（start.bat）の一部です。
（あくまで記載例となりますので、環境に応じて適宜修正してください。）

```
rem ****************************************
rem 通常起動対応処理
rem ****************************************
:NORMAL


rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK


rem **************
rem 業務通常処理
rem **************


rem プライオリティ チェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER1

rem **************
rem 最高プライオリティ での処理
rem (例) ARMBCAST /MSG "最高プライオリティサーバで起動中です" /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .

GOTO EXIT

:ON_OTHER1
rem **************
rem 最高プライオリティ 以外での処理
rem (例) ARMBCAST /MSG "プライオリティサーバ以外で起動中です" /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .

GOTO EXIT
```

```
rem ****************************************
rem フェイルオーバ対応処理
rem ****************************************
:FAILOVER

rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK


rem **************
rem フェイルオーバ後の業務起動ならびに復旧処理
rem **************


rem プライオリティ のチェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER2

rem **************
rem 最高プライオリティ での処理
rem (例) ARMBCAST /MSG "最高プライオリティサーバで起動中です（フェイルオーバ後）" /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .

GOTO EXIT

:ON_OTHER2
rem **************
rem 最高プライオリティ 以外での処理
rem (例) ARMBCAST /MSG "プライオリティサーバ以外で起動中です（フェイルオーバ後）" /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .

GOTO EXIT
```

◆ 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）

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG
-S .
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG
-S .¥<インスタンス名>
```

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

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG
-S .
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG
-S .¥<インスタンス名>
```

※ 次の例は、上記の記述を行った状態の終了スクリプト(stop.bat)の一部です。
(あくまで記載例となりますので、環境に応じて適宜修正してください。)

```
rem ****************************************
rem 通常終了対応処理
rem ****************************************
:NORMAL

rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK


rem **************
rem 業務通常処理
rem **************


rem プライオリティ チェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER1

rem **************
rem 最高プライオリティ での処理
rem (例)ARMBCAST /MSG "最高プライオリティサーバで終了中です" /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG -S .

GOTO EXIT


:ON_OTHER1
rem **************
rem 最高プライオリティ 以外での処理
rem (例)ARMBCAST /MSG "プライオリティサーバ以外で終了です" /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG -S .

GOTO EXIT
```

```
rem ****************************************
rem フェイルオーバ対応処理
rem ****************************************
:FAILOVER

rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK


rem **************
rem フェイルオーバ後の業務起動ならびに復旧処理
rem **************


rem プライオリティ のチェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER2

rem **************
rem 最高プライオリティ での処理
rem (例)ARMBCAST /MSG "最高プライオリティサーバで終了中です (フェイルオーバ後) " /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG -S .

GOTO EXIT

:ON_OTHER2
rem **************
rem 最高プライオリティ 以外での処理
rem (例)ARMBCAST /MSG "プライオリティサーバ以外で終了中です (フェイルオーバ後) " /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG -S .

GOTO EXIT
```

[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）

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT1.SQL -o C:¥mssql¥ACT1.LOG
-S .
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT1.SQL -o C:¥mssql¥ACT1.LOG
-S .¥<インスタンス名>
```

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

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT1.SQL -o C:¥mssql¥ACT1.LOG
-S .
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT1.SQL -o C:¥mssql¥ACT1.LOG
-S .¥<インスタンス名>
```

◆ 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）

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT1.SQL
-o C:¥mssql¥DEACT1.LOG -S .
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT1.SQL
-o C:¥mssql¥DEACT1.LOG -S .¥<インスタンス名>
```

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

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT1.SQL
-o C:¥mssql¥DEACT1.LOG -S .
```

【名前付きインスタンスの場合】

```
sqlcmd -Usa -P <パスワード> -i C:¥mssql¥DEACT1.SQL
-o C:¥mssql¥DEACT1.LOG -S .¥<インスタンス名>
```

◆ 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

- サービスリソース調整プロパティの［サービス］タブにて、「サービスが起動済みの場合、エラーとしない」にチェックしてください。
  双方向スタンバイ型構成の場合、フェイルオーバ先で既にSQL Server サービスが稼働中の状態となりますので、これをチェックしない場合、フェイルオーバ時にサービスリソースのアクティブ化でエラーとなり、フェイルオーバに失敗します。

【方法 2】SQL Server サービスの起動をスクリプトリソースで制御する場合

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。
スクリプトリソース内で SQL Server のサービスを監視する場合は、以下のサービス起動／停止の箇所（net start／net stop）を ARMLOAD／ARMKILL コマンドを使用するように変更してください。
但し、ARMLOADを使用してサービスを起動した場合は、/WAITオプションを使用してサービスの起動を待ち合わせてください。
ARMLOAD／ARMKILLのコマンドの詳細に関しては、「CLUSTERPRO X リファレンスガイド」を参照してください。

なお、上記に記載の ARMLOAD／ARMKILL コマンドは、CLUSTERPRO の互換コマンドであるため、可能な限り「【方法1】SQL Server サービスの起動をサービスリソースで制御する場合」に記載の方法を用いてサービス監視を行うことを推奨しております。

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）

    【既定インスタンスの場合】

    ```
    net start MSSQLSERVER
    sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .
    ```

    【名前付きインスタンスの場合】

    ```
    net start MSSQL$<インスタンス名>
    sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG
    -S .¥<インスタンス名>
    ```

  - ◇ %CLP_EVENT% が「START」または「FAILOVER」
    %CLP_SERVER% が「OTHER」

    【既定インスタンスの場合】

    ```
    net start MSSQLSERVER
    sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .
    ```

    【名前付きインスタンスの場合】

    ```
    net start MSSQL$<インスタンス名>
    sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG
    -S .¥<インスタンス名>
    ```

※ 次の例は、上記の記述を行った状態の開始スクリプト（start.bat）の一部です。
（あくまで記載例となりますので、環境に応じて適宜修正してください。）

```
rem ****************************************
rem 通常起動対応処理
rem ****************************************
:NORMAL


rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK


rem **************
rem 業務通常処理
rem **************


rem プライオリティ チェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER1

rem **************
rem 最高プライオリティ での処理
rem (例) ARMBCAST /MSG "最高プライオリティサーバで起動中です" /A
rem **************

net start MSSQLSERVER
sqlcmd -U sa -P passWORD! -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .

GOTO EXIT

:ON_OTHER1
rem **************
rem 最高プライオリティ 以外での処理
rem (例) ARMBCAST /MSG "プライオリティサーバ以外で起動中です" /A
rem **************

net start MSSQLSERVER
sqlcmd -U sa -P passWORD! -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .

GOTO EXIT
```

```
rem ****************************************
rem フェイルオーバ対応処理
rem ****************************************
:FAILOVER

rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK


rem **************
rem フェイルオーバ後の業務起動ならびに復旧処理
rem **************


rem プライオリティ のチェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER2

rem **************
rem 最高プライオリティ での処理
rem (例) ARMBCAST /MSG "最高プライオリティサーバで起動中です（フェイルオーバ後）" /A
rem **************

net start MSSQLSERVER
sqlcmd -U sa -P passWORD! -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .

GOTO EXIT

:ON_OTHER2
rem **************
rem 最高プライオリティ 以外での処理
rem (例) ARMBCAST /MSG "プライオリティサーバ以外で起動中です（フェイルオーバ後）" /A
rem **************

net start MSSQLSERVER
sqlcmd -U sa -P passWORD! -i C:¥mssql¥ACT.SQL -o C:¥mssql¥ACT.LOG -S .

GOTO EXIT
```

◆ 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）

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG
-S .
net stop MSSQLSERVER
ARMSLEEP 10
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG
-S .¥<インスタンス名>
net stop MSSQL$<インスタンス名>
ARMSLEEP 10
```

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

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG
-S .
net stop MSSQLSERVER
ARMSLEEP 10
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG
-S .¥<インスタンス名>
net stop MSSQL$<インスタンス名>
ARMSLEEP 10
```

◇ ARMSLEEP 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.bat)の一部です。
(あくまで記載例となりますので、環境に応じて適宜修正してください。)

```
rem ****************************************
rem 通常終了対応処理
rem ****************************************
:NORMAL

rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK


rem **************
rem 業務通常処理
rem **************


rem プライオリティ チェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER1

rem **************
rem 最高プライオリティ での処理
rem (例)ARMBCAST /MSG "最高プライオリティサーバで終了中です" /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG -S .
net stop MSSQLSERVER
ARMSLEEP 10

GOTO EXIT


:ON_OTHER1
rem **************
rem 最高プライオリティ 以外での処理
rem (例)ARMBCAST /MSG "プライオリティサーバ以外で終了です" /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG -S .
net stop MSSQLSERVER
ARMSLEEP 10

GOTO EXIT
```

```
rem ****************************************
rem フェイルオーバ対応処理
rem ****************************************
:FAILOVER

rem ディスクチェック
IF "%CLP_DISK%" == "FAILURE" GOTO ERROR_DISK


rem **************
rem フェイルオーバ後の業務起動ならびに復旧処理
rem **************


rem プライオリティ のチェック
IF "%CLP_SERVER%" == "OTHER" GOTO ON_OTHER2

rem **************
rem 最高プライオリティ での処理
rem (例)ARMBCAST /MSG "最高プライオリティサーバで終了中です (フェイルオーバ後) " /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG -S .
net stop MSSQLSERVER
ARMSLEEP 10

GOTO EXIT

:ON_OTHER2
rem **************
rem 最高プライオリティ 以外での処理
rem (例)ARMBCAST /MSG "プライオリティサーバ以外で終了中です (フェイルオーバ後) " /A
rem **************

sqlcmd -U sa -P passWORD! -i C:¥mssql¥DEACT.SQL -o C:¥mssql¥DEACT.LOG -S .
net stop MSSQLSERVER
ARMSLEEP 10

GOTO EXIT
```

[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）

【既定インスタンスの場合】

```
net start MSSQLSERVER
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT1.SQL -o C:¥mssql¥ACT1.LOG
-S .
```

【名前付きインスタンスの場合】

```
net start MSSQL$<インスタンス名>
sqlcmd -U sa -P<パスワード> -i C:¥mssql¥ACT1.SQL -o C:¥mssql¥ACT1.LOG
-S .¥<インスタンス名>
```

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

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT1.SQL -o C:¥mssql¥ACT1.LOG
-S .
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥ACT1.SQL -o C:¥mssql¥ACT1.LOG
-S .¥<インスタンス名>
```

◆ 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）

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT1.SQL
-o C:¥mssql¥DEACT1.LOG -S .
net stop MSSQLSERVER
ARMSLEEP 10
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT1.SQL
-o C:¥mssql¥DEACT1.LOG -S .¥<インスタンス名>
net stop MSSQL$<インスタンス名>
ARMSLEEP 10
```

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

【既定インスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT1.SQL
-o C:¥mssql¥DEACT1.LOG -S .
```

【名前付きインスタンスの場合】

```
sqlcmd -U sa -P <パスワード> -i C:¥mssql¥DEACT1.SQL
-o C:¥mssql¥DEACT1.LOG -S .¥<インスタンス名>
```

- 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 監視リソースを設定します。SQL Server 監視リソースの詳細や設定方法は、「CLSUTERPRO X リファレンスガイド」の SQL Server 監視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。

【方法1】現用系と待機系で同じログインを作成する（全てのバージョンに対応）
【方法2】包含データベースを使用する（SQL Server 2012 以降のバージョンに対応）

各方法の詳細手順を以下に記載しています。

【方法1】の手順1)、3) および【方法2】の手順1)、3)、5)、6) についてはSSMSからのGUI操作でも実施可能です。ただし、【方法1】の手順2)、4)については、クエリによる実行が必要となります。

【方法 1】現用系と待機系で同じログインを作成する（全てのバージョンに対応）

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てしまいますので、ご注意ください。

【方法 2】包含データベースを使用する（SQL Server 2012 以降のバージョンに対応）

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

SQL Server 2008 以降で使用可能な「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ので、注意してください。

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「##pasS」で暗号化して「C:¥temp¥TestCertKey」に保存し、サーバ証明書を「C:¥temp¥TestCert」へバックアップする例を示します。

```
backup certificate TestCert to file = 'C:¥temp¥TestCert'
with private key (file = 'C:¥temp¥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、現用系側と同じフォルダ（C:¥temp）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 = 'C:¥temp¥TestCert'
with private key (file = 'C:¥temp¥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 2017**
  SQL Server 2016 に記載の各機能、自動チューニング、Machine Learning Services 等
- **SQL Server 2016**
  SQL Server 2014 に記載の各機能、ライブクエリ統計、クエリストア、R Services 等
- **SQL Server 2014**
  SQL Server 2012 に記載の各機能、インメモリ OLTP、バッファプール拡張 等
- **SQL Server 2012**
  SQL Server 2008 R2 に記載の各機能、AlwaysOn 可用性グループ 等
- **SQL Server 2008 R2**
  SQL Server 2008 に記載の各機能、マルチサーバ管理、データ層アプリケーション、マスターデータ サービス 等
- **SQL Server 2008**
  Analysis Services、Reporting Services、データベース スナップショット、データベース ミラーリング、ログ配布、レプリケーション、データコレクション、パフォーマンス データコレクション、リソースガバナ、SQL Server Audit 監査機能、データ プロファイル タスク 等

2. SQL Server のクラスタ構成の留意点について

◆ 現用系、待機系の SQL Server 関連サービスのスタートアップの種類は全て「手動」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。）

◆ 現用系、待機系の SQL Server サービス（※1）の起動アカウントは、CLUSTERPROサービスの起動アカウント（「LOCAL SYSTEM」）（※2）と同じアカウントを指定してください。
CLUSTERPROサービスの起動アカウントと異なる場合、稼働中のサーバ上でデータベースのデタッチが行われない状態（OSダウンの障害が発生したとき等）でフェイルオーバすると、その後のフェイルバック時にデータベースのオープン処理が失敗する場合があります。
データベースのオープン処理に失敗すると、データベースは使用不可の状態となり、アプリケーション等の稼働に影響を及ぼすことが想定されます。

（※1）基本的に、CLUSTERPRO 環境上では SQL Server 関連サービスとして以下がサーバ上に存在している状態となります。（インストール時に指定する機能によっては、下記以外のサービスが存在する場合があります。また、「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」に記載の機能で使用するサービスは下記には含めておりません。）

SQL Server（<対象インスタンス名>）
SQL Server エージェント（<対象インスタンス名>）
SQL Server Integration Services 14.0（※3）
SQL Server Browser
SQL Server VSS Writer

CLUSTERPRO サービスの起動アカウントと同じアカウントを指定する必要のあるサー

ビスは、上記のうち「SQL Server (<対象インスタンス名>)」サービスのみです。他のサービスは任意の起動アカウントに設定して問題ありません。

（※2）**「LOCAL SYSTEM」には既定で「メモリ内のページのロック（Lock Pages in Memory）」権限が付与**されています。以下のすべての条件を満たすことで、**SQL Server が使用するメモリがページアウトされない動作**となります。

- SQL Server サービスの起動アカウントに「LOCAL SYSYEM」を設定する。
- SQL Server 2008 R2 以前かつ Standard Edition の場合、併せてトレースフラグ 845 を設定する。
  （Enterprise Edition の場合はトレースフラグの設定は不要。）

x64 環境の場合、SQL Server は既定でサーバ搭載メモリ量を上限としてメモリを使用するため、SQL Server が使用するメモリがページアウトされないことにより、OS や他のアプリケーションの稼働に影響を及ぼす可能性があります。

そのため、**必ずmax server memoryの設定を明示的に変更し、SQL Server が使用するメモリ量の上限を設定**してください。
max server memory設定を変更する方法に関しては、SQL Server Books Online（オンラインマニュアル）をご確認願います。

（※3）「SQL Server Integration Services 14.0」の数字(14.0)は、バージョンによって異なります。

- 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 終了スクリプトファイル（stop.bat）内に記述する ARMSLEEP 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 = 'Y:¥sql¥data¥TESTDB.mdf'),
    (filename = 'Y:¥sql¥data¥TESTDB_1.ndf'),
    (filename = 'Y:¥sql¥data¥TESTDB_2.ndf'),
    (filename = 'Y:¥sql¥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、現用系側で実施してください。フェイルオーバデータベースが待機系にフェイルオーバしている状態で物理ファイルの構成変更は実施しないでください。

3. 片方向スタンバイ構成における留意点について

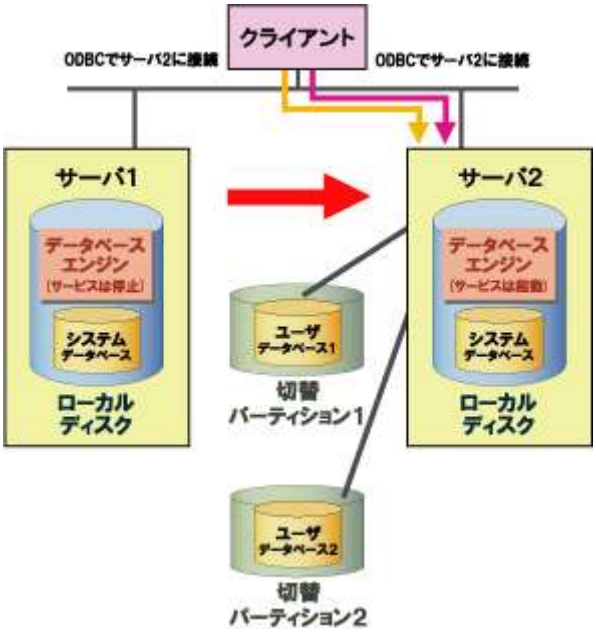
- フェイルオーバ対象のデータベースが複数存在する場合、現用系と待機系で同一の dbid で登録するために、現用系の dbid 順に待機系に create database for attach を実行してください。データベースの dbid は、以下のクエリを実行することで確認できます。以下のクエリの実行結果から、対象データベースの [dbid] 列の値を確認します。

```
exec sp_helpdb
go
```

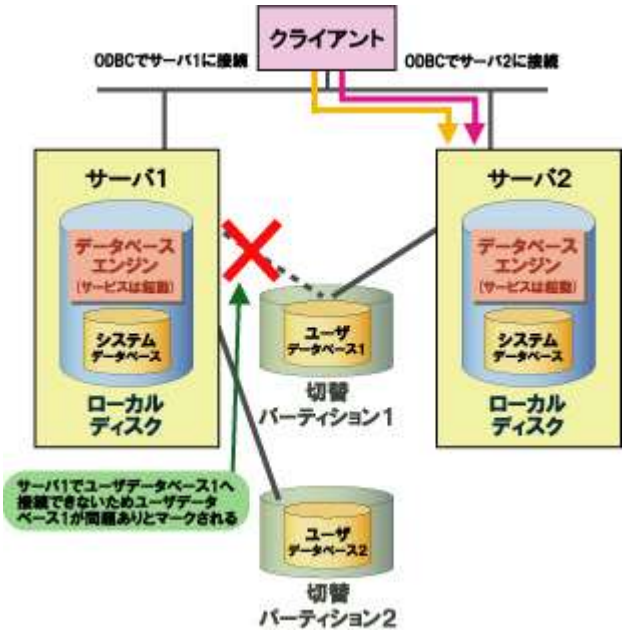
4. 双方向スタンバイ構成における留意点について

- 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。フェイルバックを実行する前にもう1つのデータベースをフェイルオーバすることはできません（以下の操作を行ってはいけません）。

① ユーザデータベース 1 をサーバ 2 へフェイルオーバ



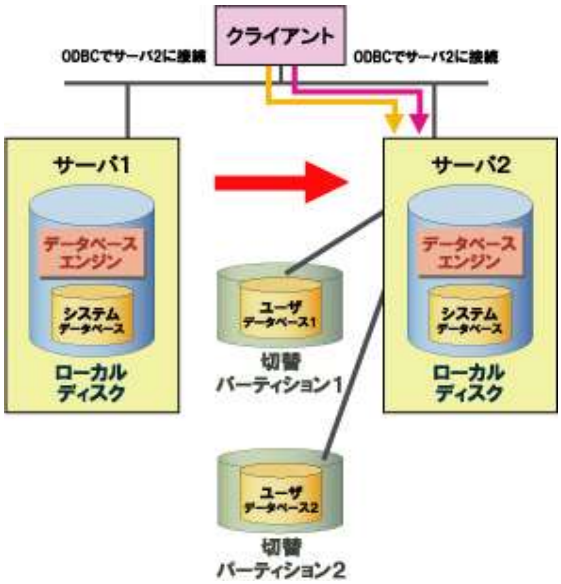
②ユーザデータベース 1 をサーバ 1 へフェイルバックを実行する前に、ユーザデータベース 2 をサーバ 1 にフェイルオーバ

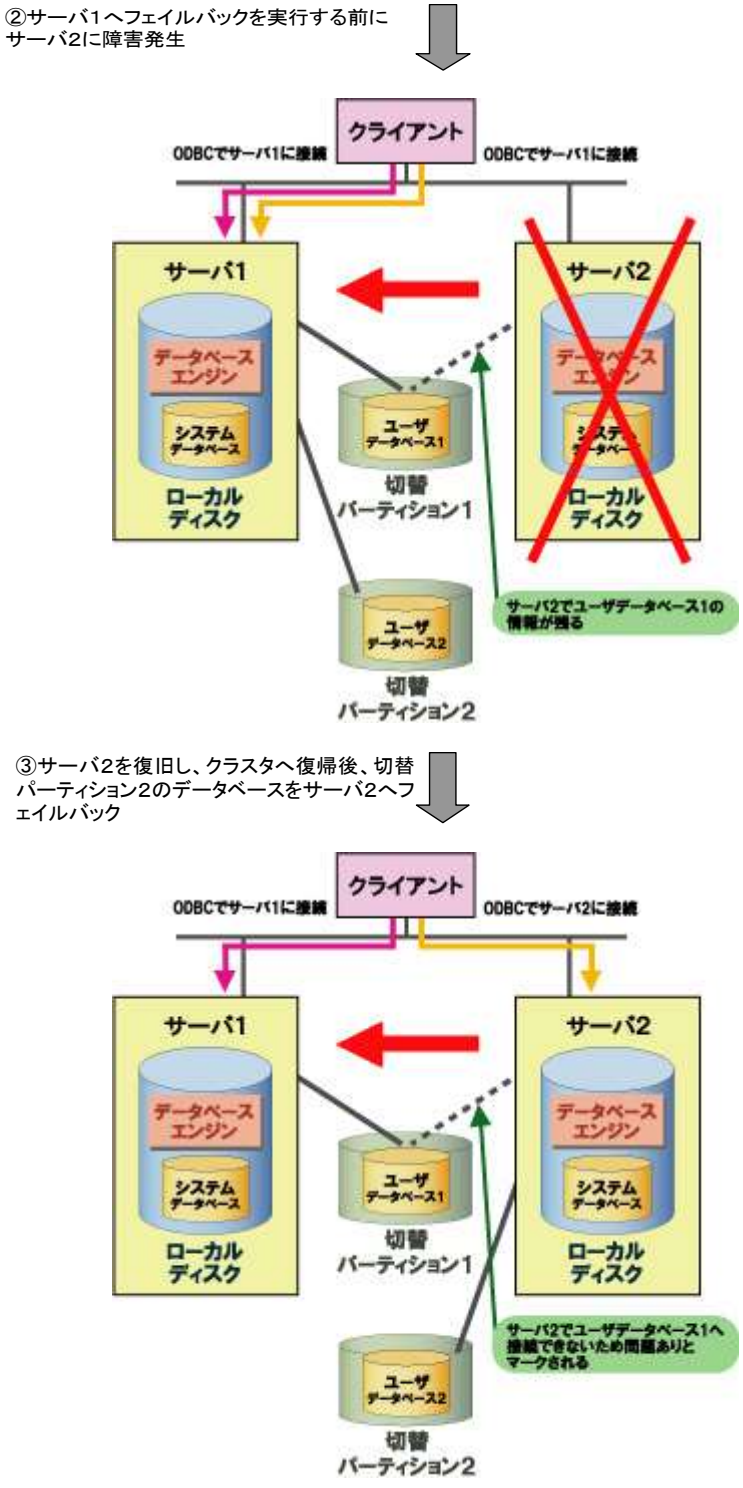


サーバ1でのユーザデータベース1、サーバ2でのユーザデータベース2は、それぞれのサーバ上の SQL Server で定義されたデータベースです。したがって、SQL Server のサービスが起動中にアクセスできない状態では問題あり(「未確認」ステータス)となり、以降アクセスできない状態となります。このような操作は運用上サポートされません。

◆ 双方向スタンバイ型で運用中、サーバ1で障害が発生し、サーバ2上で運用している(サーバ1は復旧されているが、フェイルバックは実行されていない)状態で、サーバ2に障害が発生した場合、サーバ2を起動、復旧し、SQL Server が起動される(クラスタに復帰)と、サーバ1からフェイルオーバで引き継いだデータベースが問題ありとマークされます。

① ユーザデータベース 1 をサーバ 2 へフェイルオーバ





上記図のような状況となった場合には、サーバ2上で以下のクエリを実行し、サーバ2上からユーザデータベース1の情報を削除してください。

```
use master
go
alter database <ユーザデータベース1のデータベース名> set offline
go
exec sp_detach_db '<ユーザデータベース1のデータベース名>', true
go
```

- 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留意点について

- 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留意点について

- 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。
- SQL Server Agent の機能をフェイルオーバ後も利用する場合、SQL Server Agent サービスをフェイルオーバ対象のサービスとして制御する必要がありますので、環境構成によって以下のいずれかの対応を実施する必要があります。
  - ◇ スクリプトリソースに SQL Server Agent サービスに対する net start／net stop コマンドを記述します。
  - ◇ サービスリソースに「SQLSERVERAGENT」を登録します。（名前付きインスタンスの場合は「SQLAGENT$<対象のインスタンス名>」を登録します。）
- 上記いずれかの方法により CLUSTERPRO 側でサービス制御を行う場合、SQL Server サービスを停止する処理の前に SQL Server Agent サービスが停止するよう構成してください。
  SQL Server Agent サービスを停止させずにフェイルオーバ／フェイルバックを実行すると、SQL Server サービスは、SQL Server Agent サービスの停止待ち状態となり、フェイルオーバ／フェイルバックの実行中にハングしたような状態となります。

  また、サービス起動時には SQL Server サービス起動後に SQL Server Agent サービスを起動するよう構成します。

  サービスリソースを使用している場合は、SQL Server サービスの後に SQL Server Agent サービスが起動するようサービスリソースの依存関係を設定する必要があります。この依存関係を設定しておけば終了時は逆の順番でサービスが停止されますので、停止時の順序を設定する必要はありません。
  なお、必要に応じてサービス監視リソースを設定してください。

7. ポリシーベースの管理機能を使用する場合の留意点について

- SQL Server 2008 以降で使用可能な「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。）
  ポリシーの移行方法は以下の通りです。
  (1) 現用系で SSMS を開きます。
  (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) (2)で保存したファイルを待機系へコピーします。
  (4) 待機系で SSMS を開きます。
  (5) 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。
  (6) [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. FILESTREAM 機能を使用する場合の留意点について

- SQL Server 2008 以降で使用可能な「FILESTREAM」機能を CLUSTERPRO 環境で使用す

る場合、FILESTREAM ファイルグループ（データベース作成時に指定する FILESTREAM 格納先）を切替パーティション上に指定することで、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. 変更データキャプチャ（CDC）機能を使用する場合の留意点について

- SQL Server 2008 以降で使用可能な「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。
- 本機能を使用する場合、SQL Server Agent サービスが起動されている必要がありますので、SQL Server Agent サービスを CLUSTERPRO によるサービス制御の対象としてください。
  併せて「SQL Server Agent 機能を使用する場合の留意点について」の項目もご参照ください。

10. その他の機能を使用する場合の留意点について

- SQL Server 2008 以降で使用可能なデータコレクション、パフォーマンス データコレクション、リソースガバナ、SQL Server Audit 監査機能、データ プロファイル タスクは、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これらの機能を使用したい場合は、待機系で現用系と同様の設定を作成する必要があります。
- SQL Server 2008 R2 以降で使用可能な PowerPivot、マルチサーバ管理、データ層アプリケーション、StreamInsight 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これらの機能を使用したい場合は、待機系で現用系と同様の設定を作成する必要があります。
- SQL Server 2012／2014／2016／2017 以降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