PL/SQLを使用したOracle GoldenGate Microservicesのオーケストレーション

目次

1 はじめに

Oracle GoldenGate (OGG) Microservices Architectureは、OGG環境の一部としてRESTに対応したサービスを提供する新しいアーキテクチャです。本書は、読者がOGG Microservices Architectureの概要と機能に精通していることを前提としています。REST対応のサービスによって、HTML5 Webページ、コマンドライン、APIを使用したリモートでの構成、管理、監視が可能になります。

> *本書に記載される概要、スクリプト、情報は、学習のみを目的としており、Oracle DevelopmentまたはOracle Supportではサポートされません。また、本書のために使用されたテスト・システム以外のいかなる環境においても、機能は保証されません。本書に記載される変更を適用する前に、徹底的なテストを実施し、機能とパフォーマンスを評価する必要があります。*

PL/SQLを使用してREST APIのサービスを起動することで、OGG Microservicesを管理することもできます。UTL_HTTPは、PL/SQLを使用したHTTP(S)呼出しを可能にするPL/SQLパッケージです。このデモはUTL_HTTP上に構築されており、OGGデプロイメントへのREST呼出しを行います。

本書では、ソース・データベースとターゲット・データベース間の一方向レプリケーションと双方向レプリケーションを設定する方法の概要を説明しています。本書の例では、オーケストレーション・データベースを利用してPL/SQL経由でREST呼出しを行い、レプリケーションを設定していますが、別個のオーケストレーション・データベースを使用するかどうかは任意です。

例では、Extract、Distribution path、Replicatなど、あらゆるレプリケーション・コンポーネントを作成する方法と、それらをリモート・オーケストレーション・データベースから起動する方法を示しています。

Extractの停止やExtractパラメータ・ファイルの変更など、作成したコンポーネントの管理や問合せの概要についても説明しています。

このデモで使用する設定の概要

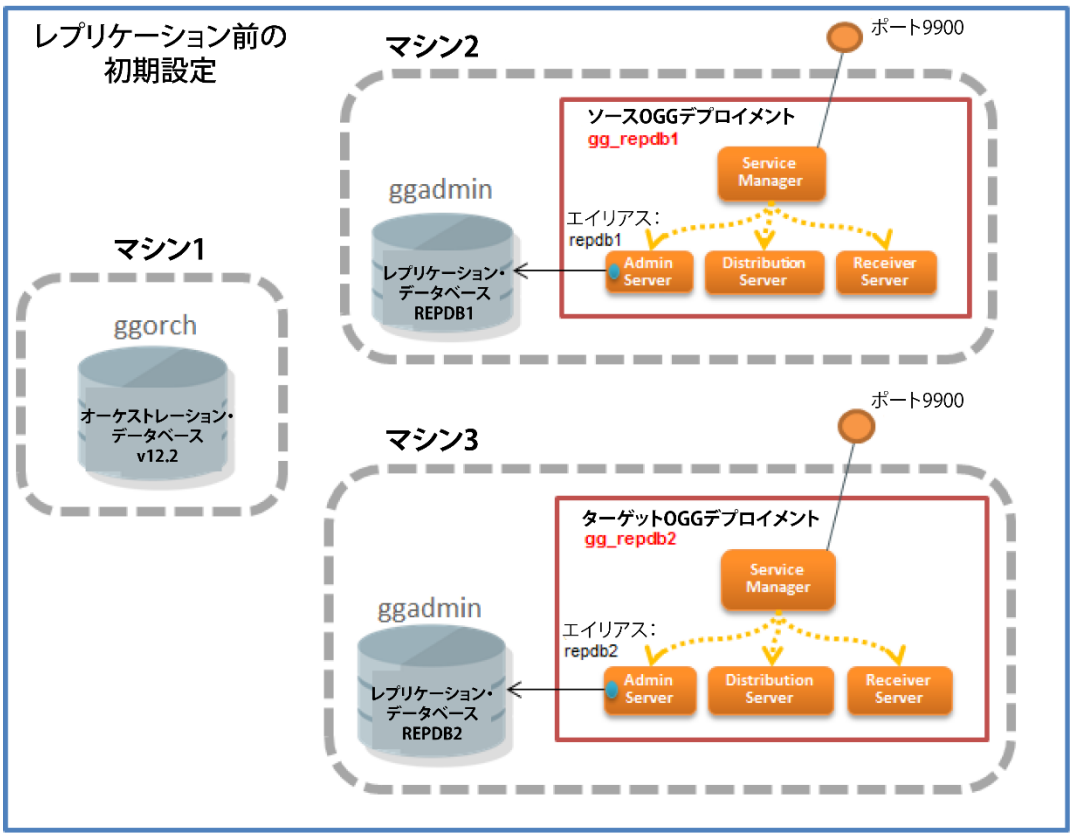

図1

図1では以下の名称を使用しています。

ggorch	オーケストレーション・データベースの OGGオーケストレーション・ユーザー
ggadmin	レプリケーション・データベースの OGGレプリケーション管理ユーザー
repdb1、***repdb2***	レプリケーション・データベースの名前
ポート ***9900***	OGGデプロイメントのService Managerポート
gg_repdb1、***gg_repdb2***	OGGデプロイメントのデプロイメント名
エイリアス：***repdb1***、エイリアス：***repdb2***	OGGデプロイメントの資格証明のエイリアス名

2 前提条件

デモのオーケストレーション・スクリプトは、図1で示すように、以下の設定を前提としています。

- マシン1にはバージョン12.2以降のデータベースが搭載されています。デモ・スクリプトはこのオーケストレーション・データベースにインストールされます。
- マシン2にはレプリケーション・データベースが設定され、Oracle GoldenGate Microservicesがデプロイされています。
- マシン3にはレプリケーション・データベースが設定され、Oracle GoldenGate Microservicesがデプロイされています。

注：デモを実行するために、データベースとOracle GoldenGateデプロイメントのすべての設定を同じマシンでホストさせることが可能です。

3 背景

3.1 OGGの新しいアーキテクチャ

Oracle GoldenGate Microservices Architecture（Oracle GoldenGate MA）は、Representational State Transfer Application Programming Interface（REST API）を基盤としています。Webベースのインタフェース、コマンドライン、またはREST APIを使用して、Oracle GoldenGateの各サービスを構成、監視、管理できます。

これまでのOGGアーキテクチャと比較して、OGG Microservices Architectureには、Service Manager、Administration Server、Distribution Server、Receiver Serverなど、さまざまなプロセスがあります（Oracle GoldenGate Microservices Architectureのコンポーネントを参照）。

3.2 OGGのインストールとデプロイメント

Oracle GoldenGate Microservices Architectureのスタート・ガイドでは、新しいアーキテクチャを使用してレプリケーションを設定する手順を詳しく説明しています。デプロイメントは、セキュアなhttpsを基盤にすることも、非セキュアなhttpを基盤にすることもできます。OGG REST APIサービスのセキュリティ・モデルには、認証と認可の両方が必要です。セキュリティ・モデルの詳細については、セキュアなデプロイメントまたはセキュアでないデプロイメントの設定およびOracle GoldenGate環境の保護を参照してください。

セキュアなデプロイメントでは、クライアントが安全にサーバーに接続できるよう、Verisignによって発行された認証局（CA）証明書を使用しています。

注：

a) Verisign以外のCAによって発行された証明書を使用することもできます。

b) 自己署名証明書も使用できます。

ORAPKIを使用すると、自己署名証明書の作成を試すことができます。

本書では、認証と認可に基づく証明書用に構成されたセキュアなOGGデプロイメント（HTTPS）向けのサンプル・コードのみを提供しています。

3.3 OGG REST API

OGG REST APIを使用すると、OGGデプロイメントとレプリケーション・サービスを管理できます。OGGコンポーネントの作成、変更、削除、または問合せを行うことができます。REST APIを使用するための前提条件については、OGGドキュメントのQuickStartセクションを参照してください。

Oracle GoldenGateの各サービス（Service Manager、Administration Server、Distribution Server、Receiver Server）は、異なるポートで実行されます。OGG RESTリソースには、以下のURL構造を使用してアクセスできます。

*http://localhost:port/*または*https://localhost:port/*（localhostおよびportは、OGGサービスが実行されているホストとポートです）。

OGGは、セキュアな設定（HTTPS）では、ポート443のリバース・プロキシを使用して構成することもできます。リバース・プロキシを使用する場合、個々のサービスのポートを覚える必要はありません。

OGG REST APIの詳細については「REST APIについて」を、リバース・プロキシ設定の詳細についてはリバース・プロキシ・サポートを参照してください。

4 オーケストレーション・スクリプトのサンプル

サンプル・コードの構成は以下のとおりです。

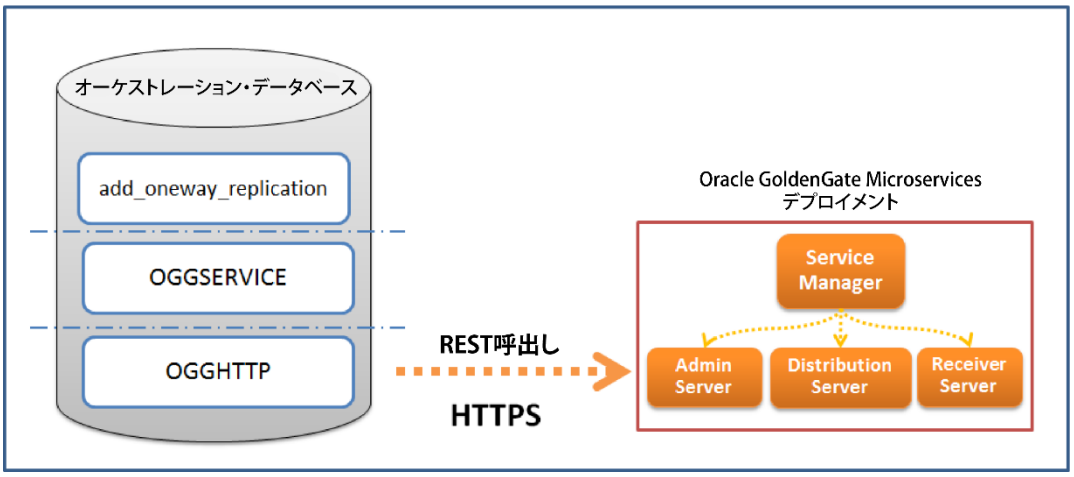

図2

4.1 ADD_ONEWAY_REPLICATION – 使用方法

ADD_ONEWAY_REPLICATION	指定した表のソース・データベースとターゲット・データベース間の単純な一方向レプリケーションを設定するサンプルのスタンドアロンPL/SQLファンクションです。OGGSERVICEから提供される下位のファンクションを使用します。
OGGSERVICE	REST呼出しを使用して、さまざまなアクションを所定のOGGデプロイメントで実行できる方法を示すパッケージです。実行できるアクションは以下のとおりです。 – コンポーネントの作成 – コンポーネントの変更 – コンポーネントの削除 – コンポーネントの問合せこれにはOGGHTTPパッケージが使用されます。 各アクションでは以下が実行されます。 1. アクションのURLの構成 2. アクションのペイロードの構成 3. URLとペイロードを使用したHTTP呼出し 4. レスポンスの処理
OGGHTTP	UTL_HTTPを使用してセキュアなHTTP呼出しを行う方法を示すパッケージです。

ADD_ONEWAY_REPLICAIONは、指定した表のソース・データベースとターゲット・データベース間の単純な一方向レプリケーションを設定するサンプルのスタンドアロンPL/SQLファンクションです。OGGSERVICEから提供される下位のファンクションを使用します。

3つのマシンがあることを前提に、レプリケーションの設定について説明しています（上記の**図1**を参照）。

オーケストレーションDBサイト

- オーケストレーション・データベースは、サンプル・オーケストレーション・スクリプトがデータベース・ユーザーGGORCHとしてインストールされた場所に作成されます。
- OGGデプロイメントに接続するためのクライアント証明書は、$ORACLE_HOME/admin/ggorch_walletディレクトリの下に置かれたウォレット内にあります。クライアント証明書で指定されたユーザー名には、以下に示すREPDB1サイトとREPDB2サイトの両方のOGGデプロイメントに対してREST API呼出しを行う権限があります。

REPDB1サイト

- レプリケーション・データベース***repdb1***が、レプリケーション・ソースとして機能するように設定されています
- セキュアなOGGデプロイメントが、ポート***9900***をリッスンするService Managerを使用して設定されています
- OGGデプロイメントの管理サーバー資格証明ストアで、dblogin資格証明エイリアス***repdb1***が、ドメインを使用してOracleGoldenGateとして作成されています
- OGGデプロイメント***gg_repdb1***が作成されています

REPDB2サイト

- レプリケーション・データベース***repdb2***が、レプリケーション・ターゲットとして機能するように設定されています
- セキュアなOGGデプロイメントが、ポート***9900***をリッスンするService Managerを使用して設定されています
- OGGデプロイメントの管理サーバー資格証明ストアで、dblogin資格証明エイリアス***repdb2***が、ドメインを使用してOracleGoldenGateとして作成されています
- OGGデプロイメント***gg_repdb2***が作成されています

以下の手順をオーケストレーション・データベースで実行し、GGORCHユーザーを構成します。

```
SQL> connect sys as sysdba
SQL> grant dba to ggorch identified by <password>;
SQL> @oggorch_setup ggorch <absolute_path_to_client_certificate>
SQL> connect ggorch
SQL> @orchestration_db_setup.sql
```

*注：<**absolute_path_to_client_certificate**>は、$ORACLE_HOME/admin/ggorch_walletディレクトリがオーケストレーション・データベースに作成され、クライアント・ウォレットがそのディレクトリにコピーされている場合は省略できます。これは、クライアント・ウォレットを検索するスクリプトによって使用されるデフォルト・ディレクトリです。*

4.1.1 一方向レプリケーションの設定

add_oneway_replicationファンクションでは、***repdb1***データベースと***repdb2***データベース間のSCOTTスキーマとHRスキーマの一方向レプリケーションを設定します。

このファンクションに対するおもな入力は以下のとおりです

- ソースOGGデプロイメント
- ソース・データベースのOGGエイリアス
- ターゲットOGGデプロイメント
- ターゲット・データベースのOGGエイリアス
- レプリケートされる表のリスト

ターゲット表をインスタンス化するための入力をオプションで受け取ることもできます（add_oneway_replicationを参照）。このファンクションは、REST呼出しを使用して、OGGデプロイメントで実行されるAdministration Server、Distribution Server、Receiver Serverに関する情報を取得します。その後、REST呼出しを使用して、リモート・オーケストレーション・データベースから以下のレプリケーション・コンポーネントを作成します。

- ソースOGGデプロイメントのExtract
- ソースOGGデプロイメントとターゲットOGGデプロイメント間のDistribution path
- ターゲットOGGデプロイメントのReplicat

ExtractとReplicatのパラメータ・ファイルは、提供された表のリストをレプリケートするように構成されます。

REPDB1サイトのOGGデプロイメント

- Extractを作成して起動します
- REPDB2サイトのOGGデプロイメントへのDistribution pathを作成して起動します。

REPDB2サイトのOGGデプロイメント

- Replicatを作成して起動します

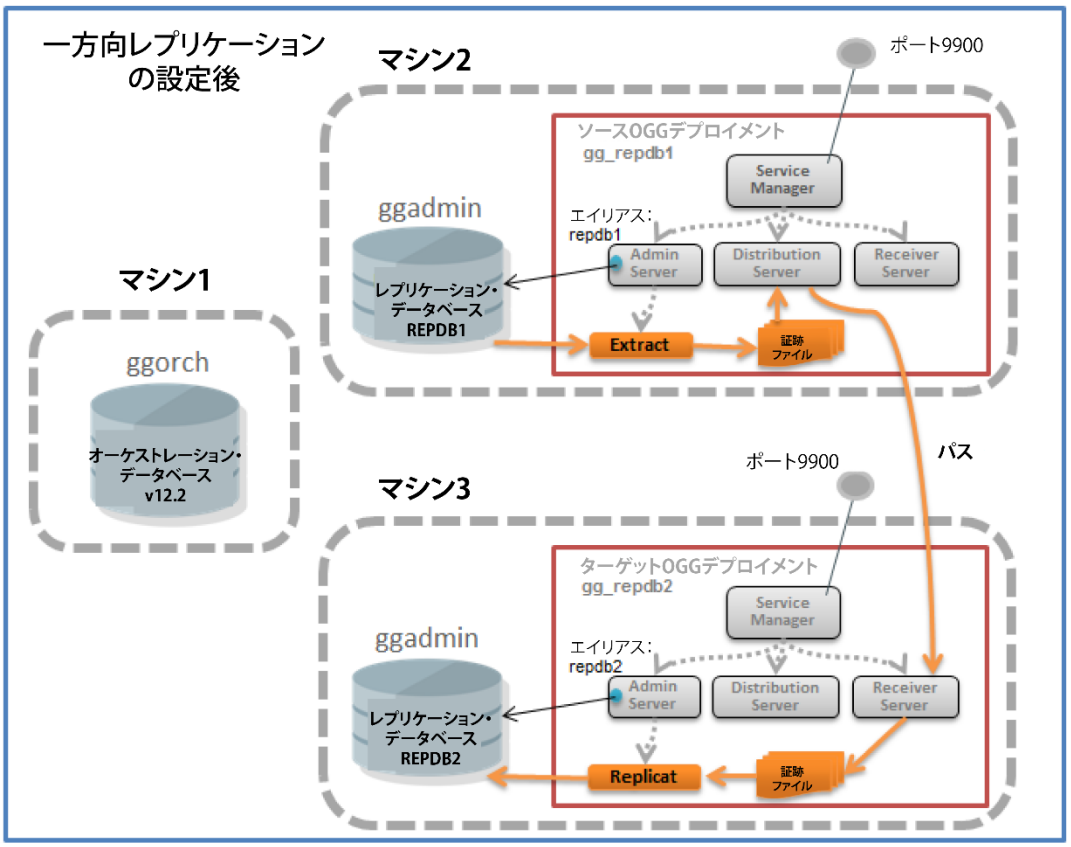

図3

repdb1データベースからrepdb2データベースへの一方向レプリケーションの作成。

以下のコードをオーケストレーション・データベースでGGORCHとして実行する必要があります（add_oneway_replicationを参照）。このコードでは、エイリアス***repdb1***がソースとして参照するデータベースと、エイリアス***repdb2***がターゲットとして参照するデータベース間の一方向レプリケーションを作成します。この設定により、ソースからターゲットに'scott'スキーマと'hr'スキーマがレプリケートされます。

```
set serverout on
DECLARE
  response_info clob; ret
  number;
BEGIN
  ret := add_oneway_replication(
           ogg_source                          => ogg_connection('&machine2', 9900, 'gg_repdb1'),
           source_dblogin_credential_alias     => 'repdb1',
           ogg_target                          => ogg_connection('&machine3', 9900, 'gg_repdb2'),
           target_dblogin_credential_alias     => 'repdb2',
           tables                              => '"scott.*","hr.*"',
           response_info                       => response_info);
END;
/
```

以下の出力には、Extract ***EFIML***、Distribution path ***PFIML***、およびReplicat ***RFIML***が作成されたことが表示されています。ExtractとDistribution pathはSOURCE OGG DEPLOYMENTで'running'と表示され、ReplicatはTARGET OGG DEPLOYMENTで'starting'と表示されています。

出力

```
---------------- REPLICATION:REPDB1 =>REPDB2 -------------------
---------------- EXTRACT -------------------
EXTRACT (Integrated) added.
Extract EFIML successfully registered with database at SCN 4623804.
EXTTRAIL added.
EXTRACT EFIML starting
EXTRACT EFIML started


---------------- DISTRIBUTION PATH -------------------
The path 'PFIML' has been added.


---------------- REPLICAT -------------------
REPLICAT (Integrated) added.
REPLICAT RFIML starting
REPLICAT RFIML started


---------- SOURCE OGG DEPLOYMENT ---------

Group   Status       Program
EFIML : ["running"] - EXTRACT
PFIML : ["running"] - PATH


---------- TARGET OGG DEPLOYMENT ---------

Group   Status       Program
RFIML : ["starting"] - REPLICAT
```

*注：それぞれの名前はExtractが**E<suffix>**、Replicatが**R<suffix>**、Distribution pathが**P<suffix>**のようになります（**<suffix>**はランダムに生成された4文字の文字列です）。*

4.1.2 双方向レプリケーションの設定

上記の設定で、ソースとターゲットの情報を交換し、***add_oneway_replication***に対して別の呼出しを行うと、***repdb1***データベースと***repdb2***データベース間のSCOTTスキーマとHRスキーマの双方向レプリケーションを設定できます。逆方向のレプリケーションを作成するための追加の呼出しは以下のとおりです。

REPDB2サイトのOGGデプロイメント

- Extractを作成して起動します
- REPDB1サイトのOGGデプロイメントへのDistribution pathを作成して起動します。

REPDB1サイトのOGGデプロイメント

- Replicatを作成して起動します

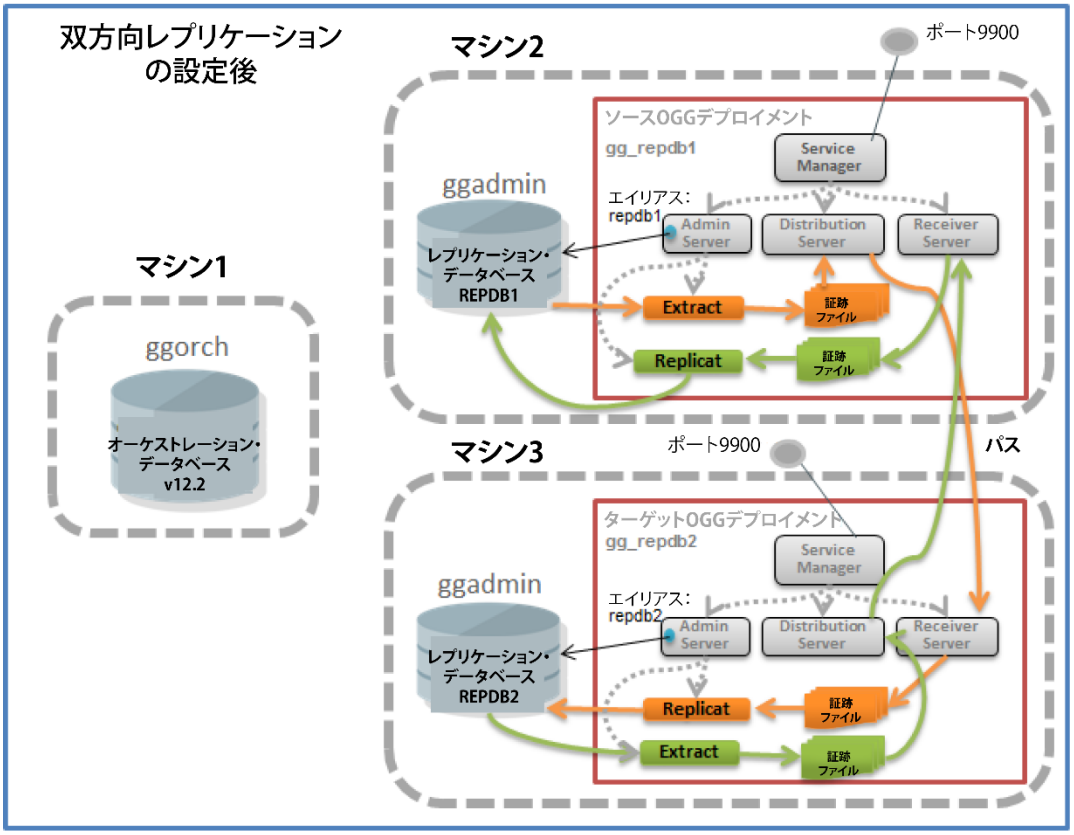

図4

repdb2データベースからrepdb1データベースへの双方向レプリケーション。

以下のコードをオーケストレーション・データベースでGGORCHとして実行する必要があります。このコードでは、エイリアス***repdb2***がソースとして参照するデータベースと、エイリアス***repdb1***がターゲットとして参照するデータベース間の一方向レプリケーションを作成します。この設定により、ソースからターゲットに'scott'スキーマと'hr'スキーマがレプリケートされます。この手順と先の手順を組み合わせると、***repdb1***と***repdb2***間で'scott'スキーマと'hr'スキーマの双方向レプリケーションが作成されます。

```
set serverout on
DECLARE
  response_info clob;
  ret number;
BEGIN
  ret := add_oneway_replication(
              ogg_source                          => ogg_connection('&machine3',9900,'gg_repdb1'),
              source_dblogin_credential_alias     => 'repdb2',
              ogg_target                          => ogg_connection('&machine2',9900,'gg_repdb2'),
              target_dblogin_credential_alias     => 'repdb1',
              tables                              => '"scott.*","hr.*"',
              response_info                       => response_info);
END;
/
```

以下の出力には、Extract ***EFCUG***、Distribution path ***PFCUG***、およびReplicat ***RFCUG***が作成されたことが表示されています。ExtractとDistribution pathはSOURCE OGG DEPLOYMENTで'running'と表示され、ReplicatはTARGET OGG DEPLOYMENTで'starting'と表示されています。先の手順で作成されたレプリケーション・コンポーネントの***EFIML***、***PFIML***、***RFIML***が出力に表示される場合もあります。

出力

```
---------------- REPLICATION:REPDB2 =>REP DB1 -------------------
---------------- EXTRACT -------------------
EXTRACT (Integrated) added.
Extract EFCUG successfully registered with database at SCN 4624869.
EXTTRAIL added.
EXTRACT EFCUG starting EXTRACT EFCUG started



---------------- DISTRIBUTION PATH -------------------
The path 'PFCUG' has been added.


---------------- REPLICAT -------------------
REPLICAT (Integrated) added.
REPLICAT RFCUG starting
REPLICAT RFCUG started

---------- SOURCE OGG DEPLOYMENT ---------
Group   Status          Program
EFCUG : ["running"]   - EXTRACT
RFIML : ["running"]   - REPLICAT
PFCUG : ["running"]   - PATH

---------- TARGET OGG DEPLOYMENT ---------
Group   Status          Program
EFIML : ["running"]   - EXTRACT
RFCUG : ["starting"]  - REPLICAT
PFIML : ["running"]   - PATH
```

4.2 レプリケーション・コンポーネントの管理

レプリケーション・コンポーネントは、***add_oneway_replication***を使用して作成した後は、更新、起動、停止、または削除できます。

以下に、Extractで実行できるアクションの例をいくつかご紹介します。必要に応じて、類似のアクションを他のレプリケーション・コンポーネントで実行できます。

4.2.1 Extract情報の取得

以下のコードは、ポート***9900***をリッスンするマシン***&machine3***上のOGGデプロイメント***gg_repdb2***において、Extract ***&extract_name***の情報を取得します。

```
set serverout on
DECLARE
  response_info CLOB;
  ret           NUMBER;
BEGIN
  ret := OGGSERVICE.get_extract_details(
                    ogg_instance    => ogg_connection('&machine3',9900,'gg_repdb2'),
                    extract_name    => '&extract_name',
                    response_info   => response_info);

 pretty_json(response_info);
END;
/
```

以下の出力は、Extract ***EFCUG***のフェッチされた情報です。“status”が“running”であり、“source”属性で“integrated"のExtractであることが表示されていることに注意してください。Extractのパラメータ・ファイル全体が“config”属性に表示される場合もあります。この出力のjsonスキーマは、こちらのドキュメントで参照できます。

出力

```
[
  {
    "messages" : [
      [
      ]
    ],
   "response" :
    [
      {
        "$schema" : "ogg:extract",
        "credentials" :
        {
          "alias" : "repdb2",
          "domain" :"OracleGoldenGate"
        },
        "begin" : "now",
        "targets" :
        [
          {
            "name" : "jq",
            "sizeMB" :500,
            "sequenceLength" : 9,
            "sequenceLengthFlip" : false,
            "sequence" :0,
            "offset" : 31556912,
           "remote" : false
          }
        ],
        "config" :
        [
          "extract EFCUG;",
          "useridalias repdb2 domain OracleGoldenGate",
          "exttrail jq",
          "table scott.*;",
          "table hr.*;"
        ],
        "description" :"EFCUG",
        "source" :
        {
          "tranlogs" : "integrated"
        },
        "registration" :
        {
          "csn" :4624869,
          "share" : true
        },
        "status" : "running"
      }
    ]
  }
]
```


4.2.2 Extractパラメータ・ファイルの更新

Extractパラメータ・ファイルの行は、REST APIペイロードでJSON配列として表示されます（セクション4.2.1の出力の“config”を参照）。

この例では、既存のパラメータ・ファイルに行を追加することで、Extractのパラメータ・ファイルを更新します。まずExtractの既存パラメータを取得し、新しいパラメータを既存のリストに追加し、修正した行リストを使用してパラメータ・ファイルを更新することで、この処理を行います。

以下のコードは、ポート***9900***をリッスンするマシン***&machine3***上のOGGデプロイメント***gg_repdb2***において、Extract ***&extract_name***のパラメータ・ファイルを更新します。

```
set serverout on
DECLARE
 response_info  CLOB;
 ret            NUMBER;
 ogg_deployment OGG_CONNECTION;
 config_jarray  JON_ARRAY_T;
 config_str     VARCHAR2(4000);
BEGIN

 ogg_deployment := ogg_connection('&machine3',9900,'gg_repdb2');

 /* Extractの既存パラメータを取得 */
 ret := OGGSERVICE.get_extract_details(
          ogg_instance => ogg_deployment,
          extract_name => '&extract_name',
          response_info => response_info);

 config_jarray := JSON_ARRAY_T.PARSE(JSON_QUERY(response_info,
                                    '$.response.config'));

 /* 新しいパラメータを既存のパラメーター式に追加 */
 config_jarray.append('DDLOPTIONS ADDTRANDATA RETRYOP RETRYDELAY 10 MAXRETRIES 10');

 /* JSON配列から括弧を削除 */
 config_str := config_jarray.to_string;
 config_str := replace(replace(config_str,'[',''),']','');

 /* Extractの修正したパラメータを更新 */
 ret := OGGSERVICE.update_extract(
                ogg_instance   => ogg_deployment,
                extract_name   => '&extract_name',
                extract_params => config_str,
                response_info  => response_info);

 /* 更新が正常に終了したら、Extract情報を取り込み、更新が完了していることを確認 */
 IF ret != OGGHTTP.resp_success THEN
   pretty_json(response_info);
 ELSE
   ret := OGGSERVICE.get_extract_details(
          ogg_instance => ogg_deployment,
          extract_name => '&extract_name',
          response_info => response_info);

   Pretty_json(response_info);
 END IF;
END;
/
```

以下の出力には、Extract ***EFCUG***のパラメータ・ファイルの更新結果が表示されています。パラメータ・ファイルに *"DDLOPTIONS ADDTRANDATA RETRYOP RETRYDELAY 10 MAXRETRIES 10* "という行が追加されていることに注意してください。

（この更新の前のパラメータ・ファイルについては、セクション4.2.1の出力を参照）

出力


```
[
  {
   "messages" :
    [
      [
      ]
    ],
   "response" :
    [
      {
        "$schema" : "ogg:extract",
        "credentials" :
        {
          "alias" : "repdb2",
          "domain" :"OracleGoldenGate"
        },
        "begin" : "now",
        "targets" :
        [
          {
            "name" : "jq",
            "sizeMB" :500,
            "sequenceLength" : 9,
            "sequenceLengthFlip" : false,
            "sequence" :0,
            "offset" : 31556912,
           "remote" : false
          }
        ],
        "config" :
        [
          "extract EFCUG;",
          "useridalias repdb2 domain OracleGoldenGate",
          "exttrail jq",
          "table scott.*;",
          "table hr.*;",
          "DDLOPTIONS ADDTRANDATA RETRYOP RETRYDELAY 10 MAXRETRIES 10"
        ],
        "description" :"EFCUG",
        "source" :
        {
          "tranlogs" : "integrated"
        },
        "registration" :
        {
          "csn" :4624869,
          "share" : true
        },
        "status" : "running"
      }
    ]
  }
]
```


4.2.3 Extractの停止

以下のコードは、ポート***9900***をリッスンするマシン***&machine3***上のOGGデプロイメント***gg_repdb2***において、Extract ***&extract_name***を停止します。


```
set serverout on
DECLARE
  response_info clob;
  ret number;
BEGIN
  ret := OGGSERVICE.STOP_EXTRACT(
             ogg_instance => ogg_connection('&machine3',9900,'gg_repdb2'),
             extract_name => '&extract_name',
             response_info => response_info);

  pretty_json(response_info);

END;
/
```

以下は、Extract ***EFCUG***の出力です。Extractが停止されたことが表示されています。

```
[
  {
    "messages" :
    [
      [
        {
          "$schema" : "ogg:message",
          "title" :"EXTRACT EFCUG stopped",
          "code" :"OGG-00975",
          "severity" :"INFO",
          "issued" :"2018-08-17T12:04:22Z",
          "type" : "http://docs.oracle.com/goldengate/c1910/gg-winux/GMESG/oggus.htm#OGG-00975"
        },
        {
          "$schema" : "ogg:message",
          "title" :"EXTRACT EFCUG stopped",
          "code" :"OGG-15426",
          "severity" :"INFO",
          "issued" :"2018-08-17T12:04:22Z",
          "type" : "http://docs.oracle.com/goldengate/c1910/gg-winux/GMESG/oggus.htm#OGG-15426"
        }
      ]
    ],
    "response" : " "
  }
]
```

4.2.4 Extractの削除

以下のコードは、ポート***9900***をリッスンするマシン***&machine3***上のOGGデプロイメント***gg_repdb2***において、Extract ***&extract_name***を削除します。

```
set serverout on
DECLARE
  response_info clob;
  ret number;
BEGIN
  ret := OGGSERVICE.DELETE_EXTRACT(
             ogg_instance   => ogg_connection('&machine3',9900,'gg_repdb2'),
             extract_name   => '&extract_name',
             response_info  => response_info);
  pretty_json(response_info);
END;
/
```

以下は、Extract ***EFCUG***の出力です。これは統合Extractのため、Extractは削除される前にデータベースから登録解除されていることに注意してください。heartbeat表のExtractに対応する行もすべて削除されます。

出力


```
[
  {
   "messages" :
    [
      [
        {
          "$schema" : "ogg:message",
          "title" :"Sending STOP request to EXTRACT EFCUG ",
          "code" :"OGG-08100",
          "severity" :"INFO",
          "issued" :"2018-08-17T12:25:03Z",
          "type" : "http://docs.oracle.com/goldengate/c1910/gg-winux/GMESG/oggus.htm#OGG-08100"
  },
        {
          "$schema" : "ogg:message",
          "title" :"EXTRACT EFCUG is down (gracefully)", "code" :"OGG-00979",
          "severity" :"INFO",
          "issued" :"2018-08-17T12:25:03Z",
          "type" : "http://docs.oracle.com/goldengate/c1910/gg-winux/GMESG/oggus.htm#OGG-00979"
  },
        {
          "$schema" : "ogg:message",
          "title" :"Successfully unregistered EXTRACT EFCUG from database.", "code" :"OGG-01750",
          "severity" :"INFO",
          "issued" :"2018-08-17T12:25:14Z",
          "type" : "http://docs.oracle.com/goldengate/c1910/gg-winux/GMESG/oggus.htm#OGG-01750"
  },
        {
          "$schema" : "ogg:message",
          "title" :"No Heartbeat entries with [EFCUG], none deleted.", "code" :"OGG-14052",
          "severity" :"INFO",
          "issued" :"2018-08-17T12:25:14Z",
          "type" : "http://docs.oracle.com/goldengate/c1910/gg-winux/GMESG/oggus.htm#OGG-14052"
  },
        {
          "$schema" : "ogg:message",
          "title" :"Deleted EXTRACT EFCUG.", "code" :"OGG-08100",
          "severity" :"INFO",
          "issued" :"2018-08-17T12:25:14Z",
          "type" : "http://docs.oracle.com/goldengate/c1910/gg-winux/GMESG/oggus.htm#OGG-08100"
        }
      ]
    ],
    "response" : " "
  }
]
```


5 トラブルシューティング

サンプルのオーケストレーション・スクリプトを使用した場合に発生する可能性のあるエラーの一部と、考えられる原因を以下に示します。

エラー	考えられる原因
"file:$ORACLE_HOME/admin/ggorch_wallet is not configured for access to user: <USER>"] }	オーケストレーション・ユーザーにウォレット・ディレクトリへのアクセス権が付与されていません。ウォレット・ディレクトリへのアクセス権を付与するサンプル・コードについては、*oggorch_setup.sql*を参照してください。
"title" :"The authorization information for <service> is missing, invalid or not properly formed.",	$ORACLE_HOME/ggorch_walletの証明書に記載されるユーザー名に、OGGデプロイメントのREST API呼出しに対するアクセス権がない可能性があります。
ORA-29024:Certificate validation Failure	$ORACLE_HOME/ggorch_walletの証明書が、OGGデプロイメントによって信頼されている認証局によって署名されていない可能性があります。
ORA-24247: network access denied by access control list (ACL)	DB管理者ユーザーにHTTPS呼出しを行う権限がありません（オーケストレーション・パッケージがインストールされている場合）。HTTPSアクセスをユーザーに付与するサンプル・コードについては、*oggorch_setup.sql*を参照してください。
ORA-28759: failure to open file	$ORACLE_HOME/admin/ggorch_walletディレクトリにウォレットがない可能性があります。

6 結論

提供されるパッケージが、OGG Microservicesのデプロイメントで可能なあらゆるアクションを実行するためのファンクションをすべて提供しているわけではないことに気付かれたかもしれません。認証のさまざまなセキュリティ・モードも提供していません。ただし、このパッケージは容易に拡張できます。拡張によって、PL/SQLのみを使用して、OGGデプロイメントを保守する極めて実用的なREST APIクライアントを実現できます。

7 付録 - I

OGGデプロイメントがポート443のリバース・プロキシを使用して構成されている場合、上記のすべての例に以下の変更が必要です。

```
ogg_connection('&machine2',9900,'gg_repdb1')を
ogg_connection('&machine2','gg_repdb1')に変更する
```

例

```
set serverout on
DECLARE
  response_info clob;
  ret number;
BEGIN
  ret := OGGSERVICE.DELETE_EXTRACT(
             ogg_instance => ogg_connection('&machine2', 'gg_repdb1'),
             extract_name => '&extract_name',
             response_info => response_info);
  pretty_json(response_info);
END;
/
```

リバース・プロキシが非標準のポートで構成されている場合、以下の変更が必要です。

```
ogg_connection('&machine2',9900,'gg_repdb1')を
ogg_connection('&machine2',&reverse_proxy_port,'gg_repdb1', 1)に変更する
```

例

```
set serverout on
DECLARE
  response_info clob;
  ret number;
BEGIN
  ret := OGGSERVICE.DELETE_EXTRACT(
             ogg_instance => ogg_connection('&machine2', &reverse_proxy_port,'gg_repdb1', 1),
             extract_name => '&extract_name',
             response_info => response_info); pretty_json
  (response_info);
END;
/
```

8 付録 - II

ソース・データベースとターゲット・データベース間の一方向レプリケーションを作成するトップ・レベルのファンクションは以下のとおりです。

```
FUNCTION ADD_ONEWAY_REPLICATION (                    NECTION,
   ogg_source IN                                     2,
   db_source_alias IN                                NECTION,
   ogg_target IN                                     2,
   db target alias IN                                2,
   tables                                  CLOB)
   response_info
```

パラメータ	説明
ogg_source	ソースOGGデプロイメントの接続情報
db_source_alias	ogg_source_deploymentで作成されたソース・データベースに対するエイリアス
ogg_target	ターゲットOGGデプロイメントの接続情報
db_target_alias	ogg_target_deploymentで作成されたターゲット・データベースに対するエイリアス
tables	レプリケートされる表とスキーマの、カンマで区切られたリスト SCOTTスキーマはSCOTT.*、SCOTTスキーマのEMP表は SCOTT.EMPと指定します。 例：'SCOTT.*, USER1.TAB, USER2.*' 上記の文字列一式により、ソースからターゲットへのSCOTT スキーマのレプリケーション、 ソースからターゲットへのUSER2スキーマのレプリケーション、ソースのUSER1.TAB表からターゲットの USER1.TAB表へのレプリケーションが設定されます。
response_info	OGGサーバーからのレスポンス・メッセージ

ファンクション内には、様々な要件に合わせてファンクションをカスタマイズできる多数の構成変数があります。

名前（extract、replicat、path、trail）は、必要であり、NULLの場合は、常に生成されます。以下の変数は希望の値に初期化できます。そうすることで、値は生成されず、初期化された値が代わりに使用されます。

```
extract_name            VARCHAR2(5)
extract_mode            VARCHAR2(20)
path_name               VARCHAR2(5)
replicat_name           VARCHAR2(5)
replicat_mode           VARCHAR2(20)
source_trail            VARCHAR2(2)
source_trail_path       VARCHAR2(2)
target_trail            VARCHAR2(2)
target_trail_path       VARCHAR2(2)
extract_params_template VARCHAR2(4000)                  :=
                '"extract <extract_name>;",'                          ||
                '"useridalias <db alias> domain OracleGoldenGate",' ||
                  '"exttrail <extract_trail>",'                        ||
                  '<tables_stmt>' ;
replicat_params_template VARCHAR2(4000)                   :=
                  '"replicat <replicat_name>;",'                       ||
                '"useridalias <db_alias> domain OracleGoldenGate",' || '
                <map_stmt>';
```

9 付録 - III

ターゲット・データベースのレプリケーション・オブジェクトは、***ogginstantiate.sql***ファイルで提供されるOGGINSTANTIATEパッケージを使用して、ソース・データベースのレプリケーション・オブジェクトからインスタンス化できます。

OGGINSTANTIATEパッケージを実行するユーザーには、以下の権限付与が必要です。

- grant execute on dbms_datapump to <ユーザー>
- grant create table to <ユーザー>

レプリケーションを設定する前に、ターゲット・レプリケーション・データベースからOGGINSTANTIATEパッケージを実行する必要があります。データベース・リンクを使用して、オブジェクトをソース・データベースからインポートすると、インスタンス化が完了します。OGGINSTANTIATEパッケージは、DBMS_DATAPUMP APIの例で提供されるコードを使用して構築されています。

OGGINSTANTIATEによって提供される唯一のプロシージャの署名は以下のとおりです。

```
/*
   PROCEDURE instantiate
     ソース入力のスキーマ/表を使用して、ターゲット・スキーマ/表をインスタンス化します。
     source_obj    - インスタンス化で使用されるソース・オブジェクトは、
                     全データベースのインスタンス化では'*.*'の形式にします。
                      （スキーマのインスタンス化では'SCOTT.*'、表のインスタンス化では'SCOTT.EMP'）
     target_obj    - インスタンス化で使用されるターゲット・オブジェクトは、
                     全データベースのインスタンス化では'*.*'の形式にします。
                      （スキーマのインスタンス化では'SCOTT.*'、表のインスタンス化では'SCOTT.EMP'）
     source_dblink - ソース・データベースに対するデータベース・リンク

*/
--------------------------------------------------------------------------------

  PROCEDURE instantiate(source_obj     IN VARCHAR2,
                        target_obj     IN VARCHAR2,
                        source_dblink  IN VARCHAR2);
--------------------------------------------------------------------------------
```

例 – 1

全データベースのインポートによって、ソース・データベースからターゲット・データベースをインスタンス化します。***src_dblink***は、ソース・データベースに対するデータベース・リンクです

```
sqlplus> set serverout on
sqlplus> execute OGGINSTANTIATE.INSTANTIATE('*.*', '*.*', src_dblink)
```

例 – 2

ソース・データベースのscottスキーマから、ターゲット・データベースのscottスキーマのみをインスタンス化します。***src_dblink***は、ソース・データベースに対するデータベース・リンクです

```
sqlplus> set serverout on
sqlplus> execute OGGINSTANTIATE.INSTANTIATE('scott.*', 'scott.*', src_dblink)
```

注：target_objが'scott.'ではなく'hr.*'と指定されている場合、ソース・データベースの'scott'スキーマがターゲット・データベースの'hr'スキーマにインポートされます。*

例 – 3

ソース・データベースのscott.emp表を使用して、ターゲット・データベースのscott.emp表のみをインスタンス化します。***src_dblink***は、ソース・データベースに対するデータベース・リンクです

```
sqlplus> set serverout on
sqlplus> execute OGGINSTANTIATE.INSTANTIATE('scott.emp', 'scott.emp', src_dblink)
```

注：target_objが'scott.emp'ではなく'hr.emp'と指定されている場合、ソース・データベースの'scott.emp'表がターゲット・データベースの'hr.emp'表にインポートされます。

上記のすべての例において、ソース・データベースからインポートしているオブジェクトがターゲット・データベースに存在する場合、オブジェクトはインポートされません。

10 参考資料

[1] Oracle GoldenGate Microservices Architecture

[2] 自己署名証明書の作成

[3] セキュアなデプロイメントまたはセキュアでないデプロイメントの設定

[4] Oracle GoldenGate Microservices Architectureのコンポーネント

[5] OGG Microservices REST API Reference guide

[6] Oracle GoldenGate環境の保護

[7] UTL_HTTP

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

PL/SQLを使用した
Oracle GoldenGate
Microservicesの
オーケストレーション・
ユーティリティ

2018年10月
著者：Parthasarathy Raghunathan