Oracle Maximum
Availability Architecture

Oracle GoldenGate Hub構成による Oracle Databaseの移行

Oracle ホワイト・ペーパー | 2019 年 8 月

目次

概要

この Oracle GoldeGate Maximum Availability Architecture（MAA）移行ソリューションは、11g Release 2 以上の Oracle データベースを設定して、どのシステム・プラットフォーム（AIX、HP UX、Linux など）からでも、単一テナントからマルチテナント・アーキテクチャに、あるいは透過的データ暗号化（TDE）を使用しないデータベースから TDE データベースに移行できるように、段階的な手順を提供します。これら同じ MAA ベスト・プラクティスは通常、オンプレミスまたはクラウドのソース・データベース・システムとターゲット・データベース・システム間の移行を対象としていますが、クラウド固有の詳細については別のホワイト・ペーパーで概説します。

この MAA ソリューションでは、以下のメリットを実現する新しい Oracle GoldenGate Hub ソリューションを使用します。

- 最小限からゼロの停止時間
- データベース・バージョン間のサポート
- プラットフォーム間のサポート
- 単一テナントまたはマルチテナントに依存しない
- ソースおよびターゲット・データベース・システムへの Oracle GoldenGate リソースの影響を軽減
- Oracle GoldenGate Hub アーキテクチャは複数の Oracle Database の移行に使用できます。

以下のダイアグラムに、このホワイト・ペーパーで紹介する移行フローを簡潔に示します。

構成の概要

この項では、Oracle GoldenGate Microservices Architecture を使った Oracle GoldenGate 構成を紹介します。

Oracle GoldenGate

Oracle GoldenGate は、同種システムと異種システムの間で、リアルタイムでのログベースのチェンジ・データ・キャプチャおよび配信機能を提供します。このテクノロジーを使用すると、低負荷で費用対効果に優れたリアルタイム・データ統合と、継続的可用性ソリューションを実現できます。

Oracle GoldenGate は、トランザクションの整合性を維持し、既存インフラストラクチャでのオーバーヘッドを最小限に抑えながら、コミットされたトランザクションからデータをレプリケートします。このアーキテクチャでは、1対多、多対多、カスケード、双方向などのさまざまなデータ・レプリケーション・トポロジがサポートされています。多様なユースケースには、リアルタイム・ビジネス・インテリジェンス、問合せのオフロード、停止時間なしのアップグレードと移行、アクティブ/アクティブ・データベースを使用したデータ分散、データ同期、高可用性が含まれます。

Oracle GoldenGate Microservices Architecture は、Oracle GoldenGate 環境の一部として REST 対応サービスを提供する新しい管理アーキテクチャとして、Oracle GoldenGate Release 12.3 で導入されました。REST 対応サービスを使用すると、HTML5 Web ページ、コマンドライン・インタフェース、API を介してリモートでの構成、管理、監視を行うことができます。図 1 に、Oracle GoldenGate Microservices Architecture を示します。

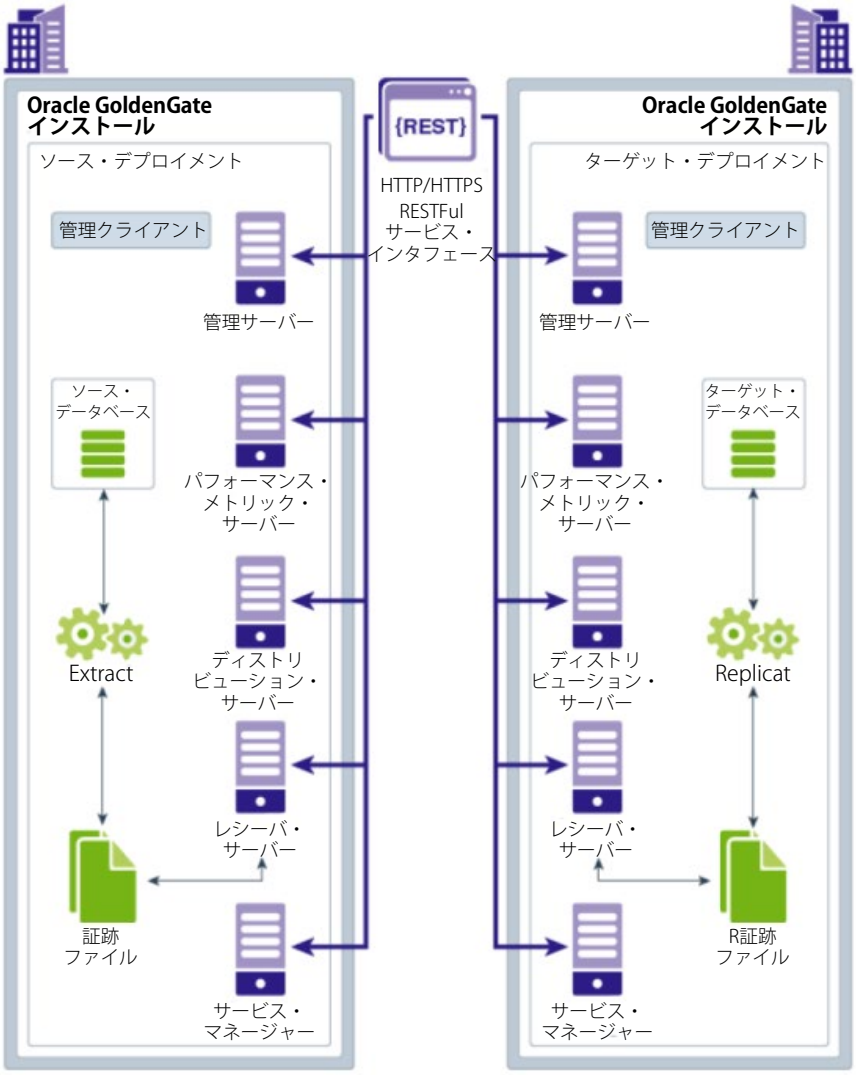


図 1：Oracle GoldenGate Microservices Architecture

Oracle GoldenGate Microservices Architecture について詳しくは、次の場所の Oracle GoldenGate ドキュメントを参照してください。

https://docs.oracle.com/en/middleware/goldengate/core/19.1/using/getting-started-oracle-goldengate.html#GUID-61088509-F951-4737-AE06-29DAEAD01C0C

Oracle GoldenGate Hub

Oracle GoldenGate Hub は、運用対象のデータベースとは異なるホストに Oracle GoldenGate ソフトウェアを配置するアーキテクチャの概念です。

ハブは、ターゲット・データベースに近いネットワークに配置する必要があるため、2 通りのアーキテクチャの構成を考えることになります。

1. **個別の GoldenGate Hub サーバー** – 図 2 に示すように、GoldenGate ソフトウェアとプロセスを個別のサーバーで稼働します。ソース・データベース・バージョンとターゲット・データベース・バージョン双方のための Oracle クライアント・ソフトウェアを GoldenGate Hub に配置する必要があります。このアーキテクチャの利点の 1 つは、GoldenGate が使用する大量のリソースがソースおよびターゲット・データベース・サーバーから分離されることです。GoldenGate Hub サーバーのもう 1 つの利点は、複数のデータベースの移行に使用できる点です。欠点は、すべての GoldenGate トラフィックに対処するために、十分なメモリ、CPU、I/O、ネットワーク帯域幅を追加サーバーに割り当てる必要があることです。

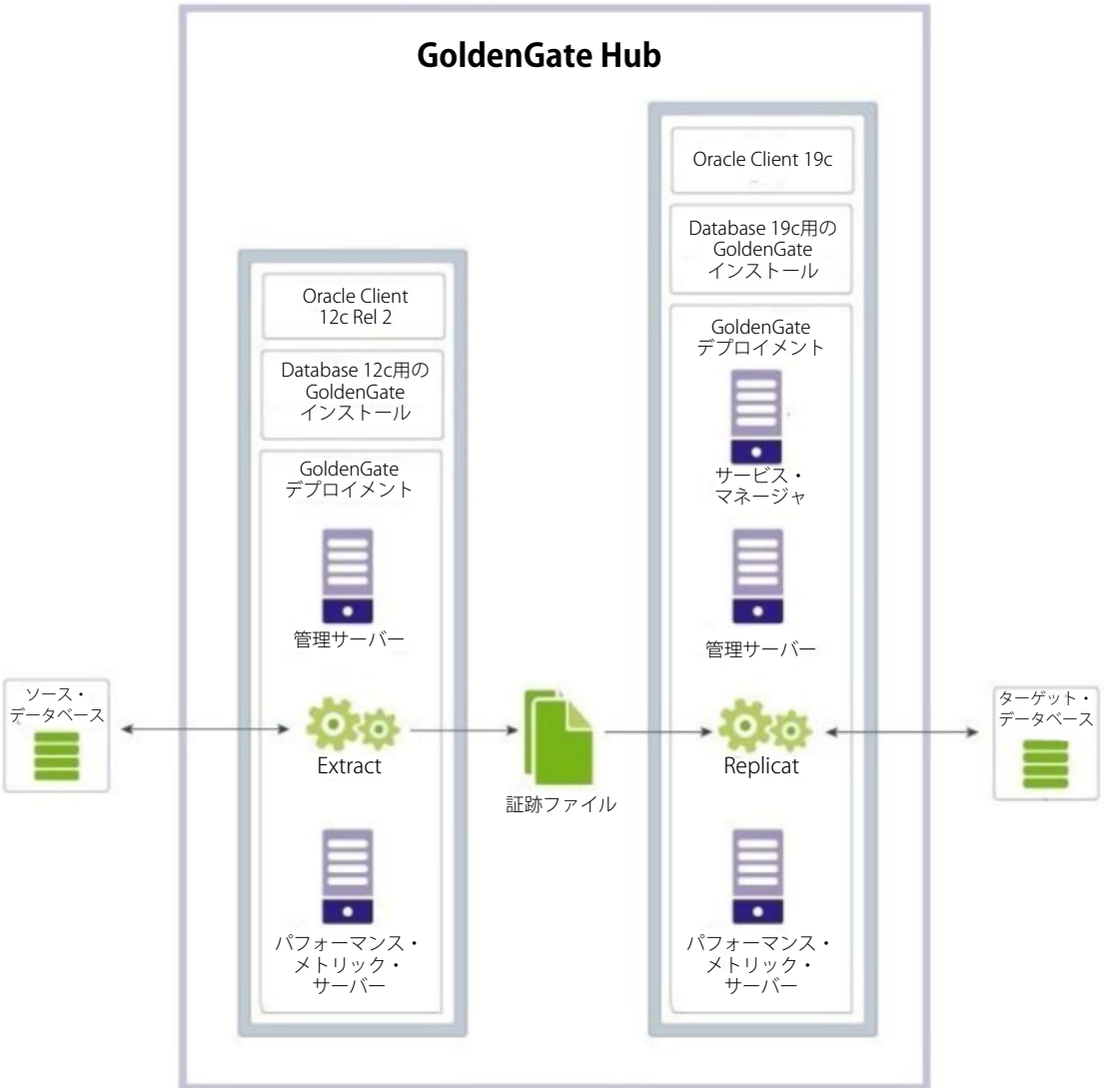


図 2：専用サーバー上で稼働する Oracle GoldenGate Hub アーキテクチャ

2. **Oracle GoldenGate Hub** をターゲット・データベース・サーバーに共同配置 – 図 3 に示すように、GoldenGate ソフトウェアとプロセスはターゲット・データベース・ホスト上で稼働します。はるかにパワフルなサーバー、または Oracle GoldenGate が使用できる豊富なリソース（CPU、メモリ、I/O）を持つサーバーにデータベースを移行する場合に、このトポロジをお奨めします。ターゲット・データベース・バージョンが異なる場合は、ソース・データベース・バージョンに一致させるために、追加の Oracle Client ソフトウェアが必要です。

このアーキテクチャの利点は、GoldenGate によるリソース使用の多くをソース・データベース・ホストから切り離し、追加の専用サーバーが不要なことです。豊富なリソースを備えた新しいサーバーに移行する場合は、このソリューションが適切です。欠点は、ターゲット・データベース・ホストに、GoldenGate とターゲット・データベース用の十分なリソースが必要なことです。

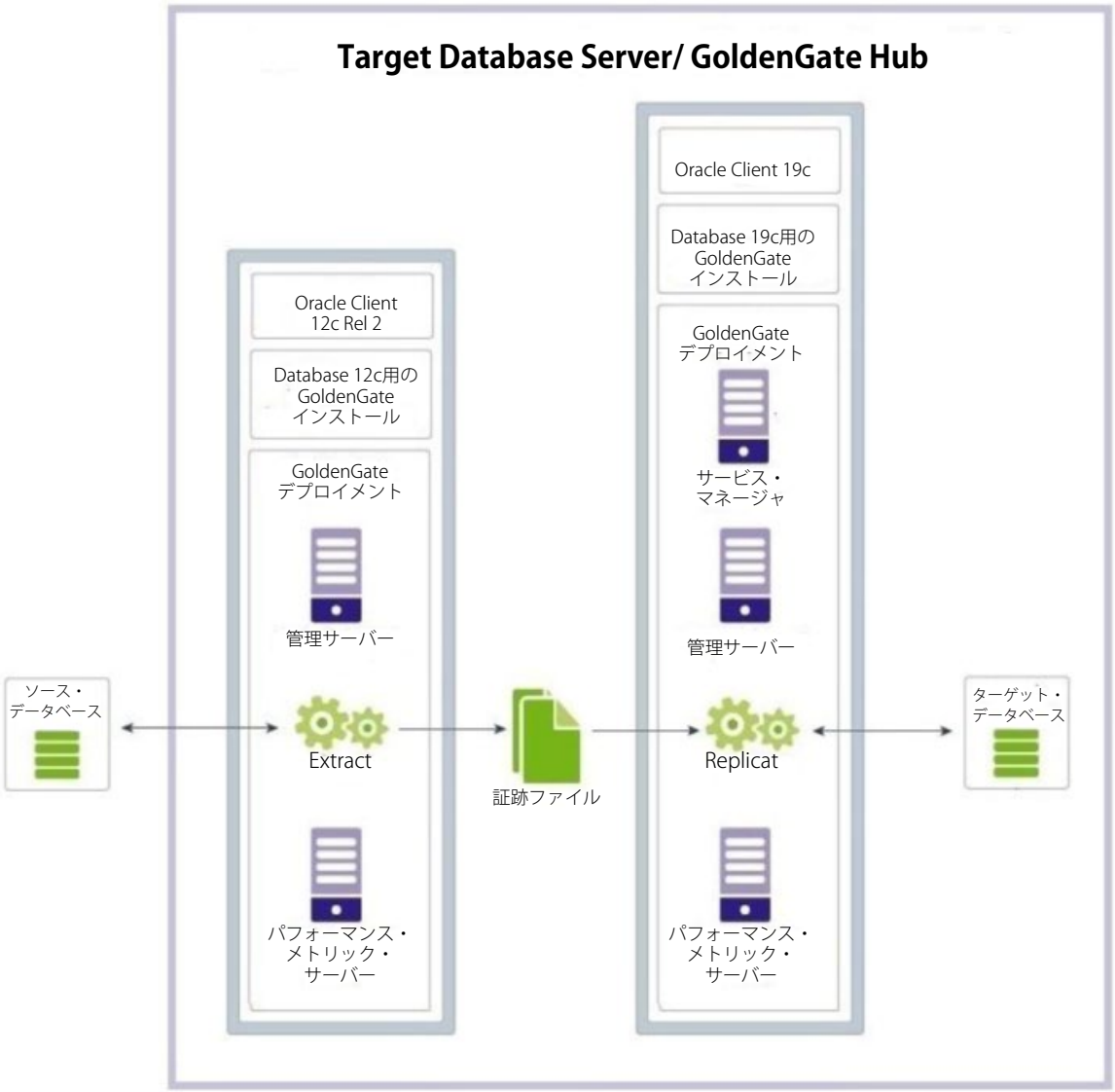


図 3：ターゲット・データベース・サーバー上で稼働する Oracle GoldenGate

このホワイト・ペーパーの手順に従う場合、Oracle GoldenGate Release 19c 以上が必要です。最新バージョンの Oracle GoldenGate は、次のサイトからダウンロードできます。

http://www.oracle.com/technetwork/middleware/goldengate/downloads/index.html

ターゲット・データベースのインスタンス化

Oracle GoldenGate でソース・データベースとターゲット・データベース間のレプリケーションを行う前に、Replicat が変更を適用したときにデータの競合が生じないように、一貫性のある状態のレプリケート済みオブジェクトを両方のデータベースに含める必要があります。ターゲット・データベースのインスタンス化の方法は、ソースおよびターゲットの Oracle データベース・バージョン、データベース・サーバーのオペレーティング・システムのエンディアンネス、およびおもなデータベース構造の相違（暗号化された、またはマルチテナント・データベースへの移行など）によって推進されます。

このホワイト・ペーパーでは、次のインスタンス化方法について説明します。

1. **RMAN 複製データベース** – Oracle のソースおよび移行先データベースのバージョンが同じで、プラットフォーム・エンディアン形式が同じである場合に、ソース・データベースを Recovery Manager によって複製できます。このインスタンス化方法は、ターゲット・データベースをインスタンス化する最速の方法です。
2. **Data Pump によるエクスポート/インポート** – ソース・データベースは、Oracle Data Pump を使ってエクスポートされ、空のターゲット・データベースにインポートされます。ソースおよび移行先 Oracle データベースのバージョンが異なる場合、プラットフォーム・エンディアン形式が異なる場合、またはデータベース構造の変更がある場合（単一テナントからマルチテナント・アーキテクチャまたは暗号化されたデータベースへの移行など）に、このインスタンス化方法が必要です。

上記のいずれの場合でも、ソース・データベースはインスタンス化の間中オンライン状態です。

インスタンス化の間、高速データベース・レプリケーションに最適な環境を実現するには、特定のデータベース操作を行わないことをお奨めします。データベースの移行中、次の操作は行わないでください。

1. **データ定義言語（DDL）** – DDL のレプリケーション中、同じオブジェクト上で DML と DDL との間にロック問題が生じないように、Replicat はデータをシリアライズします。
2. **大規模バッチ DML** – 数百万行に影響する単一のトランザクションのような大規模バッチ操作を実行すると、レプリケーション速度が遅くなる可能性があります。

2 種類のインスタンス化方法を「ターゲット・データベースのインスタンス化」の項で説明します。

このホワイト・ペーパー全体を通じて使用されるネーミング規則

このホワイト・ペーパー全体を通じて、2 つのデプロイメント（SOURCE および TARGET）を使って Oracle GoldenGate 構成を管理する REST API エンドポイントの例を示します。自動データベース移行スクリプトにコマンドを容易に組み込めるように、REST API を使用します。このスクリプトは、curl と python がインストールされた GoldenGate Hub にアクセスできる任意のサーバーからローカルまたはリモートに実行できます。または、Admin Client コマンドを使って、GoldenGate Hub を管理することもできます。Admin Client は、GoldenGate レプリケーションの作成と管理に使われるスタンドアロンのコマンドライン・インタフェースです。Admin Client コマンドについては、『Command Line Interface Reference for Oracle GoldenGate』を参照してください。

https://docs.oracle.com/en/middleware/goldengate/core/19.1/gclir/index.html

このホワイト・ペーパー全体を通じて使用され、GoldenGate REST API で指定されるおもな引数の例を以下に示します。

```
$ curl -s –K access.cfg https://<GG Hub>/<Deployment
Name>/adminsrvr/services/v2/credentials/goldengate -XGET | python -m json.tool
```

access.cfg：GoldenGate 管理者アカウントの名前とパスワードがコマンドラインに表示されないように、curl によって読み取られる構成ファイルにユーザー名とパスワードを含めることをお奨めします。

例：

```
user = "oggadmin:password"
```

- GG Hub：GoldenGate Hub サーバーのホスト名または IP アドレス。たとえば、gghub-server などになります。
- デプロイメント名：これは、Oracle GoldenGate デプロイメントの名前です。たとえば、SOURCE または TARGET などになります。

例：

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/credentials/goldengate -XGET | python -m
json.tool
```

このホワイト・ペーパーで例示される Extract 名は EXT1 で、Replicat は REP1 といいます。

REST の呼び出しは、Oracle GoldenGate Hub、または HTTPS プロトコルを介して Oracle GoldenGate Hub にアクセスできる任意のマシンから行うことができます。

構成の前提条件

Oracle GoldenGateのソース・データベース・サポートを評価する

以下に詳述するように、Oracle GoldenGate には、ソース・データベースに対していつくかの要件があります。

- **データベース・パッチ要件**

最新のバンドル・パッチと PSU（Patch Set Update）をソース・データベースとターゲット・データベース双方に適用することがベスト・プラクティスです。完全な推奨パッチ・リストは、データベース・バージョン 11.2.0.4 以上を対象とする My Oracle Support Note 2193391.1 に掲載されています。

ソース・データベースと GoldenGate Hub 間のネットワーク・ラウンドトリップの待機時間が 8 ミリ秒を超過する場合、および最新のデータベース・バンドル・パッチまたは PS/CPU（Critical Patch Update）に含まれていない場合は、バグ 28849751 用のパッチもソース・データベースに適用することをお奨めします。

- **データタイプのサポート**

データタイプのレプリケーションの制約が原因で、Oracle GoldenGate による抽出に完全に対応しないオブジェクトがあるかどうかを判別するために、ソース・データベースでディクショナリ・ビュー DBA_GOLDENGATE_SUPPORT_MODE を使用します。

```
SQL> SELECT owner, object_name FROM DBA_GOLDENGATE_SUPPORT_MODE
WHERE support_mode NOT IN ('FULL','ID_KEY');
```

上記の問合せで挙げられている表はすべて、GoldenGate Extract のパラメータ TABLEEXCLUDE owner.object_name を使ってキャプチャから除外する必要があります。これらのオブジェクトは、データベース移行プロセスの最後にターゲット・データベースに手動でコピーする必要があります。

統合 Extract に対応していないデータタイプのリストについては、Oracle GoldenGate のドキュメントを参照してください。

https://docs.oracle.com/en/middleware/goldengate/core/19.1/oracle-db/1-understanding-whats-supported.html#GUID-110CD372-2F7E-4262-B8D2-DC0A80422806

GoldenGate の TABLEEXCLUDE パラメータの詳細については、次のサイトを参照してください。

https://docs.oracle.com/en/middleware/goldengate/core/19.1/reference/index.html

3. 行の一意性

Oracle GoldenGate が、レプリケートした更新と削除の正しいターゲット行を特定するには、ソース表とターゲット表に一意の行 ID が必要です。

これは通常、主キーまたは一意のキー索引で処理されます。そのようなキーを持たないことが判明した表がある場合、GoldenGate は、許容可能なすべての列を含んだ疑似キーを表内に作成する必要があります。バーチャル・カラム、UDT、関数ベースのカラム、拡張（32K）VARCHAR2/NVARCHAR2 カラム、Oracle GoldenGate ユーザーが Oracle GoldenGate 構成から明示的に除外するカラムは除外されます。

ソース・データベースがバージョン 12g Release 2 以上である場合は、データ・ディクショナリ・ビューの DBA_GOLDENGATE_NOT_UNIQUE を使って、主キーまたは非 NULL の一意の列を持たないすべての表を特定します。

常に一意の値を含んだ列が表にある場合は、代替キーを定義できます。この代替キーは、KEYCOLS 句を Extract の TABLE パラメータと Replicat の MAP パラメータに含めることで定義します。Oracle GoldenGate が見つけた既存の主キーまたは一意のキーは、指定したキーによってオーバーライドされます。

ソース表で行を確実に一意にするための詳細については、Oracle GoldenGate のドキュメントを参照してください。

https://docs.oracle.com/en/middleware/goldengate/core/19.1/oracle-db/additional-oracle-goldengate-configuration- considerations.html#GUID-644099C5-8950-496C-8592-446FB1566AFD

ソース・データベース構成

ソース・データベースの前提条件は、MAA のホワイト・ペーパー『Oracle GoldenGate Performance Best Practices』（https://www.oracle.com/technetwork/database/availability/maa-gg-performance-1969630.pdf）の「Configuring the Source Database」の項に記載されています。おもな前提条件は次のとおりです。

- データベースで ARCHIVELOG モードを有効にする。
- GoldenGate Extract プロセスによってすべての変更が REDO 内で見つかるように、データベース強制ロギングを有効にする。
- データベースの最小サプリメンタル・ロギングを有効にする。レプリケートされたオブジェクト用に追加のスキーマ・レベルのサプリメンタル・ロギングも必要です。
- 初期化パラメータの STREAMS_POOL_SIZE でストリート・プールを構成する。
- 初期化パラメータの ENABLE_GOLDENGATE_REPLICATION を有効にすることで、GoldenGate レプリケーションを有効にする。
- 統合 Extract のパフォーマンス分析用に UTL_SPADV パッケージをインストールする。

また、構成を必要とする項目がさらに 2 つあります。

Oracle Netの接続性

ハブで稼働するリモート GoldenGate Extract のパフォーマンスを最大限に引き出せるように、Oracle Net の接続性を最適化する必要があります。それには、別個の Oracle Net Listner をソース・データベース・ホスト上に作成します。この Oracle Net Listner は、現在の本番リスナー・サービスへの干渉をなくす役目も果たします。

リモート GoldenGate Extract のパフォーマンスの改善のため、より大きいサイズの SDU を使用するようソース・データベース・リスナーを設定することをお奨めします。リスナーが最大限の SDU サイズを設定していても、クライアントによってリクエストされた場合は、最小サイズが使用されます。たとえば、リスナーが SDU を 2 MB に設定したが、クライアントがデフォルトの 8 KB の SDU をリクエストした場合、データベースへの接続では、8 KB の SDU サイズが使用されます。Oracle Database 11g Release 2 の SDU サイズの最大値は 64 KB（65536 バイト）で、それより後のデータベース・リリースの最大値は 2 MB（2097152 バイト）です。

ソース・データベースへの接続には、Oracle Net または Secure Sockets Layer（SSL）の暗号化を使用することをお奨めします。SSL と Oracle Net について詳しくは、『Oracle Database Security Guide』を参照してください。

https://docs.oracle.com/en/database/oracle/oracle-database/19/dbseg/configuring-secure-sockets-layer-authentication.html#GUID-6AD89576-526F-4D6B-A539-ADF4B840819F

データベース移行のためだけに個別のリスナーを作成します。個別の TNS_ADMIN ディレクトリをソース・データベース・ホストに作成することで、現在の移行リスナーと新しく構成した移行リスナーが確実に分離されます。新しい TNS_ADMIN ディレクトリには、sqlnet.ora パラメータ・ファイルと listener.ora パラメータ・ファイルが格納されます。次の例の sqlnet.ora ファイルと listener.ora ファイルには SSL が構成され、Oracle Database 12g Release 1 以上の場合、SDU サイズが大きくなります。

sqlnet.ora

```
SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS = TRUE
SQLNET.WALLET_OVERRIDE = FALSE
SQLNET.EXPIRE_TIME = 10
WALLET_LOCATION =
(SOURCE=(METHOD=FILE)(METHOD_DATA=(DIRECTORY=/u01/oracle/tcps_wallets)))
SSL_VERSION = 1.2

# Parameters required for Net encryption if not using SSL Authentication, replacing
# the above parameters:
# SQLNET.ENCRYPTION_SERVER = accepted
# SQLNET.ENCRYPTION_TYPES_SERVER= (AES256)

DEFAULT_SDU_SIZE = 2097152
```

listener.ora

```
Migration =
(DESCRIPTION_LIST =
        (DESCRIPTION = (SDU =
            2097152)
            (ADDRESS = (PROTOCOL = TCPS)(HOST = <source database host>)(PORT = 2484))
        )
    )
```

```
SID_LIST_Migration = (SID_LIST = (SID_DESC = (SDU = 2097152) (SID_NAME =
<ORACLE_SID>) (ORACLE_HOME = <ORACLE_HOME>)))
```

ソース・データベースが Oracle RAC である場合、データベースのインスタンスを実行しているすべてのクラスタで移行リスナーを構成します。このようにすれば、移行中、1 つのインスタンスが停止しても、存続しているインスタンスにサービスを移行できます。すべての Oracle RAC ノード上で SSL ウォレットを構成します。

移行リスナーの起動/停止/ステータスの取得を行うには、次の環境変数を確実に設定してから、以下のコマンドを実行します。

```
export ORACLE_HOME=<oracle home directory> export
PATH=$PATH:$ORACLE_HOME/bin
export TNS_ADMIN=<TNS admin directory>
```

リスナーを起動するには、次のコマンドを実行します。

```
$ lsnrctl start Migration
```

リスナーを停止するには、次のコマンドを実行します。

```
$ lsnrctl stop Migration
```

リスナーの現在のステータスを取得するには、次のコマンドを実行します。

```
$ lsnrctl status Migration
```

Oracle GoldenGateデータベース管理者アカウント

ソース・データベースが現在、GoldenGate 構成の一部である場合、Oracle GoldenGate 管理者アカウントはすでに存在している可能性があります。ユーザーがすでに存在する場合、権限が正しく付与されていることを確認し、まだの場合は付与する必要があります。

GoldenGate 管理者ユーザーが存在しない場合は、ユーザーを作成する必要があります。推奨されるユーザー名は、単一テナントと PDB の場合は GGADMIN、CDB データベースの場合は C##GGADMIN です。

Oracle GoldenGate管理者アカウントが存在するかどうかを確認する

Oracle Multitenant を使用している場合は、以下の例のような文を使って、すべての PDB も確認します。

```
SQL> SELECT name, username FROM cdb_goldengate_privileges a, v$pdbs b
WHERE a.con_id = b.con_id UNION SELECT decode(a.con_id,1,'CDB ROOT'), username
FROM cdb_goldengate_privileges a, v$pdbs b WHERE a.con_id=1;
```

単一テナント・データベースの場合は、次の例を使用します。

```
SQL> SELECT username FROM dba_goldengate_privileges;
```

GoldenGate をまだデータベース上で構成していない場合、これらの問合せをしても行は返されないはずです。

GoldenGate 管理者ユーザーがすでに存在する場合は、データベース移行 GoldenGate 構成に使用します。以下に示す権限が現在の GoldenGate 管理者アカウントに付与されていることを確認します。

このホワイト・ペーパー全体を通じて、データベースの GoldenGate 管理者アカウントの名前は常に GGADMIN になります。

次の例を使って、新しい GoldenGate 管理者アカウントを作成します。

注：ソース・データベースが PDB である場合、レプリケート対象のすべての PDB 上でこのアカウントを作成する必要があります。

```
SQL> create user ggadmin identified by <password> default tablespace users temporary
tablespace temp;
SQL> grant connect, resource to ggadmin;
SQL> grant select any dictionary to ggadmin;
SQL> grant create view to ggadmin;
SQL> grant execute on dbms_lock to ggadmin;
SQL> exec dbms_goldengate_auth.GRANT_ADMIN_PRIVILEGE('ggadmin');
```

ソース・データベースが PDB である場合、個別のアカウントを CDB に作成する必要があります。

```
SQL> create user c##ggadmin identified by <password> default tablespace users
temporary tablespace temp;
SQL> grant connect, resource to c##ggadmin;
SQL> grant select any dictionary to ggadmin;
SQL> grant create view to c##ggadmin;
SQL> grant execute on dbms_lock to c##ggadmin;
SQL> exec dbms_goldengate_auth.GRANT_ADMIN_PRIVILEGE('c##ggadmin',container=>'all');
```

注：単一の PDB からデータをレプリケートすることだけに関心がある場合は、'all'を PDB 名に置き換えてください。

ターゲット・データベース構成

Oracle Data Pump エクスポート/インポートを使ってターゲット・データベースをインスタンス化する場合は、前提条件として次の手順を実行する必要があります。

ターゲット・データベースを作成する

スクリプトまたは Oracle Database Configuration Assistant（DBCA）を使って、空のターゲット・データベースを作成できます。

System 表領域、UNDO 表領域、一時表領域、オンライン REDO ログのサイズを、少なくともソース・データベースと同じにしておいてください。

ターゲット・データベースで暗号化を使用する場合は、暗号化された表領域をその時点で作成する必要があります。

データベース初期化パラメータを設定する

GoldenGate Replicat によるターゲット・データベースへの変更の適用を可能にするには、

以下に示すように、ENABLE_GOLDENGATE_REPLICATION パラメータを設定する必要があります。

```
SQL> ALTER SYSTEM SET enable_goldengate_replication=TRUE scope=both;
```

ターゲット・データベースが PDB である場合、このパラメータを CDB に設定する必要があります。

GoldenGate管理者データベース・アカウントを作成する

GoldenGate 管理者データベース・アカウントをターゲット・データベースに作成する必要があります。ターゲット・データベースが PDB である場合は、ソース・データベースの移行先 PDB に管理者アカウントを作成します。

GoldenGate 管理者アカウントを次のコマンドで作成します。

```
SQL> create user ggadmin identified by <password> default tablespace users temporary
tablespace temp;
SQL> grant connect, resource, dba to ggadmin;
SQL> grant select any dictionary to ggadmin; SQL> grant create view to ggadmin;
SQL> grant execute on dbms_lock to ggadmin;
SQL> exec dbms_goldengate_auth.GRANT_ADMIN_PRIVILEGE('ggadmin');
```

注：DDL とシーケンスのサポートには、DBA ロールが必要です。DDL またはシーケンスをレプリケートしない場合、DBA ロールは必要ありません。

データベース・ディレクトリ・オブジェクトを作成する

ターゲット・データベースへの Oracle Data Pump のインポートには、ディレクトリ・オブジェクトが必要です。また、インポートされる import logfile と export dumpfile の保存にも使用されます。Oracle Data Pump のネットワーク・インポートを使用している場合、export dumpfile は作成されません。

次のサンプル・コマンドは、ディレクトリ・オブジェクトを作成します（SYS または SYSTEM ユーザーとして）。

```
SQL> CREATE OR REPLACE DIRECTORY dpump AS '/u01/oracle/datapump';
```

ソース・データベースへのデータベース・リンクを作成する（任意）

ターゲット・データベースのインスタンス化に Oracle Data Pump のネットワーク・インポートを使用する場合は、以下の例に示すように、まず、ターゲット・データベースからソース・データベースへのデータベース・リンクを作成する必要があります。リンクはターゲット SYS または SYSTEM ユーザーが作成する必要があります。

```
SQL> create database link gg18.us.oracle.com connect to system identified by password using 'gg18';
```

注：TNS エントリがターゲット・データベース・サーバーのソース・データベース用の tnsnames.ora ファイルに追加されていることを確認します。

Oracle GoldenGate Hub構成

GoldenGate Hub には、3 種類のソフトウェアのインストールと構成が必要です。

- ソースの Oracle データベース・バージョンとターゲットの Oracle データベース・バージョンを一致させるための Oracle Client ソフトウェア
- ソースの Oracle データベース・バージョンとターゲットの Oracle データベース・バージョンを一致させるための Oracle GoldenGate ソフトウェア
- Oracle GoldenGate Microservices リバース・プロキシによって使用される NGINX リバース・プロキシ・サーバー

以下の項では、必要なソフトウェアのインストールについて詳しく説明します。

Oracle Clientソフトウェアをインストールする

Oracle GoldenGate には、Oracle Client Runtime インストールの一部としてインストールされるライブラリが必要です。インストールされるクライアント・ソフトウェア・バージョンは、GoldenGate の接続先データベースと同じリリース番号である必要があります。たとえば、ソースが Oracle Database 12c Release 2 で、ターゲットが Oracle Database 18c である場合、2 種類のクライアント・ソフトウェアのインストールが必要です。Oracle Database 12c 用と Oracle Database 18c 用それぞれのクライアント・ソフトウェアが必要です。ソース・データベースとターゲット・データベース双方のバージョンが同じ場合は、1 つの Oracle Client ソフトウェアのインストールが必要です。

注：Oracle Instant Client ソフトウェアは、必要なライブラリの一部が含まれていないため、インストールしないでください。

Oracle Client ソフトウェアは、https://edelivery.oracle.com からダウンロードできます。Oracle Software Delivery Cloud でリリース・カテゴリを選択し、"Oracle Database Client"を検索します。

以下に例示されているように、ソフトウェアのインストール時に Oracle Client Runtime ソフトウェアをインストールすることを選択してください。

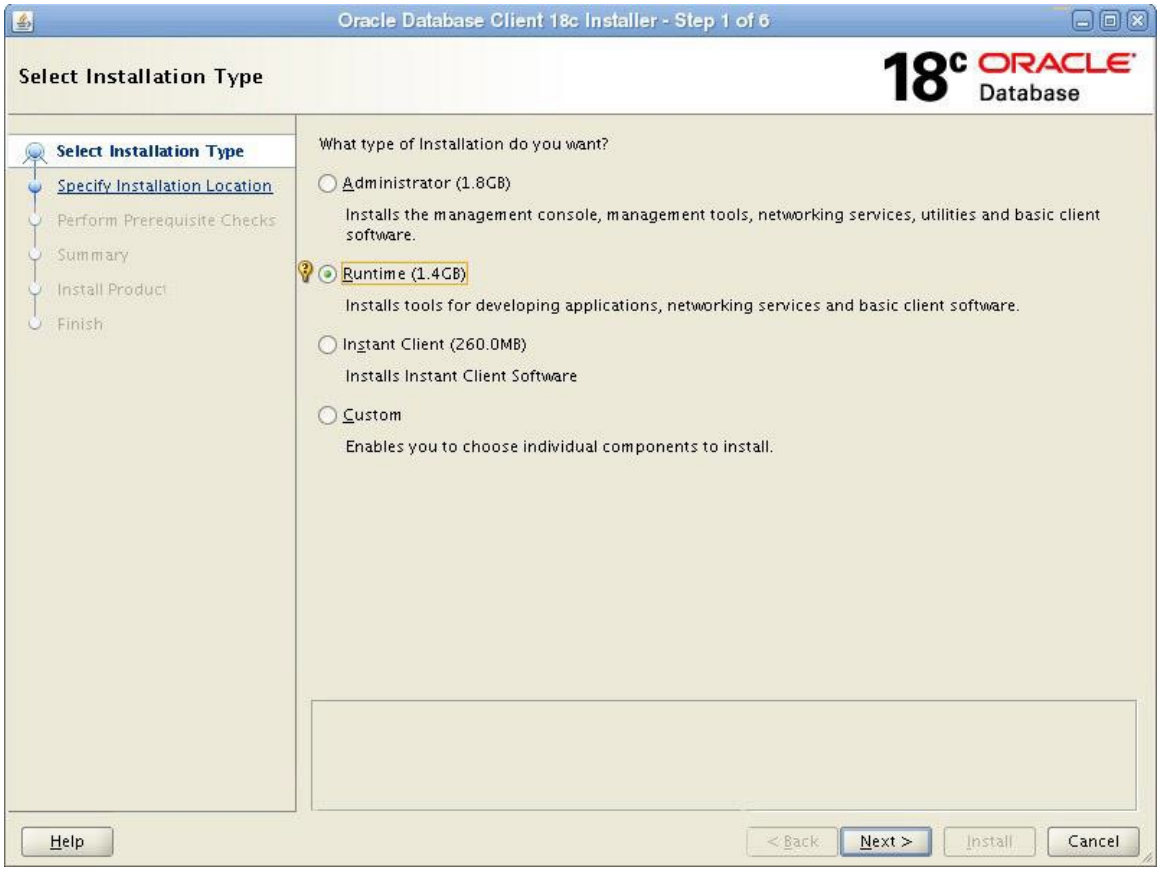


図 4：Oracle Client ソフトウェアのインストール

ソース・データベース・リリースとターゲット・データベース・リリースの両方に一致する Oracle Client ソフトウェアを別個の ORACLE_HOME ディレクトリにインストールする必要があることに留意してください。

Oracle GoldenGateソフトウェアをインストールする

ソースおよびターゲットの Oracle データベース・リリースをサポートする Oracle GoldenGate ソフトウェア・リリースをインストールする必要があります。Oracle GoldenGate ソフトウェアの互換性マトリックスは、

https://www.oracle.com/technetwork/middleware/ias/downloads/fusion-certification-100350.html に掲載されています。

最新の Oracle GoldenGate ソフトウェアを https://edelivery.oracle.com からダウンロードしてください。リリース・カテゴリを選択し、"Oracle GoldenGate"を検索します。入手可能な最新リリースの Oracle GoldenGate（現時点では 19c Release 1）をインストールすることをお奨めします。Oracle GoldenGate 19c Release 1 を使用すると、GoldenGate Extract を稼働するオペレーティング・システムのエンディアンネスがソース・データベース・プラットフォームのエンディアンネスと異なる場合に、エンディアン間抽出を行うことができます。

ソース・データベースとターゲット・データベースのリリースが異なる場合は、Oracle GoldenGate ソフトウェアを 2 回（データベース・リリースごとに 1 回ずつ）インストールしてください。図 5 に、データベース・リリースのソフトウェア・インストーラ・オプションを示します。

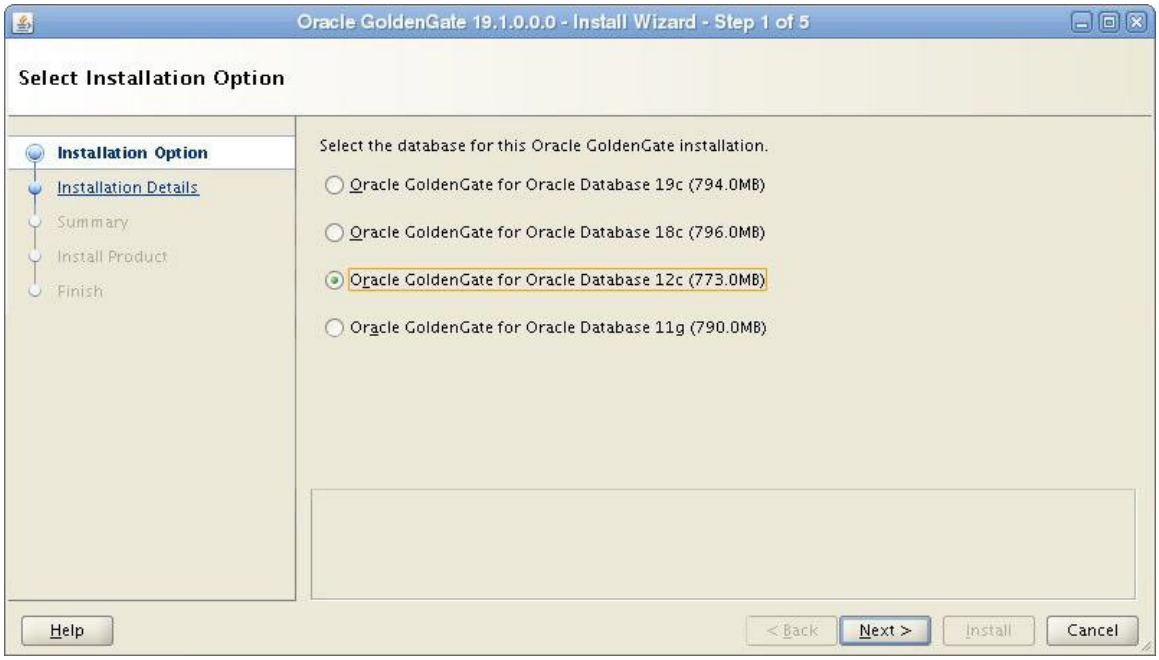


図 5：Oracle Database バージョンに合った Oracle GoldenGate リリースを選択する

Oracle GoldenGate ソフトウェアを個別のディレクトリにインストールする必要がある点に留意してください。

NGINXリバース・プロキシ・サーバーをインストールする

Oracle GoldenGate リバース・プロキシ機能を使用すると、GoldenGate デプロイメントに関連付けられたすべての GoldenGate Microservices に対して単一接点を設けることができます。リバース・プロキシがない場合、GoldenGate デプロイメントのマイクロサービスには、ホスト名または IP アドレス、および個別のポート番号（各サービスに 1 つずつ）からなる URL を使って接続します。たとえば、サービス・マネージャに接続するには、

http://gghub.example.com:9100 を使用できます。管理サーバーは http://gghub.example.com:9101、デプロイメント No. 2 の管理サーバーは https://gghub.example.com:9111 という具合になります。

リバース・プロキシがある場合、マイクロサービスはデプロイメント名に置き換えられているため、マイクロサービスへの接続にポート番号は不要です。前の例を使って、サービス・マネージャに接続するには https://gghub.example.com の URL を使用します。デプロイメント No. 1（名前：ソース）の管理サーバーの場合は https://gghub.example.com/Source/adminsrvr を使用し、デプロイメント No. 2（名前：ターゲット）の管理サーバーの場合は https://gghub.example.com/Target/adminsrvr となります。

マイクロサービスに容易にアクセスし、セキュリティと管理性を強化するには、Oracle GoldenGate リバース・プロキシをお奨めします。

Oracle GoldenGate リバース・プロキシ機能は NGINX ベースのリバース・プロキシを使用します。以下の手順は、リバース・プロキシの構成方法を示します。

1. NGINX がまだインストールされていないことを確認します（root として）。

```
$ sudo rpm -qa |grep nginx
```

インストールされていない場合、何も返されません。

NGINX がインストールされている場合、出力は次のようになります。

```
nginx-mod-http-xslt-filter-1.12.2-2.el7.x86_64
nginx-mod-http-image-filter-1.12.2-2.el7.x86_64
nginx-filesystem-1.12.2-2.el7.noarch
nginx-mod-mail-1.12.2-2.el7.x86_64
nginx-mod-http-perl-1.12.2-2.el7.x86_64
nginx-1.12.2-2.el7.x86_64
nginx-all-modules-1.12.2-2.el7.noarch
nginx-mod-http-geoip-1.12.2-2.el7.x86_64
nginx-mod-stream-1.12.2-2.el7.x86_64
```

2. NGINX がまだインストールされていない場合はインストールしてください。次の手順は、YUM（Yellowdog Updated Modified）がインストールされ、パッケージのインストールが容易になるように構成されていることを前提としています。

```
$ sudo yum install epel-release
$ sudo yum update
$ sudo yum install nginx
```

3. NGINX を起動します。

```
$ sudo nginx
```

4. NGINX が実行されていることを確認します。

```
$ curl -I 127.0.0.1

HTTP/1.1 200 OK
Server: nginx/1.12.2
```

```
Date:Wed, 4 Apr 2019 21:48:43 GMT
Content-Type: text/html
Content-Length:3700
Last-Modified:Wed, 4 Apr 2019 05:06:50 GMT Connection:
keep-alive
ETag:"5abdc5ea-e74"
Accept-Ranges: bytes
```

NGINX のインストールに関する完全なドキュメントについては、https://docs.nginx.com/nginx/admin-guide/installing- nginx/installing-nginx-open-source/にアクセスしてください。

Oracle GoldenGate Microservicesデプロイメントを構成する

Oracle Client ソフトウェアと GoldenGate ソフトウェアのインストールと同様に、運用対象のデータベースごとに個別の GoldenGate デプロイメントが必要です。ソース・データベースとターゲット・データベース双方のリリースが同じ場合は、1 つのデプロイメントのみ必要です。

各デプロイメントは、管理サーバーと（オプションの）パフォーマンス・メトリック・サーバーで作成されます。証跡ファイルは Replicat プロセスと同じサーバーに保存されるため、ディストリビューション・サーバーまたはレシーバ・サーバーを作成する必要はありません。

最初のデプロイメントの作成によって 1 つの Oracle GoldenGate サービス・マネージャが作成されます。その後のデプロイメントは、既存のサービス・マネージャを使って作成されます。

Oracle GoldenGate デプロイメントを作成する方法は 2 つあります。

1. デプロイメントを作成するためのグラフィカル・ツールである Oracle GoldenGate Configuration Assistant（oggca.sh）を使用
2. すべてのデプロイメント・パラメータ値を含むレスポンス・ファイルをサイレント・モードで使用

デプロイメントの方法は両方とも、同じ結果になりますが、複数のデプロイメントを作成する場合は、レスポンス・ファイルを使用するオプション 2 を使用した方が簡単です。付録 A には、2 つのデプロイメントの作成に必要な変更をパラメータに加えたレスポンス・ファイルのサンプルを掲載しています。デプロイメントのうちの 1 つは、ソース・データベースとターゲット・データベースのリリースが異なり、Oracle GoldenGate 19c を使用する場合です。Oracle GoldenGate ソフトウェアのインストールには、デフォルトのレスポンス・ファイルは含まれていません。初期デプロイメントは、oggca.sh を使って作成する必要があるためです。図 6 に示す以下のスクリーンショットでは、レスポンス・ファイルは oggca.sh で作成されます。

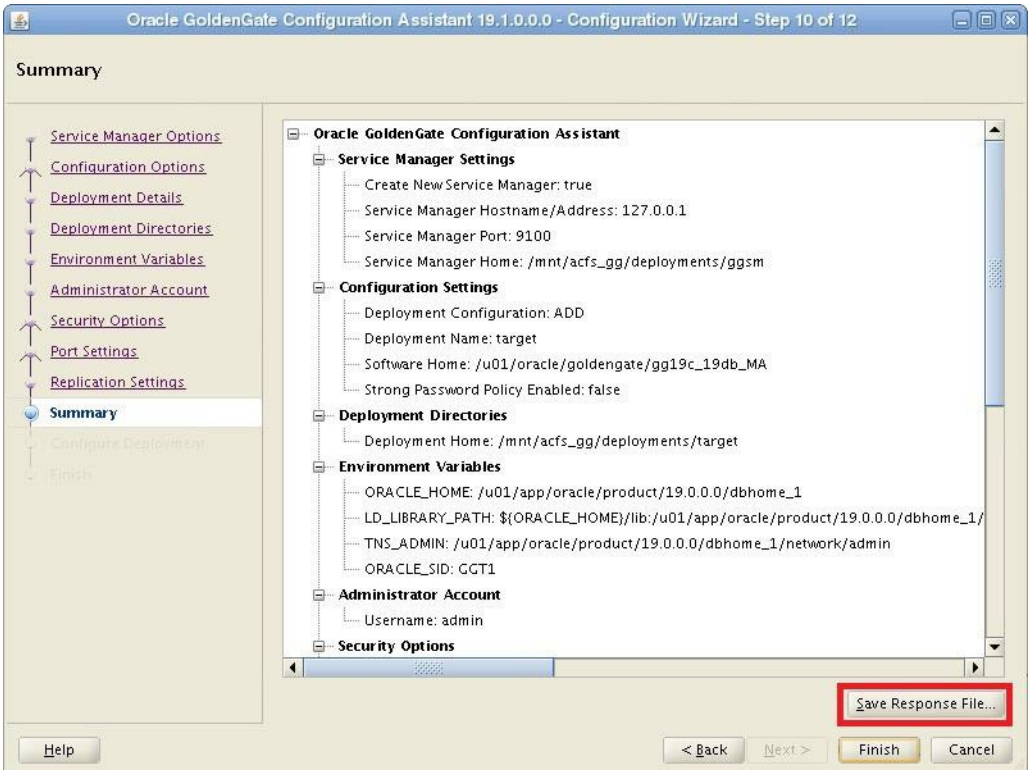


図 6：デプロイメント・レスポンス・ファイルの作成

付録 A のサンプルのレスポンス・ファイルを使用して正しいパラメータを設定した後、次のコマンドを使って初期デプロイメントを作成します。

```
$ cd $GG_HOME/bin
$ ./oggca.sh -silent -responseFile /u01/oracle/target_deployment.rsp
```

完了すると、次のメッセージが表示されます。

```
Linux, x64, 64bit (optimized) on Mar 28 2019 14:38:11
Operating system character set identified as UTF-8.
Stopping...STOPPED.
Successfully Setup Software.
```

ソース・データベースとターゲット・データベースの Oracle リリースが、12g Release 2 と 19c などのように異なる場合は、デプロイメントをさらに作成する必要があります。

2 番目の Oracle GoldenGate インストールのデプロイメントを繰り返します。最初のデプロイメントとは異なるようにパラメータを設定します。特にポート番号とデプロイメント名を変えるようにしてください。2 番目のデプロイメントでは、新しいサービス・マネージャを作成せずに、最初のデプロイメントですでに作成したサービス・マネージャを使用します。2 番目のデプロイメントを作成する場合のサンプルのレスポンス・ファイルについては、付録 A を参照してください。

注：パラメータ名とパラメータの数がリリースごとに異なる可能性があるため、インストールした Oracle GoldenGate ソフトウェア・バージョンのレスポンス・ファイル・テンプレートを必ず使用してください。

各デプロイメントの作成後、デプロイメントは自動的に起動します。デプロイメントの起動と停止を行うためのサンプル・スクリプトについては、付録 B を参照してください。

このホワイト・ペーパー全体を通じて、2 つの GoldenGate デプロイメントはソースおよびターゲットと呼びます。

NGINXリバース・プロキシを構成する

GoldenGate Hub では GoldenGate サービス・マネージャが 1 つだけ使用されるので、リバース・プロキシは 1 つのリバース・プロキシ・サーバーを使って簡単に構成できます。

1. NGINX 用のサーバー証明書を作成する。

NGINX が HTTPS を使ってクライアント・リクエストを認証するには、証明書が必要です。証明書を作成または取得する際には、会社の正しい標準に従う必要があります。

証明書は/etc/nginx/ogg.pem の場所にコピーしてください。

新しい証明書は、構成のリロード時に NGINX によって後で読み取られます。

2. GoldenGate 環境変数を設定し、NGINX 構成ファイルを作成する。

OGG_HOME 環境変数が、サービス・マネージャを実行する GoldenGate ソフトウェア（上記で作成した最初のデプロイメント）のホーム・ディレクトリに設定されていることを確認します。

```
export OGG_HOME=/app/oracle/goldengate/target
export PATH=$PATH:$OGG_HOME/bin

$ $OGG_HOME/lib/utl/reverseproxy/ReverseProxySettings –u <username> -P <password> -o ogg.conf
http://localhost:9100
```

–u パラメータと–P パラメータで指定したユーザー名とパスワードは、初期デプロイメントの作成時に（ADMINISTRATOR_USER および ADMINISTRATOR_PASSWORD レスポンス・ファイル・パラメータに）指定した GoldenGate 管理者（非データベース）アカウントを反映している必要があります。ポート番号（9100）も、初期デプロイメントの作成時に（PORT_SERVICEMANAGER レスポンス・ファイル・パラメータに）指定したサービス・マネージャのポート番号と一致している必要があります。

デプロイメントを追加または削除するなど、GoldenGate デプロイメントの構成に変更が加えられるたびに、このコマンドを再実行してください。

3. 既存の NGINX 構成を新しい構成に置き換える。

```
$ sudo mv ogg.conf /etc/nginx/conf.d/
```

4. 新しい NGINX 構成をテストする。

```
$ sudo nginx –t

nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

5. NGINX と新しい構成をリロードする。

```
$ sudo nginx -s reload
```

6. Oracle GoldenGate Microservices の接続をテストする。

簡単なテスト方法は、次のコマンドを使ってデプロイメントの状態を問い合わせる方法です。

```
$ curl -s –K access.cfg https://localhost/services/v2/config/health -XGET | python -m json.tool
```

サンプル出力：

```
{
        "$schema": "api:standardResponse", "links": [
                {
                        "href": "https://localhost/services/v2/config/health",
                        "mediaType": "application/json",
                        "rel": "canonical"
                },
                {
...
        "response": {
                "$schema": "ogg:health",
                "criticalResources": [
                        {
                                "deploymentName": "source",
                                "healthy": true,
                                "name": "adminsrvr",
                                "status": "running",
                                "type": "service"
                        },
                        {
                                "deploymentName": "target”,
                                "healthy": true,
                                "name": "adminsrvr",
                                "status": "running",
                                "type": "service"
                        },
                ],
                "deploymentName":"ServiceManager",
                "healthy": true,
                "serviceName":"ServiceManager",
                "started":"2019-03-28T21:52:42.835Z"
        }
}
```

Oracle GoldenGateの構成

この項では、ターゲット・データベースをインスタンス化する前に GoldenGate Hub で実行する必要がある、Oracle GoldenGate の構成手順について説明します。

ソース側のGoldenGate資格証明を作成する

GoldenGate プロセスがソース・データベースとターゲット・データベースに接続するには、GoldenGate データベース資格証明が必要です。GoldenGate 資格証明ストアは、クリア ・テキスト・パスワードが Extract または Replicat パラメータ・ファイルのいずれにも保存されないようにします。ターゲット・データベースはまだ作成されていないため、この手順ではソース・データベースの資格証明のみを作成します。ターゲット・データベースの資格証明は、後の手順で作成します。

Oracle Net構成ファイルの準備

データベース資格証明は Oracle Easy Connect ネーミング・メソッドを使用するので、ローカルの tnsnames.ora ファイルを使用する必要がありません。ただし、データベース・バージョンが Release 19c より前で、SSL 認証を使用している場合は除きます。そのような場合、Easy Connect のネーミングは使用できず、tnsnames.ora が必要です。

以下は、19c より前の Oracle RAC データベースで SSL 認証を使用する場合の、tnsnames.ora エントリの例です。

```
GGSOURCE =
    (DESCRIPTION=
        (SDU=2097152)           # Or set to 65536 for 11.2.0.4 database
        (CONNECT_TIMEOUT=10)(RETRY_COUNT=3)
        (ADDRESS=(PROTOCOL=TCPS)(HOST=<primary db scan address>)(PORT=2484))
        (CONNECT_DATA=(SERVICE_NAME=<db service name>))
    )
```

ソース・データベースが SSL 認証を使用していて、データベースのバージョンが 19c より前の場合、次のパラメータを指定した sqlnet.ora ファイルが必要です。

```
SSL_CLIENT_AUTHENTICATION=TRUE
SSL_SERVER_DN_MATCH=OFF
SQLNET.EXPIRE_TIME = 10
SQLNET.WALLET_OVERRIDE = FALSE
WALLET_LOCATION =
(SOURCE=(METHOD=FILE)(METHOD_DATA=(DIRECTORY="/u01/oracle/network")))
SSL_VERSION = 1.2
DEFAULT_SDU_SIZE=2097152              # Change to 65536 for 11.2.0.4 database
```

注：ソース・データベース環境のSSLウォレットを必ず、GoldenGate HubのDIRECTORYディレクトリにコピーしてください。

最後に、ソース・データベースが SSL 認証の代わりに Oracle Net 暗号化を使用している場合（データベースのバージョンは問わない）、次のパラメータを指定した sqlnet.ora ファイルが必要です。

```
SQLNET.ENCRYPTION_CLIENT = required
SQLNET.ENCRYPTION_TYPES_CLIENT= (AES256)
```

```
# The SDU size is only required for databases earlier than 19c:
DEFAULT_SDU_SIZE=2097152              # Change to 65536 for 11.2.0.4 database
```

Oracle Database 19c は Oracle Easy Connect ネーミング・メソッドを拡張して、Oracle Easy Connect Plus という名称にしました。Oracle Easy Connect Plus ネーミング・メソッドについて詳しくは、『Oracle Net Administrators Guide』を参照してください。

https://docs.oracle.com/en/database/oracle/oracle-database/19/netag/configuring-naming-methods.html#GUID-8C85D289-6AF3-41BC-848B-BF39D32648BA

GoldenGateデータベースの資格証明を作成する

tnsnames.ora ファイルを使用している場合は、TNS エイリアスを使ってソース・データベースへの資格証明を作成します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/credentials/goldengate/source -X POST --data
'{"userid":"ggadmin@GGSOURCE", "password":"<password>"}' | python -m json.tool
```

tnsnames.ora ファイルを使用していないが、データベース・バージョンが 19c より前のリリースである場合は、次の Easy Connect のネーミング例（および SCAN リスナー）を使ってソース・データベースへの資格証明を作成します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/credentials/goldengate/source -X POST --data
'{"userid":"ggadmin@database-host-scan:1521/database_service_name",
"password":"<password>"}' | python -m json.tool
```

次の例は、Oracle RAC と SSL 認証が構成された 19c ソース・データベース用の資格証明を作成する方法を示します。

```
curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/credentials/goldengate/source -X POST --data
'{"userid":"ggadmin@tcps://database-host-scan:2484/db_service_name?sdu=2097152&
SSL_SERVER_DN_MATCH=NO&ssl_server_cert_dn='cn=common_name&wallet_location='/u01/orac le/network'",
"password":"<password>"}' | python -m json.tool
```

以下は、sqlnet.ora ファイルで有効にした Oracle Net 暗号化を使用する資格証明を Oracle RAC データベースで作成する例です。

```
$ curl -s –K access.cfg https://gghub-
 server/SOURCE/adminsrvr/services/v2/credentials/goldengate/source -X POST --data
'{"userid":"ggadmin@tcp://database-host-scan:1521/db_service_name?sdu=2097152",
"password":"<password>"}' | python -m json.tool
```

マルチテナント・データベースの場合は、上記のガイドラインに従って、移行対象の CDB と PDB 用の資格証明を個別に作成する必要があります。

作成された GoldenGate 資格証明を表示するには、次のコマンドを使用します。

1. デプロイメントの GoldenGate 資格証明の名前を表示します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/credentials/goldengate -XGET | python -m json.tool | grep name

"name": "source_cdb"
"name": "source_pdb"
```

2. 資格証明の接続の詳細を表示します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/credentials/goldengate/source_cdb -XGET | python
-m json.tool

        "response": {
                "$schema": "ogg:credentials",
                "userid": "ggadmin@tcps://database-host-
scan:2484/db_service_name?sdu=2097152&SSL_SERVER_DN_MATCH=NO&ssl_server_cert_dn='cn=c
ommon_name&wallet_location='/u01/oracle/network'"
        }
```

3. 資格証明を削除する場合は、次のコマンドを使用します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/credentials/goldengate/source_cdb
-X DELETE | python -m json.tool
```

ソース・データベース上でハートビート表を作成する

ソース・データベースとターゲット・データベース間のレプリケーションの待機時間を監視するには、GoldenGate ハートビート表が必要です。ソース・データベースにすでに GoldenGate ハートビート・オブジェクトが含まれている場合は、それらを使用してください。

1. ハートビート表がすでに存在するかを確認し、以下に示すように、前の手順で作成したソース・データベースの資格証明を使用します。

注：マルチテナント・データベースの場合、ハートビート表は、レプリケートされる PDB に配置する必要があります。必ずソース PDB 資格証明を使用して、ハートビート表を確認してください。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/connections/goldengate.source_pdb>/tables/heartbeat -XGET | python -m json.tool
```

次の出力例は、ハートビート表が現在構成されていないことを示します。

```
        "messages": [
                {
                        "$schema": "ogg:message",
                        "code":"OGG-08100",
                        "issued":"2019-01-08T23:06:38Z",
```

```
                "severity":"INFO",
                "title":"Heartbeat table ggadmin.gg_heartbeat does not exist.",
                "type": http://docs.oracle.com/goldengate/c1910/gg-
winux/GMESG/oggus.htm#OGG-08100
            },
            {
                "$schema": "ogg:message",
                "code":"OGG-08100",
                "issued":"2019-01-08T23:06:38Z",

                "severity":"INFO",
                "title":"Heartbeat table ggadmin.gg_heartbeat_seed does not exist.",
                 "type": http://docs.oracle.com/goldengate/c1910/gg-
winux/GMESG/oggus.htm#OGG-08100
            },
            {
                "$schema": "ogg:message",
                "code":"OGG-08100",
                "issued":"2019-01-08T23:06:38Z",
                "severity":"INFO",
                "title":"Heartbeat table ggadmin.gg_heartbeat_history does not exist.",
                "type": http://docs.oracle.com/goldengate/c1910/gg-
winux/GMESG/oggus.htm#OGG-08100
            }
        ]
```


次の出力例は、GoldenGate ハートビート表がすでに存在することを示します。


```
        "messages": [
            {
                "$schema": "ogg:message",
                "code":"OGG-08100",
                "issued":"2019-01-08T23:31:08Z",
                "severity":"INFO",
                "title":"HEARTBEAT table ggadmin.gg_heartbeat exists.",
                "type": http://docs.oracle.com/goldengate/c1910/gg-
winux/GMESG/oggus.htm#OGG-08100
            },
...
            {
                "$schema": "ogg:message",
                "code":"OGG-08100",
                "issued":"2019-01-08T23:31:08Z",
                "severity":"INFO",
                "title":"HEARTBEAT table ggadmin.gg_heartbeat_seed supplemental logging
ENABLED.",
```

```
                    "type": http://docs.oracle.com/goldengate/c1910/gg-
winux/GMESG/oggus.htm#OGG-08100
                },
...
        "response": {
                "$schema": "ogg:tablesHeartbeat",
                "addTrandata": true,
                "frequency": 300,
                "partitioned": false,
                "purgeFrequency":1,
                "retentionTime": 30, "targetOnly": false
        }
```

ハートビート表がすでに使用されている場合は、同じ表が新しく移行した GoldenGate Extract によって使用されます。

2. 既存のハートビート表の頻度がデフォルトの 60 秒ではない場合、現在の値をメモし、移行の間 60 秒に変更します。

注：移行完了後、元の値に戻せるように、変更する前に頻度の値を覚えておいてください。

ハートビート更新頻度を変更します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/connections/goldengate.source_pdb/tables/heartbea t -X PATCH --data
'{"frequency":60}'| python -m json.tool
```

3. ハートビート表が存在しない場合は、ソース・データベース内にハートビート表を作成します。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/connections/goldengate.source_pdb/tables/heartbeat -X POST
--data '{}'| python -m json.tool
```

ターゲット・データベースのハートビート表は、データベースのインスタンス化後に作成します。

インスタンス化のためにソース・データベースのスキーマを準備する

データベース移行の一部であり、レプリケート対象となっているすべてのデータベース・スキーマのうちのソース・スキーマをレプリケーションできるように準備する必要があります。

注：インスタンス化のためにデータベース・スキーマを準備する作業は、GoldenGate Extract プロセスの作成前に完了する必要があります。

1. 次のコマンドを使って、各データベース・スキーマをインスタンス化します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/connections/goldengate.source_pdb/trandata/schema
-X POST --data '{"operation":"add","schemaName":"soesmall",
"prepareCsnMode":"nowait"}' | python -m json.tool
```

2. 次のコマンドを使って、ソース・データベースの準備した各スキーマを確認します。

次のコマンドで 1 つのスキーマを確認します。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/connections/goldengate.source_pdb/trandata/schema -X POST -- data
'{"operation":"info","schemaName":"soesmall"}'| python -m json.tool
```

次のコマンドで、すべてのデータベース・スキーマ（レプリケートするスキーマはこれらのスキーマから取り出すことができます）を確認します。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/connections/goldengate.source_pdb/trandata/schema -X POST -- data
'{"operation":"info","schemaName":"*"}'| python -m json.tool
```

GoldenGate Extractプロセスを作成および起動する

次の項では、Extract の存在確認、Extract の追加、プロセスの起動と停止、および Extract の削除方法について説明します。

1. Extract が存在することを確認します（すでに作成していて、削除されていない場合）。

```
$ curl -s –K access.cfg https://gghub-
 server/SOURCE/adminsrvr/services/v2/extracts/EXT1 -X GET | python -m json.tool
```

次の例は、Extract が存在しない場合のレスポンスを示します。

```
      "messages": [
            {
                  "$schema": "ogg:message",
                  "code":"OGG-12029",
                  "issued":"2019-01-31T04:52:41Z",
                  "severity":"INFO",
                  "title":"The extract with name 'EXT1' does not exist.",
                   "type": http://docs.oracle.com/goldengate/c1910/gg-
winux/GMESG/oggus.htm#OGG-12029
            }
```

次の例は、Extract が存在する場合のレスポンスを示します。

```
    "messages": [],
      "response": {
```

```
        "$schema": "ogg:extract", "begin":
        "now", "config": [
                "Extract EXT1",
                "ExtTrail aa",
                "UseridAlias source_cdb DOMAIN goldengate",
                 "TRANLOGOPTIONS PERFORMANCEPROFILE HIGH",
                "TRANLOGOPTIONS _readaheadcount 64"
                "REPORTCOUNT EVERY 15 MINUTES, RATE",
                 "STATOPTIONS REPORTFETCH",
                "DDL EXCLUDE ALL",
                "Table pdb1.soesmall.*;"
        ],
        "credentials": {
                "alias": "source_cdb",
                "domain": "goldengate"
        },
        "registration": {
                "containers": [
                        "PDB1"
                ],
                "csn":51162830
        },
        "source": {
                "tranlogs": "integrated"
        },
        "status": "stopped",
        "targets": [],
        "type":"Integrated"
}
```

2. Extract プロセスを作成します。

以下は、推奨される Extract 最小パラメータ値です。

```
-- Replace with trail file naming standard

EXTTRAIL aa
TRANLOGOPTIONS PERFORMANCEPROFILE HIGH
TRANLOGOPTIONS _readaheadcount 64                -- Required for streaming protocol (bug fix 28849751)
DISCARDFILE APPEND
REPORTCOUNT EVERY 15 MINUTES, RATE STATOPTIONS
REPORTFETCH
```

```
-- It is recommended to prevent DDL on the source database during the database
-- migration:
DDL EXCLUDE ALL

-- Repeat TABLE command for each schema being replicated:
TABLE [ container. ]<schema_name>.*;

-- Repeat TABLEEXCLUDE command for each table that was highlighted as not supported
-- for GoldenGate replication:
TABLEEXCLUDE [ container. ]<schema_name>.<tablename>;
```

ソース・データベースが PDB の場合、データベースの REDO ストリーム全体を読み取る必要があるため、Extract を CDB に接続する必要があります。PDB からレプリケートされるオブジェクトは、TABLE パラメータによって特定されます。

次のコマンドを使って、Extract を作成します。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/extracts/EXT1 -X POST --data '{"config":["Extract EXT1",
"ExtTrail aa","UseridAlias source_db DOMAIN goldengate","TRANLOGOPTIONS PERFORMANCEPROFILE HIGH",
"TRANLOGOPTIONS _readaheadcount 64", "REPORTCOUNT EVERY 15 MINUTES, RATE","STATOPTIONS
REPORTFETCH","DDL EXCLUDE ALL","Table
pdb1.soesmall.*;"],"source":{"tranlogs":"integrated"},"credentials":{"alias":"source_
db","domain":"goldengate"},"begin":"now","targets":[{"name":"aa","sizeMB":500}],"regi stration":{"containers":["pdb1"]}}'|
python -m json.tool
```

注：バグ 28849751 のパッチをソースのオンプレミス・データベースに適用した後、_readaheadcount パラメータが必要です。

注：複数のスキーマを抽出する場合は、複数の Table <Source PDB>.<Schema Name>.;パラメータを使用します。*

次のコマンドを使って、非 CDB データベースの Extract を作成します。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/extracts/EXT1 -X POST --data '{"config":["Extract EXT1",
"ExtTrail aa","UseridAlias source_db DOMAIN goldengate","TRANLOGOPTIONS PERFORMANCEPROFILE HIGH",
"TRANLOGOPTIONS _readaheadcount 64", "REPORTCOUNT EVERY 15 MINUTES, RATE","STATOPTIONS
REPORTFETCH","DDL EXCLUDE ALL","Table
pdb1.soesmall.*;"],"source":{"tranlogs":"integrated"},"credentials":{"alias":"source_
db","domain":"goldengate"},"begin":"now","targets":[{"name":"aa","sizeMB":500}],"registration":"default"}'| python -m
json.tool
```

3. Extract 構成を確認します。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/extracts/EXT1 -X GET | python -m json.tool
```

4. Extract のステータスを確認します。

Extract は自動的に起動しないので、ステータスは停止になっているはずです。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/extracts/EXT1/info/status -X GET | python -m json.tool
```

次の例は、Extract が起動しない場合に予想される結果を示します。

```
    "messages": [],
    "response": {
        "$schema": "ogg:extractStatus",
        "lag":0,
        "lastStarted": null,
        "position":"0.0",
        "sinceLagReported": 208,
        "status": "stopped"
    }
```

5. Extract を起動します。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/commands/execute -X POST --data
'{"name":"start","processName":"EXT1"}' | python -m json.tool
```

2 分待ってから、Extract のステータスを確認します。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/extracts/EXT1/info/status -X GET | python -m json.tool
```

適切に起動した後の出力は次のようになります。

```
    "response": {
        "$schema": "ogg:extractStatus",
         "lag":1,
        "lastStarted":"2019-01-31T19:09:45.524Z",
        "position":"0.51251093",
        "processId":4702,
        "sinceLagReported": 2,
         "status": "running"
    }
```

AUTOSTARTタスクを作成する

デプロイメントが起動したときに、GoldenGate プロセスを自動的に起動させるには、AUTOSTART プロファイルと AUTORESTART プロファイルが必要です。GoldenGate プロセスは、これらのプロセスを自動的に起動させるアクティブ・プロファイルの一部としてデフォルト設定されていません。

1. AUTOSTART プロファイルが存在するか確認してください。存在する場合は、削除または更新することができます。

AUTOSTART プロファイルが存在するかどうかを確認します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/config/types/ogg:managedProcessSettings/values/og
g:managedProcessSettings:MIGRATE01 -XGET | python -m json.tool
```

次の出力例は、プロファイルが存在しない場合を示します。

```
        "messages": [
                {
                        "$schema": "ogg:message",
                         "code":"OGG-12029",
                        "issued":"2019-02-06T17:22:43Z",
                        "severity":"INFO",
                        "title":"The value with name 'ogg:managedProcessSettings:MIGRATE01' does not exist.",
                        "type": http://docs.oracle.com/goldengate/c1910/gg-
winux/GMESG/oggus.htm#OGG-12029
                }
        ]
```

プロファイルが存在する場合は、次の出力が表示されます。

```
        "response": {
                "autoRestart": {
                 "delay":60,
                        "disableOnFailure": true,
                        "enabled": true,
                        "onSuccess": false,
                        "retries":5,
                        "window":1200
                },
                "autoStart": {
                        "delay": 60,
                         "enabled": true
                }
        }
```

プロファイルがすでに存在する場合は、削除するか、次の推奨設定値で更新できます。プロファイルを削除する場合：

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/config/types/ogg:managedProcessSettings/values/og
g:managedProcessSettings:MIGRATE01 -XDELETE | python -m json.tool
```

推奨設定値で現在のプロファイルを更新する場合：

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/config/types/ogg:managedProcessSettings/values/og
g:managedProcessSettings:MIGRATE01 -XPUT --data '{"autoStart": {"enabled": true,"delay": 60},"autoRestart": {"delay":
60,"disableOnFailure": true,"enabled":
true,"onSuccess": false,"retries": 5,"window": 1200}}' | python -m json.tool
```

2. ソース・デプロイメントおよびターゲット・デプロイメント双方の次の AUTOSTART プロファイルと AUTORESTART プロファイルを作成します。

プロファイルが現在存在しない場合、または前の手順で削除した場合は、プロファイルを作成します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/config/types/ogg:managedProcessSettings/values/og
g:managedProcessSettings:MIGRATE01 -XPOST --data '{"autoStart": {"enabled":
true,"delay": 60},"autoRestart": {"delay": 60,"disableOnFailure": true,"enabled":
true,"onSuccess": false,"retries": 5,"window": 1200}}' | python -m json.tool
```

3. AUTOSTART プロファイルを Extract に割り当てます。

注：GoldenGate Replicat はまだ作成していないので、この時点でプロファイルを割り当てることはできません。

```
$ curl -s –K access.cfg https://gghub-server/SOURCE
/adminsrvr/services/v2/extracts/EXT1 -X PATCH --data '{"managedProcessSettings":
"MIGRATE01"}' | python -m json.tool
```

Extract に新しいプロファイルが構成されていることを確認します（部分的な出力を示します）。

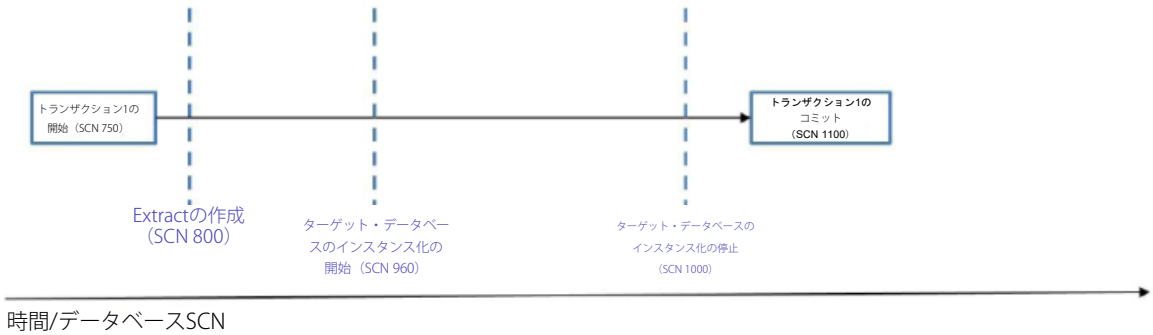
```
$ curl -s –K access.cfg https://gghub- server/SOURCE/adminsrvr/services/v2/extracts/EXT1 -X GET | python -m
json.tool

    "managedProcessSettings":"MIGRATE01",
```

ソース・データベースの長時間実行トランザクションを監視する

データベースを使ってターゲット・データベースをインスタンス化するには、GoldenGate ですべてのトランザクション・データが確実にレプリケートされるよう、GoldenGate Extract の作成時点でアクティブだったトランザクションを確実にコミットさせることが重要です。

たとえば、次のタイムラインについて考えてみます。



トランザクション 1 はソース・データベースで SCN 1100 の時点にコミットされますが、Extract によって取得されません。それは、Extract は完全なトランザクションのみを処理し、プロセスはトランザクション開始後に作成されたからです。SCN 750 は 800 より前なので、このトランザクションは無視されます。

次の問合せをソース・データベースに対して実行して、Extract の作成前に開始されたトランザクションすべてがコミットまたはロールバックされた時点を特定し、ターゲット・データベースのインスタンス化を安全に開始できるようにします。

```
SQL> SELECT capture_name, c.start_scn, t.start_scn
FROM dba_capture c, v$transaction t WHERE t.start_scn < c.start_scn;
```

問合せをしても行が返されない場合は、ターゲット・データベースのインスタンス化を開始しても安全です。

ターゲット・データベースのインスタンス化

このホワイト・ペーパーでは、ターゲット・データベースをインスタンス化する次の 2 つの方法を紹介します。

1. Oracle RMAN 複製データベース – Oracle のソースおよび移行先データベースのバージョンが同じで、プラットフォーム・エンディアン形式が同じである場合に、ソース・データベースを Recovery Manager によって複製できます。このインスタンス化方法は、ターゲット・データベースをインスタンス化する最速の方法です。
2. Data Pump によるエクスポート/インポート – ソース・データベースは、Oracle Data Pump を使ってエクスポートされ、空のターゲット・データベースにインポートされます。ソースおよび移行先 Oracle データベースのバージョンが異なる場合、プラットフォーム・エンディアン形式が異なる場合、またはデータベース構造の変更がある場合（単一テナントからマルチテナントまたは暗号化されたデータベースへの移行など）に、このインスタンス化方法が必要です。

Oracle RMAN複製データベース

ソースと移行先のプラットフォームのエンディアンネスが一致し、Oracle Database バージョンが同じである場合は、Oracle RMAN を使ってソース・データベースを複製する方法によって、データベース全体をはるかに簡単にコピーできます。Oracle RMAN を使ったソース・データベース複製方法の完全な詳細については、次のサイトの『Oracle Backup and Recovery User's Guide』を参照してください。

https://docs.oracle.com/en/database/oracle/oracle-database/19/bradv/rman-duplicating-databases.html#GUID-F31F9FCE-B610-49EB-B9DB-44B9AA4E838F

複製後、ターゲット・データベースを開いたら、さらにいくつかの構成手順を実行する必要があります。

1. 複製後、ターゲット・データベースを開いたときに、データベースがリカバリされた時点の SCN を記録する必要があります。この SCN は、Oracle GoldenGate Replicat プロセスを起動する後の手順で使用します。SCN はターゲット・データベースの alert.log またはデータ・ディクショナリから得ることができます。両方の例を以下に示します。

   SCN を alert.log エントリから取得します。

   ```
   RESETLOGS after incomplete recovery UNTIL CHANGE 681543060 time 05/01/2019
   15:05:20
   ```

   データ・ディクショナリの問合せを使って SCN を取得します。

   ```
   SQL> select RESETLOGS_CHANGE# - 1 from v$database;
   ```

注：このSCNを記録しておいてください。GoldenGate Replicatプロセスの起動時に後で使用します。

2. データベースのインスタンス化パラメータ ENABLE_GOLDENGATE_REPLICATION が TRUE に設定されていることを確認します。設定されていない場合は、次のコマンドを使って設定してください。

```
SQL> alter system set enable_goldengate_replication=TRUE scope=both;
```

Oracle Data Pumpエクスポート/インポート

Oracle Data Pump エクスポート/インポートは、ソース・データベースとターゲット・データベースのバージョンが異なり、ハードウェア・プラットフォームまたはデータベース構造が異なる場合に、ターゲット・データベースをインスタンス化するための代替方法になります。たとえば、非暗号化データベースや暗号化データベースに移行する場合です。

ターゲット・データベースはあらかじめ作成しておきます。Data Pump エクスポート/インポートには、ターゲット・データベースをインスタンス化するための多くのオプションがあります。

Data Pumu エクスポート/インポートを使用する場合には、重要な考慮点がいくつかあります。

- ターゲット・データベースのデータベース・リンク経由でソース・データベースから直接インポートする場合は、Data Pump インポートと NETWORK_LINK パラメータを使用して、ターゲット・データベースのインスタンス化にかかる時間を短縮することを検討してください。
- ソース・データベースをエクスポートするときには、GoldenGate 管理者をエクスポートしないでください。エクスポート・パラメータ EXCLUDE=SCHEMA:"='GGADMIN'"を持つユーザーは除外します。
- Data Pump エクスポート/インポートの進捗状況を V$SESSION_LONGOPS データ・ディクショナリ・ビューで監視します。
- Oracle GoldenGate でレプリケートするオブジェクトは前の手順ですでに準備してあるので、オブジェクトがターゲット・データベースにインポートされた後、Replicat は複製データを適用しなくても、適用すべき証跡ファイルのトランザクションを判別できます。Data Pump パラメータ FLASHBACK_SCN を使用する必要はありません。

Oracle Data Pump の使用方法については、Oracle Database ユーティリティのドキュメントを参照してください。

https://docs.oracle.com/cd/F19136_01/sutil/index.html

Oracle GoldenGateの構成を完了する

この項では、ターゲット・データベースのインスタンス化後に必要な Oracle GoldenGate の構成について説明します。

ターゲット・データベースのGoldenGate資格証明を作成する

Oracle Data Pump を使ってターゲット・データベースをインスタンス化した場合、前の手順「Oracle GoldenGate 管理者データベース・アカウントを作成する」で GoldenGate データベース管理者アカウントをすでに作成しているはずです

GoldenGate プロセスがソース・データベースとターゲット・データベースに接続するには、GoldenGate データベース資格証明が必要です。前述「ソース側の GoldenGate 資格証明を作成する」の項ですでに説明したコマンドを使って、ターゲット・データベースの GoldenGate 資格証明を作成します。

ターゲット・データベースがマルチテナント・データベースである場合、GoldenGate 資格証明は CDB ではなく、PDB に接続する必要があります。

ターゲット・データベースのGoldenGateチェックポイント表を作成する

チェックポイント表は、このホワイト・ペーパーで使用が推奨されている、統合されていないパラレル Replicat の必須コンポーネントです。

統合されていない Replicat は自身のリカバリ・チェックポイントを、ターゲット・データベースに格納されたチェックポイント表に保持しています。チェックポイントは、Replicat トランザクション内のチェックポイント表に書き込まれます。チェックポイントはトランザクションとともに成功または失敗するので、プロセスまたはデータベースの失敗があっても、Replicat は、トランザクションが 1 回しか適用されないようにします。

RMAN DUPLICATE を使ってターゲット・データベースを作成した場合は、次のコマンドを使ってチェックポイント表の存在を確認します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/commands/execute -X POST --data '{"name":
 "report","reportType": "checkpointTables","credentials": {"alias":
"target_atp1","domain": "goldengate"},"specification": "ggadmin.gg_checkpoints"}'|
python -m json.tool
```

チェックポイント表のエイリアス、ドメイン、仕様は、ターゲット・データベースに合わせて置き換えます。

チェックポイント表がすでに存在する場合、出力例は次のようになります。

```
    "response": {
        "$schema": "ogg:commandResult",
        "tables": [
            "GGADMIN.GG_CHECKPOINTS"
        ]
    }
```

チェックポイント表が存在しない場合、出力例は次のようになります。

```
    "response": {
        "$schema": "ogg:commandResult",
         "tables": []
    }
```

チェックポイント表がすでに存在する場合は、削除してから新しいチェックポイント表を作成します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/connections/goldengate.target/tables/checkpoint - X POST --data
'{"operation": "delete","name": "ggadmin.gg_checkpoints"}'| python -m
json.tool
```

チェックポイント表を作成します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/connections/goldengate.target/tables/checkpoint - X POST --data '{"operation":
"add","name": "ggadmin.gg_checkpoints"}'| python -m
json.tool
```

次に、表の作成を確認します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/connections/goldengate.target/tables/checkpoint -
```

```
X POST --data '{"operation": "info","name": "ggadmin.gg_checkpoints"}'| python -m  json.tool
```

返されたメッセージの例：

```
{
    "$schema": "ogg:message",
    "code":"OGG-08100",
    "issued":"2019-02-04T17:18:09Z",
    "severity":"INFO",
    "title":"Checkpoint table ggadmin.gg_checkpoints created 2019-02-04 17:17:36.",
    "type": http://docs.oracle.com/goldengate/c1910/gg-winux/GMESG/oggus.htm#OGG-08100
}
```

ターゲット・データベースにハートビート表を作成する

ソース・データベースとターゲット・データベース間のレプリケーションの待機時間を監視するには、GoldenGate ハートビート表が必要です。RMAN DUPLICATE を使ってインスタンス化した後、ターゲット・データベースに GoldenGate ハートビート・オブジェクトがすでに含まれている場合、オブジェクトをいったん削除してから再作成する必要があります。

次のコマンドを使って、ターゲットのハートビート表オブジェクトを削除します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/connections/goldengate.target /tables/heartbeat - X DELETE| python -m
json.tool
```

ターゲットのハートビート表オブジェクトを作成します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/connections/goldengate.target/tables/heartbeat -X
POST --data '{"targetOnly":true}'| python -m json.tool
```

Oracle GoldenGate Replicatを作成する

RMAN を使ってターゲット・データベースをインスタンス化した場合は、Replicat を作成する前に、データベース移行用にすでに作成した GoldenGate Extract を登録解除し、削除する必要があります。

次のコマンドを使って、Extract をデータベースから登録解除し削除します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/extracts/EXT1 -X DELETE | python -m json.tool
```

Replicat プロセスの作成は、ターゲット・データベースをどのようにインスタンス化したかによって異なります。RMAN または Data Pump によるインスタンス化に応じて、次の手順に従います。

REST API を使って Replicat を作成する場合、プロセス・パラメータ・ファイルが自動的に作成されるため、手動で作成する必要はありません。

Oracle RMANのインスタンス化のためにReplicatを作成する

次のサンプル・コマンドでは、統合されていないパラレル Replicat を作成します。

```
$ curl -s –K access.cfg https://gghub-server/TARGET
adminsrvr/services/v2/replicats/REP1 -X POST --data '{"credentials": {"alias":
 target","domain": "goldengate"},"source": {"path":
"/u01/oracle/goldengate/deployments/SOURCE/var/lib/data","name": "aa"},"begin":
{"sequence": 0,"offset": 0}, "checkpoint": {"table": "ggadmin.gg_checkpoints"},"mode":{"type":
"nonintegrated","parallel": true},"config":["Replicat REP1", "UseridAlias target DOMAIN
goldengate","MAP_PARALLELISM 4","MIN_APPLY_PARALLELISM 2", "MAX_APPLY_PARALLELISM 50", REPORTCOUNT
EVERY 15 MINUTES, RATE", "BATCHSQL", "Map pdb1.*.*"," Target
PDB1.*.*;"]}' | python -m json.tool
```

Replicat が作成されたことを確認します。

```
$ curl -s –K access.cfg https://gghub-
 server/TARGET/adminsrvr/services/v2/replicats/REP1 -X GET | python -m json.tool
```

Replicat を起動するには、前述の「Oracle RMAN 複製データベース」の項で記録した SCN が必要です。前述の例の SCN 値、681543060 と以下のサンプル・コマンドを使用して、Replicat を起動します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/commands/execute -X POST --data
'{"name":"start","processName":"REP1","AT":681543060}' | python -m json.tool
```

Replicat のステータスを確認します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/replicats/REP1/info/status -X GET | python -m json.tool
```

出力例：

```
    "response": {
            "$schema": "ogg:replicatStatus",
             "lag":0,
            "lastStarted":"2019-06-14T19:40:10.419Z",
            "position": {
                    "name": "aa",
                    "offset":190639772,
                    "path": "/u01/oracle/goldengate/deployments/SOURCE/var/lib/data/",
                     "sequence":5
            },
            "processId":55184,
            "sinceLagReported": 8,
            "status": "running"
    }
```

Data Pumpのインスタンス化のためにReplicatを作成する

ターゲット・データベースを Oracle Data Pump でインスタンス化する場合、Replicat プロセスの作成時に追加のパラメータが必要になります。ENABLE_INSTANTIATION_FILTERING パラメータは、Data Pump エクスポート・ファイルに記録されている、各表のインスタンス化 SCN より前にコミットされた証跡ファイル・データを除外し適用しないように Replicat に指示します。インスタンス化 SCN は、表のインポート時にターゲット・データベースのデータ・ディクショナリに保存されます。

Replicat を作成します。

```
$ curl -s –K access.cfg https://gghub-server/TARGET
adminsrvr/services/v2/replicats/REP1 -X POST --data '{"credentials": {"alias":
target","domain": "goldengate"},"source": {"path":
"/u01/oracle/goldengate/deployments/SOURCE/var/lib/data","name": "aa"},"begin":
{"sequence": 0,"offset": 0}, "checkpoint": {"table":
"ggadmin.gg_checkpoints"},"mode":{"type": "nonintegrated","parallel":
true},"config":["Replicat REP1", "UseridAlias target DOMAIN
goldengate","MAP_PARALLELISM 4","MIN_APPLY_PARALLELISM 2", "MAX_APPLY_PARALLELISM 50", REPORTCOUNT
EVERY 15 MINUTES, RATE", "BATCHSQL", "DBOPTIONS
ENABLE_INSTANTIATION_FILTERING","Map pdb1.*.*"," Target PDB1.*.*;"]}' | python –m json.tool
```

Replicat が作成されたことを確認します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/replicats/REP1 -X GET | python -m json.tool
```

Replicat を起動します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/commands/execute -X POST --data
'{"name":"start","processName":"REP1"}' | python -m json.tool
```

起動した Replicat のステータスを確認します。

```
$ curl -s –K access.cfg https://gghub-
 server/TARGET/adminsrvr/services/v2/replicats/REP1/info/status -X GET | python -m
json.tool
```

AUTOSTARTタスクをReplicatに割り当てる

Replicat プロセスが実行されるようになったので、次のサンプル・コマンドを使って、前述の「AUTOSTART タスクを作成する」の AUTOSTART タスクを Replicat に割り当てます。

```
$ curl -s –K access.cfg https://gghub-server/TARGET
/adminsrvr/services/v2/replicats/REP1 -X PATCH --data '{"managedProcessSettings":
"MIGRATE01"}' | python -m json.tool
```

Replicat に新しいプロファイルが構成されていることを確認します（部分的な出力を示します）。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/replicats/REP1 -X GET | python -m json.tool

      "managedProcessSettings":"MIGRATE01",
```

GoldenGateのレプリケーションを監視する

Extract と Replicat が両方とも実行されている状態で、ターゲット・データベースのインスタンス化の間に生成された、未処理のすべてのトランザクション・データが適用されるまで、GoldenGate のエンド・ツー・エンドの待機時間を監視し、適用後、ユーザーとアプリケーションを移行先データベースに切り替えることができます。

まず、Extract プロセスと Replicat プロセスの証跡ファイルの進捗状況を比較します。

Extract チェックポイントの進捗状況を表示します。

```
$ curl -s –K access.cfg https://gghub-server
/SOURCE/adminsrvr/services/v2/extracts/EXT1/info/checkpoints -XGET | python -m json.tool
```

出力例：

```
"current": {
"name": "aa",
      "offset":388314032,
      "path": "/u01/oracle/goldengate/deployments/SOURCE/var/lib/data/",
      "sequence":311,
      "sequenceLength": 9,
      "sequenceLengthFlip": false,
       "timestamp":"2019-02-08T21:43:36.660Z"
}
```

Replicat チェックポイントの進捗状況を表示します。

```
$ curl -s –K access.cfg https://gghub-server /SOURCE
/TARGET/adminsrvr/services/v2/replicats/REP1/info/checkpoints -XGET | python -m json.tool
```

出力例：

```
"current": {
"name": "aa",
      "offset":346955548,
      "path": "/u01/oracle/goldengate/deployments/SOURCE/var/lib/data",
      "sequence":311,
      "sequenceLength":9,
      "sequenceLengthFlip": false,
```

```
    "timestamp":"2019-02-08T21:48:12.906Z"
}
```

Replicat が Extract の書き込み先と同じ証跡ファイルから読み取りを行う際（チェックポイントのオフセットは類似）、Replicat は、ターゲット・データベースのインスタンス化の間に生成されたすべての証跡ファイルを適用します。

次に、ハートビート表を使って Oracle GoldenGate のエンド・ツー・エンドの待機時間を監視します。ラグ時間が許容できる程度に短くなるまで（2 秒未満など）、次のコマンドを実行します。

```
$ curl -s –K access.cfg https://gghub-server/TARGET
/adminsrvr/services/v2/connections/goldengate.target/tables/heartbeat/REP1 -XGET | python -m json.tool
```

出力例：

```
        "heartbeats": [
            {
                "ageSeconds":5.02,
                "lagSeconds":1.27,
                "path":"EXT1 ==> REP1",
                "source":"DSGG:PDB1",
                "target":"GGT:PDB1"
            }
        ]
```

移行先データベースのテスト

GoldenGate の適用を一時停止した後、ターゲット・データベースをテストできるように、保証付きリストア・ポイント（GRP）を作成します。テストが終わり、データベースを GRP にフラッシュバックすると、GoldenGate Replicat が再起動し、レプリケーションが再開します。すべてのテストが完了したら、GRP を削除できます。レプリケーションの一時停止、GRP の作成、テスト、データベースのフラッシュバック、レプリケーションの再開のサイクルは、何度でも実行できます。

レプリケーションを一時停止する

GoldenGate Replicat のプロセスを停止してレプリケーションを一時停止します。Extract は実行したままなので、ソース・データベースのデータを抽出し続けて、新しい証跡ファイル・データを作成します。

1. Replicat を停止します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/commands/execute -X POST --data
'{"name":"stop","processName":"REP1","force":false}' | python -m json.tool
```

2. Replicat のステータスを確認します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/replicats/REP1 -X GET | python -m json.tool
```

Replicat のステータスが完全に停止するまで、続行するのは危険です。

出力例：

```
"sinceLagReported": 14,
"status": "stopped"
```

保証付きリストア・ポイント（GRP）を作成する

ターゲット・データベース・データベースがマルチテナント・データベースである場合は、CDB から GRP を作成します。

CDB に SYSDBA ユーザーとして接続し、次の文を実行します。

```
SQL> CREATE RESTORE POINT <GRP name> FOR PLUGGABLE DATABASE <PDB name> GUARANTEE FLASHBACK
DATABASE;
```

単一テナント・データベースの場合は、次のコマンドを使ってリストア・ポイントを作成します。

```
SQL> CREATE RESTORE POINT <GRP name> GUARANTEE FLASHBACK DATABASE;
```

GRP の作成を確認します。

```
SQL> set linesize 140 pages 1000
SQL> col con_id format 9999
SQL> col pdb_restore_point format a20
SQL> col name format a20
SQL> col guarantee_flashback_database format a5
SQL> col scn format 999999999999
SQL> col time format a48
SQL> alter session set nls_date_format='dd-Mon-yyyy hh:mi:ss';
SQL> select con_id,name,scn,time,guarantee_flashback_database,pdb_restore_point from v$restore_point;
```

ターゲット・データベースを検証する

ターゲット・データベースをアプリケーションの検証やデータベース表に対する DML/DDL など健全性チェックに使用できるようになりました。テスト期間は通常、2 時間もかからないはずです。

移行を開始する前に、包括的なパフォーマンスと機能のテストを完了させておくことをお奨めします。そのような包括的なテストに現在の移行期間を使うことには、次のリスクが伴います。

1. 移行期間が長引く。
2. GoldenGate に必要な時間が長引いて、ソース・データベース上で絶えず生成されるトランザクションのレプリケートに追い付かなくなる可能性がある。
3. ターゲット・システムと移行全体に悪影響を及ぼす問題が生じる。

テスト中、「GoldenGate のレプリケーションを監視する」の項に詳述されているように、GoldenGate のラグとチェックポイントの相違（現在の Extract の証跡ファイルと最後に適用された Replicat の証跡ファイル）を監視し続けることをお奨めします

ターゲット・データベースをフラッシュバックする

テストが完了したら、ターゲット・データベースを閉じて、GRP にフラッシュバックし、再び開くことができます。

マルチテナント・データベースを実行している場合は、SYSDBA ユーザーとして CDB に接続している間、次のすべてのコマンドを実行する必要があります。

a. データベースをシャットダウンする。

単一テナントの場合：

```
SQL> shutdown immediate
```

マルチテナントの場合：

```
SQL> ALTER PLUGGABLE DATABASE <PDB name> CLOSE IMMEDIATE;
```

b. データベースをフラッシュバックする。

単一テナントの場合：

```
SQL> FLASHBACK DATABASE TO RESTORE POINT <GRP name>;
```

マルチテナントの場合：

```
SQL> FLASHBACK PLUGGABLE DATABASE <PDB name> TO RESTORE POINT <GRP name>;
```

c. データベースを起動する。

単一テナントの場合：

```
SQL> ALTER DATABASE OPEN RESETLOGS;
```

マルチテナントの場合：

```
SQL> ALTER PLUGGABLE DATABASE <PDB name> OPEN RESETLOGS;
```

データベースをフラッシュバックしたら、テストおよび同じ GRP へのフラッシュバックのサイクルを複数回繰り返すことができます。ただし、レプリケーションの一時停止が長ければ長いほど、レプリケーションに追い付くまでにかかる時間が長くなります。

保証付きリストア・ポイントを削除する

データベースを開いて、テストが完了したら、GRP を削除します。テスト・サイクルを複数回実行する場合は、テスト・サイクルを再び繰り返す前に、レプリケーションを再開し、GoldenGate が追い付くようにすることをお奨めします。

単一テナントの場合：

```
SQL> DROP RESTORE POINT <GRP name>;
```

マルチテナントの場合：

```
SQL> DROP RESTORE POINT <GRP name> FOR PLUGGABLE DATABASE <PDB name>;
```

レプリケーションを再開する

GoldenGate Replicat のプロセスを再起動して、レプリケーションを再開します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/commands/execute -X POST –data
'{"name":"start","processName":"REP1"}' | python -m json.tool
```

GoldenGate のチェックポイントとラグの監視を続け、テストが完了したら、最終的なスイッチオーバーに進みます。

移行先データベースにスイッチオーバーする

テストが完了したら、ターゲット・データベースにスイッチオーバーできます。

GoldenGateのレプリケーション・ラグが許容できる程度に小さいかどうかを判別する

許容可能なレプリケーション・ラグの小ささは、ソース・ワークロードとネットワーク待機時間に応じて大きく異なる場合があります。

GoldenGate Extract および Replicat の現在のチェックポイント位置とともにラグを監視して、両方のプロセスが同じ証跡ファイル内で動作していることを確認する必要があります。通常、同じ証跡ファイル内で動作している場合、Extract と Replicat 間の待機時間が少ないことを意味します。「GoldenGate のレプリケーションを監視する」の項で指定されているコマンドを使って、Extract と Replicat が許容可能なラグに達したときを判別します。

ソース・データベースでのトランザクションの開始を停止する

データ損失ゼロのスイッチオーバーを完了するには、SRVCTL を使ってソース・データベースのサービスを停止することで、ソース・データベース上のトランザクションの発生を阻止します。

ソース・データベース上で新しい接続とトランザクションを禁止した後、データベース上のすべてのアクティブなトランザクションが完了するまで監視します。

次に例を示します。

```
SQL> select XIDUSN, XIDSLOT, XIDSQN, STATUS, USED_UBLK, USED_UREC from v$transaction where RECURSIVE='NO';
```

行が返らない場合、ソース・データベースがアイドル状態であることを推察できます。

Extractで未処理のトランザクションが完了したことを確認する

ソース・データベース上ですべてのトランザクションが停止したら、Extractを監視して、未処理の全トランザクションの抽出に確実に追い付くようにします。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/extracts/EXT1/info/checkpoints -XGET | python -m json.tool
```

Extractの出力のチェックポイント位置はあまり変わらないはずで、GoldenGateハートビート表によってのみ5分ごとに更新されます。

Extractを停止する

Extract がこれ以上前に進まないことがわかったら、Extract を停止できます。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/commands/execute -X POST --data
'{"name":"stop","processName":"EXT1","force":true}' | python -m json.tool
```

ステータスが停止したことを確認します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/extracts/EXT1 -X GET | python -m json.tool|grep status
```

Replicatがすべての証跡ファイル・データを適用するまで待機する

Extract が停止したら、Replicat が未処理の全証跡ファイル・データを適用するまで待つ必要があります。それには、LOGEND コマンドで Replicat を監視します。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/replicats/REP1/command -X POST --data
'{"command":"LOGEND"}' | python -m json.tool
```

出力例：

```
"replyData": {
        "$schema": "er:logEndResult",
        "allRecordsProcessed": true
}
```

Replicat が証跡ファイル・データをすべて適用すると、“true”の値の allRecordsProcessed が返されます。

ターゲット・データベースにスイッチオーバーする

レプリケートされたすべてのデータがターゲット・データベースに適用されたので、 Replicatを停止することができます。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/commands/execute -X POST --data
'{"name":"stop","processName":"REP1","force":true}' | python -m json.tool
```

Replicat が停止したら、データベース・クライアントをターゲット・データベースにスイッチオーバーします。

移行プロセス中、ソース・データベースとターゲット・データベースの両方で自由に読取り/書込みを実行できるので、アプリケーション・データベース・サービスを有効にし、ターゲット・サービスを使用するようにアプリケーションを切り替える作業は、システム管理者またはデータベース管理者の判断で行われます。読み取り専用アプリケーションの場合、GoldenGate Replicat が未処理のソース・トランザクションすべてを適用して停止すると、スイッチオーバーが即座に実行されるので、これらのサービスに対しアプリケーションの停止時間をゼロにすることができます。

読取り/書込み可能なアプリケーションについては、最新データが確実に操作されるよう、アプリケーションをスイッチオーバーする前に、すべてのトランザクションがターゲット・データベースに適用済みであることを確認しなければならない場合があります。

必要な場合は、Oracle GoldenGate Veridata を使って、移行先データベースにスイッチオーバーする前に、同期外れのデータを見つけて修復することができます。詳しくは、Oracle GoldenGate Veridata のドキュメントを参照してください。

https://docs.oracle.com/cd/E86974_01/books.html

GoldenGateの構成の削除

Oracle データベースの移行後、次のコマンドを使って GoldenGate の構成を削除できます。

GoldenGateプロセスを削除する

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/replicats/REP1 -X DELETE | python -m json.tool

$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/extracts/EXT1 -X DELETE | python -m json.tool
```

次のコマンドを使って証跡ファイルを削除します。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/commands/execute -X POST --data '{"name":
"purge","purgeType": "trails","trails":[{"name":"aa"}],"useCheckpoints":
false,"keep": [{"type": "min","units": "files","value": 0}]}' | python -m json.tool
```

AUTOSTARTタスクを削除する

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/config/types/ogg:managedProcessSettings/values/ogg:managedProcessSettings:
MIGRATE01 -XDELETE | python -m json.tool
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/config/types/ogg:managedProcessSettings/values/ogg:managedProcessSettings:M
IGRATE01 -XDELETE | python -m json.tool
```

ターゲット・データベースのハートビート表を削除する

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/connections/goldengate.source_pdb/tables/heartbea t -X DELETE| python -m
json.tool
```

移行の一環としてハートビート表をソース・データベース内に作成した場合は、そのハートビート表も削除する必要があります。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/connections/goldengate.source_pdb/tables/heartbea t -X DELETE| python -m
json.tool
```

チェックポイント表を削除する

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr//services/v2/connections/goldengate.target/tables/checkpoint
-X POST --data '{"operation": "delete","name": "ggadmin.gg_checkpoints"}'| python –m json.tool
```

Oracle GoldenGate資格証明を削除する

Oracle GoldenGate 資格証明を削除しても、ユーザー・アカウントはデータベースから削除されません。

```
$ curl -s –K access.cfg https://gghub-
server/TARGET/adminsrvr/services/v2/credentials/goldengate/target -X DELETE | python-m json.tool
```

次の例は、マルチテナント・ソース・データベースの場合です。

```
$ curl -s –K access.cfg https://gghub-
server/SOURCE/adminsrvr/services/v2/credentials/goldengate/source_cdb -X DELETE | python -m json.tool

$ curl -s –K access.cfg https://gghub-
serverSOURCE/adminsrvr/services/v2/credentials/goldengate/source_pdb -X DELETE | python -m json.tool
```

付録A – Oracle GoldenGateデプロイメント作成のレスポンス・ファイルの例

ほとんどの構成変数は、デプロイメントおよび GoldenGate プロセス・ファイルが格納されているファイル・システム・ディレクトリに関連付けられているため、これらのディレクトリが作成済みのディレクトリと一致していることが重要です。

最新の GoldenGate インストール・ソフトウェアの適切なレスポンス・ファイルを使用して、パラメータが正しく指定されていることを確認してください。

次のサンプル・レスポンス・ファイルでパラメータを変更する必要があります。

Section A - General

DEPLOYMENT_NAME=TARGET *# Make sure this is a unique deployment name*

Section B - Administrator Account

ADMINISTRATOR_USER=admin *# GG deployment administration account, NOT a database account.*

ADMINISTRATOR_PASSWORD=*password* *# Note: you MUST specify a password*

Section C - Service Manager

 # File system directory for the Service Manager deployment.This only gets set for the first deployment creation.

SERVICEMANAGER_DEPLOYMENT_HOME=/mnt/acfs_gg/deployments/ggsm

HOST_SERVICEMANAGER=127.0.0.1 *# Leave as localhost (127.0.0.1)*

PORT_SERVICEMANAGER=9100 *# Set to a unused port number, unique for this deployment.For the second # deployment creation, set this to the port# used for the initial deployment # creation.*

CREATE_NEW_SERVICEMANAGER=true *# Only set to a true for first deployment creation.*

Section D - Software Home

OGG_SOFTWARE_HOME=/app/goldengate/target *# GoldenGate software location*

Section E - Deployment Directories

OGG_DEPLOYMENT_HOME=/ mnt/acfs_gg/deployments/target *# Deployment name should be unique*
OGG_ETC_HOME=/mnt/acfs_gg/deployments/target/etc
OGG_CONF_HOME=/mnt/acfs_gg/deployments/target/etc/conf
OGG_SSL_HOME=/mnt/acfs_gg/deployments/target/etc/ssl
OGG_VAR_HOME=/mnt/acfs_gg/deployments/target/var
OGG_DATA_HOME=/mnt/acfs_gg/deployments/target/var/lib/data

Section F - Environment Variables

ENV_ORACLE_HOME=/u01/app/oracle/product/19.1/client

ENV_LD_LIBRARY_PATH=${ORACLE_HOME}/lib:/u01/app/oracle/product/19.1/client/lib:/u01/app/oracle/product/19.1/client/rdbms/lib:/u01/app/oracle/product/19.1/client/jdbc/lib:/u01/oracle/goldengate/target/lib

ENV_TNS_ADMIN=/u01/app/oracle/network/admin *# Use a single TNS_ADMIN for all deployments*

Section I - Services

PORT_ADMINSRVR=9101 *# Set to an unused port number, unique for this deployment.*
DISTRIBUTION_SERVER_ENABLED=false *# Not using due to local trail files.*
PORT_DISTSRVR=9102 *# Set to an unused port number, unique for this deployment.*
RECEIVER_SERVER_ENABLED=false *# Not using due to local trail files.*
PORT_RCVRSRVR=9103 *# Set to an unused port number, unique for this deployment.*
PORT_PMSRVR=9104 *# Set to an unused port number, unique for this deployment.*
UDP_PORT_PMSRVR=9105 *# Set to an unused port number, unique for this deployment.*

Section J - REPLICATION OPTIONS

OGG_SCHEMA=ggadmin *# Set to schema used for GG administrator in the database.MAKE SURE*
 # SOURCE & TARGET DATABASE SCHEMA NAMES MATCH.

2 番目のデプロイメントについては、次の主要パラメータの相違を含めて別のディレクトリ、ポート番号を指定した、2 番目のレスポンス・ファイルを作成します。

Section A - General

DEPLOYMENT_NAME=SOURCE *# Make sure this is a unique deployment name, should use SOURCE or*
 # TARGET

Section C - Service Manager

SERVICEMANAGER_DEPLOYMENT_HOME= *# Leave blank for subsequent deployment creations.*

HOST_SERVICEMANAGER=127.0.0.1 *# Leave as localhost (127.0.0.1)*

PORT_SERVICEMANAGER=9100 *# Set this to the port# used for the initial deployment creation.*

CREATE_NEW_SERVICEMANAGER=false *# Set to false because the initial deployment will have already*
 # created the Service Manager.

Section D - Software Home

OGG_SOFTWARE_HOME=/app/oracle/goldengate/source *# GoldenGate software location specified*
 # above in step #2 for the correct database compatibility

Section E - Deployment Directories

OGG_DEPLOYMENT_HOME=/mnt/acfs_gg/deployments/source *# Store on the encrypted file system.*
 # Should be either 'source' or 'target'.

OGG_ETC_HOME=/mnt/acfs_gg/deployments/source/etc
OGG_CONF_HOME=/mnt/acfs_gg/deployments/source/etc/conf
OGG_SSL_HOME=/mnt/acfs_gg/deployments/source/etc/ssl

OGG_VAR_HOME=/mnt/acfs_gg/deployments/source/var

OGG_DATA_HOME=/mnt/acfs_gg/deployments/source/var/lib/data

Section F - Environment Variables

ENV_ORACLE_HOME=/u01/app/oracle/product/12.2/client *# Assuming the source database is 12.2,*
 # but this may differ depending on the source database
 # version

ENV_LD_LIBRARY_PATH=${ORACLE_HOME}/lib:/u01/app/oracle/product/19.1/client/lib:/u01/app/oracle/product/19.1/client/rdbms/lib:/u01/app/oracle/product/19.1/client/jdbc/lib:/u01/oracle/goldengate/source/lib

ENV_TNS_ADMIN=/u01/app/oracle/network/admin *# Use a single TNS_ADMIN for all deployments*

Section I - Services

PORT_ADMINSRVR=9111 *# Set to an unused port number, unique for this deployment*

DISTRIBUTION_SERVER_ENABLED=false

PORT_DISTSRVR=9112 *# Set to an unused port number, unique for this deployment*

RECEIVER_SERVER_ENABLED=false

PORT_RCVRSRVR=9113 *# Set to an unused port number, unique for this deployment*

PORT_PMSRVR=9114 *# Set to an unused port number, unique for this deployment*

UDP_PORT_PMSRVR=9115 *# Set to an unused port number, unique for this deployment*

付録B – Oracle GoldenGateデプロイメントのスクリプトの起動と停止の例

Oracle GoldenGateデプロイメントの起動

サービス・マネージャが最後に停止したときにデプロイメント・サービスが実行されていなかった場合を除き、サービス・マネージャが起動するとデプロイメントは自動的に起動します。

サービス・マネージャを起動して 2 つのデプロイメントを起動するには、次の環境変数を設定します。

```
export OGG_HOME=/app/oracle/goldengate/target
export OGG_ETC_HOME=/mnt/acfs_gg/deployments/ggsm/etc
export OGG_VAR_HOME=/mnt/acfs_gg/deployments/ggsm/var
export PATH=$OGG_HOME/bin:$PATH
export JAVA_HOME=$OGG_HOME/jdk
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OGG_HOME/lib
```

次に、サービス・マネージャのデプロイメント・ディレクトリにある startSM.sh スクリプトを実行します。

次に例を示します。

```
/mnt/acfs_gg/deployments/ggsm/bin/startSM.sh
```

注：startSM.shスクリプトの場所は、初期GoldenGateデプロイメントの作成テンプレート・ファイル（SERVICEMANAGER_DEPLOYMENT_HOME）で指定された場所と同じです。

Oracle GoldenGateデプロイメントの停止

デプロイメントでサービス・マネージャが現在実行中のときにそのデプロイメントを停止すると、サービス・マネージャによってデプロイメントが自動的に再起動されます。そのため、サービス・マネージャが実行されているデプロイメントを停止してから 2 番目のデプロイメントを停止することが重要です。

どちらのデプロイメントがサービス・マネージャを実行しているかを判別してください。

```
$ ps -ef | grep ServiceManager | grep -v grep
oracle 6240 1 0 Mar21 ?00:09:19 /app/oracle/goldengate/target/bin/ServiceManager – quiet
```

この例の場合、サービス・マネージャは、/app/oracle/goldengate/target をホームとする Oracle GoldenGate ソフトウェアから起動されています。これは、以下で指定する OGG_HOME ディレクトリです。

サービス・マネージャが実行されているデプロイメントに合わせて環境変数を設定します（サービス・マネージャを起動したときの環境変数と同じではありません）。

```
export OGG_HOME=/app/oracle/goldengate/target
export OGG_ETC_HOME=/mnt/acfs_gg/deployments/target/etc
export OGG_VAR_HOME=/mnt/acfs_gg/deployments/target/var
export PATH=$OGG_HOME/bin:$PATH
```

```
export JAVA_HOME=$OGG_HOME/jdk
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OGG_HOME/lib
```

それから、次のスクリプト（stopDeployment.sh $OGG_HOME）でデプロイメントを停止します。

```
#!/bin/bash
# test if we have one arguments on the command line

if [ $# != 1 ]
then
    echo " "
    echo "Usage: ./stopDeployment.bsh <GG Home Directory>"
    echo " "
    exit 1
fi

oggHome=$1

# Check oggHome exsists!
if [ ! -d ${oggHome} ]
then
    echo " "
    echo "*** ERROR:Directory ${oggHome} does NOT exist!"
    echo " "
    exit 1
fi
if [[ -z "${OGG_VAR_HOME}" ]]; then
        echo "Error:OGG_VAR_HOME environment variable is not set."
        exit
fi

function getPID {
        PID=$(cat "${OGG_VAR_HOME}/run/ServiceManager.pid" 2>/dev/null)
}

function isRunning {
        kill -0 ${PID} 2>/dev/null
}

##
##     Check if any OGG components are running
##
function isOGGRunning {
        local pgrep=$(command -v pgrep 2>/dev/null)
        local pattern=${oggHome}/bin/${OGGProcesses}
        if [[ ! -z ${pgrep} ]]; then
                pgrep -f ${pattern} &>/dev/null
        else
                ps -aef | egrep ${pattern} &>/dev/null
```

```
        fi
}

##
##     Task: runCleanStop
##     Stop all OGG services
##
function runCleanStop {
        local timeout=10
         while (true); do
                pkill -f ${oggHome}/bin/'(adminsrvr|distsrvr|pmsrvr|recvsrvr|ServiceManager)' isOGGRunning || break
                sleep 1
                timeout=$(expr ${timeout} - 1)
                if [ ${timeout} -eq 0 ]; then
                        return 1
                fi
        done
        return 0
}

runCleanStop
getPID
if (! isRunning); then
        echo "Service Manager is not running"
        exit
fi

echo "Stopping Service Manager process (PID: ${PID})..."
pkill -P ${PID}

for attempt in $(seq 10); do
         if (! isRunning); then
                echo "Service Manager stopped"
                exit 0
        fi
        sleep 1
done
echo "Service Manager process could not be stopped"
exit 1
```

後に続くデプロイメントを停止するには、後続のデプロイメントのディレクトリ（OGG_ETC_HOME および OGG_VAR_HOME）に合わせて環境変数を設定します。

```
export OGG_HOME=/app/oracle/goldengate/source
export OGG_ETC_HOME=/mnt/acfs_gg/deployments/deployments/source/etc
export OGG_VAR_HOME=/mnt/acfs_gg/deployments/source /var
export PATH=$OGG_HOME/bin:$PATH
```

```
export JAVA_HOME=$OGG_HOME/jdk
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$OGG_HOME/lib
```

次に、サービス・マネージャのデプロイメントを停止するために使った同じ stopDeployment.sh スクリプトを実行します。

Oracle Corporation, World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065, USA

海外からのお問い合わせ窓口
電話：+1.650.506.7000
ファクシミリ：+1.650.506.7200

ORACLE

Oracle GoldenGate Hub構成による
Oracle Databaseの移行
2019年8月

著者：Stephan Haisley

Integrated Cloud Applications & Platform Services

CONNECT WITH US

blogs.oracle.com/oracle
facebook.com/oracle
twitter.com/oracle
oracle.com

Oracle is committed to developing practices and products that help protect the environment