



EDB

Migration Toolkit

リリース **55.2.0**

EnterpriseDB

2022 年 01 月 04 日

目次

第 1 章	Supported operating systems and database versions	3
1.1	データベースのバージョン	3
1.2	サポートされているオペレーティングシステムのバージョン	3
第 2 章	Migration methodology	5
2.1	移行プロセス	5
2.2	アプリケーションを Postgres に接続する	6
第 3 章	Functionality overview	9
3.1	オブジェクト移行のサポート	10
3.2	オンライン移行とオフライン移行	11
第 4 章	Installing Migration Toolkit	13
4.1	CentOS または RHEL 7 および 8 ホストへの Migration Toolkit のインストール	13
4.1.1	RPM インストールのアップグレード	16
4.2	CentOS または RHEL 7 ppc64le ホストへの Migration Toolkit のインストール	16
4.3	Debian または Ubuntu ホストへの Migration Toolkit のインストール	18
4.4	SLES ホストへの Migration Toolkit のインストール	19
4.5	Windows に Migration Toolkit をインストールする	20
4.6	Mac OS X に Migration Toolkit をインストールする	30
4.7	ソース固有のドライバーのインストール	35
第 5 章	Building the toolkit.properties file	37
5.1	Advanced Server URL の定義	38
5.2	PostgreSQL URL の定義	39
5.3	BigAnimal URL の定義	40
5.4	オラクル URL の定義	41
5.5	MySQL URL の定義	41
5.6	Sybase URL の定義	43
5.7	SQL Server URL の定義	43
第 6 章	Invoking Migration Toolkit	45
6.1	埋め込みたバイナリゼロ (NULL 文字) を含む文字データのインポート	45
6.2	オラクルからスキーマを移行する	46
6.3	Oracle 以外のソースデータベースからの移行	46
第 7 章	Migration Toolkit command options	49
7.1	オフライン移行オプション	50
7.1.1	オフライン移行スクリプトの実行	51
7.2	インポートオプション	52
7.3	スキーマ作成オプション	52

7.4	スキーマオブジェクト選択オプション	53
7.5	移行オプション	55
7.6	並列データローディングの移行オプション	58
7.6.1	並列データ読み込みアルゴリズムの概要	59
7.6.2	システムリソースの推奨事項とコンストレイン**	61
7.7	Oracle 固有のオプション	61
7.8	その他のオプション	65
7.9	例	66
第 8 章	Migration errors	71
8.1	接続エラー	71
8.1.1	無効なユーザー名/パスワード	71
8.1.2	接続が拒否されました：致命的：パスワード	72
8.1.3	例外：ORA-28000：アカウントはロックされています	72
8.1.4	例外：oracle.jdbc。ドライバ。OracleDriver	72
8.1.5	I/O 例外：ネットワークアダプターは接続を確立できませんでした	73
8.1.6	例外：「ターゲット」データベースに指定された URL は無効です	73
8.2	移行エラー	73
8.2.1	エラー：最後の予想列の後の追加データ	73
8.2.2	テーブルへのデータローディングエラー：* TABLE_NAME *	74
8.2.3	制約 CONS_NAME_FK の作成エラー	74
8.2.4	テーブルへのデータ読み込みエラー	75
第 9 章	Error codes	79
9.1	エラーコードの概要	79
9.1.1	クラス 02-警告	80
9.1.2	クラス 10-無効なユーザ入力	80
9.1.3	クラス 11-構成の問題	80
9.1.4	クラス 12-不十分な特権	81
9.1.5	クラス 13-サポートされていない機能	81
9.1.6	クラス 14-欠落オブジェクト	83
9.1.7	クラス 15-スキーマの移行	83
9.1.8	クラス 16-手続き言語の移行	84
9.1.9	クラス 17-データのローディング	85
第 10 章	Unsupported features	87
10.1	サポートされていない Postgres 機能	87
10.2	よくある質問	88
第 11 章	Release Notes	91
11.1	Version 55.2.0	91
11.2	Version 55.1.0	92
11.3	Version 55.0.0	92

Migration Toolkit は、移行プロセスをきめ細かく制御できる強力なコマンドラインツール。Migration Toolkit は、データベースオブジェクトとデータを EDB Postgres Advanced Server または PostgreSQL データベースに以下から移行するのに役立ちます。

- オラクル
- MySQL
- Microsoft SQL Server

Migration Toolkit を使用して、データベースオブジェクトとデータを Sybase Adaptive Server Enterprise から EDB Postgres Advanced Server に、または EDB Postgres Advanced Server と PostgreSQL 間で移行することもできます。

Migration Toolkit は、 EDB Postgres Advanced Server インストーラまたは Stack Builder を使用してインストールできます。Stack Builder は、 EDB Postgres Advanced Server と [EnterpriseDB website](#) から入手できる PostgreSQL ワンクリックインストーラの両方で配布されます。

!!! Note * この文書の Postgres という用語は、 EDB Postgres Advanced Server または PostgreSQL のインストールを指します。Stack Builder という用語は、StackBuilder Plus（ EDB Postgres Advanced Server とともに配布）または Stack Builder（EnterpriseDB からの PostgreSQL ワンクリックインストーラと共に配布）のいずれかを指します。

第1章 Supported operating systems and database versions

1.1 データベースのバージョン

Migration Toolkit では、次のデータベース製品バージョンを使用できます。

PostgreSQL バージョン 10、11、12、13、および 14 EDB Postgres Advanced Server バージョン 10、11、12、13、および 14 オラクル 10g リリース 2 オラクル 11g リリース 2 オラクル 12c リリース 1 オラクル 18c リリース 1 オラクル 19c リリース 1 SQL Server 2008 SQL Server 2012 SQL Server 2014 - MySQL 5.5 - MySQL 5.7 - MySQL 8.0 - Sybase Adaptive Server Enterprise 15.7

他のデータベース製品のサポートが必要な場合は、EnterpriseDB アカウントマネージャーまたは sales@enterprisedb.com にお問い合わせください。

!!! Note * Migration Toolkit は、オラクル Real Application Clusters (RAC) および Exadata での使用はテストされておらず、公式に使用をサポートしていません。ただし、Migration Toolkit は、単一の永続ノードに接続されている場合に機能する場合があります。詳細については、EDB 担当者にお問い合わせください。

1.2 サポートされているオペレーティングシステムのバージョン

- Centos または RHEL または OL 7 および 8
- CentOS または RHEL PPCLE 7
- SLES 12 SP5
- Debian 9 および 10
- Ubuntu 18.04 および 20 Windows (64 ビット)、2016、および 2019
- Mac OS X 10.12 以降

!!! Note * ojdbc7.jar により、ResultSet からのフェッチ中に Integer.MAX_VALUE (2147483647) を超えるレコードの制限により、ラージデータの移行が不完全になる可能性があります。この問題を回避するには、ojdbc8.jar にアップグレード処理することをお勧めします。

第2章 Migration methodology

多くの理由で、あるデータベースから別のデータベースへの移行を検討する場合があります。移行により、新しいまたはより優れたテクノロジーを活用できます。現在のデータベースがシステムのスケールリングを可能にする適切な機能セットを提供していない場合、必要な機能を提供するデータベースに移行することは、会社にとって最適な動きです。

移行はコスト効果も高くなります。かなりのメンテナンスコストでシステムを移行すると、システムの維持に費やすお金を節約できます。使用中のデータベースの数を統合することにより、社内の管理コストも削減できます。使用するデータベースプラットフォームの数を減らす（またはデータベースの互換性を活用する）ことにより、IT 予算をより多く活用できます。

複数のデータベースプラットフォームを使用すると、ベンダが価格を引き上げたり、会社のディレクティブを変更した場合に、適切な移行パスを提供できます。EnterpriseDB は長年にわたり、企業が既存のデータベースシステムを Postgres に移行するのを支援してきました。

</ div>

2.1 移行プロセス

推奨される Postgres への移行パスには、次の主な手順が含まれます。

1. 移行に含めるデータベースオブジェクトとデータを決定して、移行プロセスを開始します。ソースシステムのアーキテクチャと実装の確かな知識を持った人を含む移行チームを形成します。
2. 潜在的な移行の問題を特定します。Oracle から EDB Postgres Advanced Server への移行の場合、EDB EDB Postgres Advanced Server でサポートされている互換性機能の詳細については、[EnterpriseDB Postgres Advanced Server documentation](#) を参照してください。このレビューにヘルプ EnterpriseDB 移行評価サービスの使用を検討してください。
3. 移行環境を準備します。必要なソフトウェアを入手してインストールし、サーバー間の接続を確立します。
4. 移行にラージのデータが含まれる場合は、データを移動する前にスキーマ定義を移行することを確認してください。DDL 移行の結果を確認し、移行の概要で報告された問題を解決します。[Migration errors](#) には、移行の問題の解決に関する情報が含まれています。
5. データを移行します。小さいデータセットの場合は、Migration Toolkit を使用します。オラクルが EDB Postgres Advanced Server に移行し、データセットがラージ場合、またはデータ転送が遅いことに気付いた場合は、他の利用可能なデータ移動方法のいずれかを利用してください。
 - -Oracle データベースと互換性のある EDB Postgres Advanced Server データベースリンク機能をオラクルします。-データに BLOB または CLOB データがある場合、オラクルスタイルのデータベースリンクの代わりに dblink_ora スタイルのデータベースリンクを使用します。

- これらのメソッドはどちらもオラクル Call Interface (OCI) を使用してオラクルに接続します。接続後、SQL ステートメントを使用して「リンクされた」オラクルデータベースからデータを選択し、EDB Postgres Advanced Server データベースにデータを挿入します。

6. データ移行の結果を確認し、移行の概要で報告された問題を解決します。

7. 新しく移行された Postgres データベースで動作するようにアプリケーションを変換します。JDBC や ODBC などのオープンスタンダード接続を使用するアプリケーションでは、通常、データベースコネクション文字列と EnterpriseDB ドライバの選択のみを変更する必要があります。詳細については、*Connecting an application to Postgres* を参照してください。

8. システムのパフォーマンスをテストし、新しいサーバーを調整します。EDB Postgres Advanced Server データベースに移行する場合は、EDB Postgres Advanced Server のパフォーマンスチューニングユーティリティを利用してください。

- -Dynatune を使用して、データベース構成リソースを動的に調整します。-オプティマイザーヒントを使用して、クエリーを指定しパス。-ANALYZE コマンドを使用して、データベース統計を取得します。
- EDB Postgres Advanced Server で利用可能なパフォーマンスチューニングツールに関する情報および参照。

</ div>

2.2 アプリケーションを Postgres に接続する

クライアントアプリケーションを変換して Postgres データベースを使用するには、接続プロパティを変更して新しいターゲットデータベースを指定する必要があります。Java アプリケーションの場合、JDBC ドライバ名前 (Class.forName) と JDBC URL を変更します。

オラクルで実行されている Java アプリケーションには、次の接続プロパティがあります。

```
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con =
DriverManager.getConnection
("jdbc:oracle:thin:@localhost:1521:xe",
*
"user",
*
"password")
```

接続文字列を変更して、Postgres サーバーに接続します。

```
Class.forName("com.edb.Driver")
Connection con = DriverManager.getConnection
("jdbc:edb://localhost:5444/edb",
*
"user",
*
"password");
```

ODBC アプリケーションを変換して Postgres のインスタンスに接続するのは 2 段階のプロセスです。

1. ODBC アプリケーションを接続するには、ODBC データソース管理者を使用して、新しいターゲットデータベースの接続プロパティを定義するデータソースを作成します。

- ほとんどの Linux および Windows システムには、ODBC データソースを作成および編集できるグラフィカルツールが含まれています。ODBC をインストールした後、**管理ツール**メニューで ODBC データソースアドミニストレーターへのリンクを確認します。**[追加]** を選択して、新しいデータソースの作成ウィザードをスタートします。次に、ダイアログを完了して、新しいターゲットデータソースを定義します。

2. 新しいデータソースを使用するようにアプリケーションを変更します。

- アプリケーションには、`SQLConnect` または `SQLDriverConnect` への呼び出しが含まれています。呼び出しを編集して、データソース名前を変更します。次の例では、データソースの名前付けは `OracleDSN` です。

```
* result = SQLConnect(conHandle,          // Connection handle
*                                     (returned)
*      "OracleDSN", SQL_NTS,            // Data source name
*      username, SQL_NTS,               // User name
*      password, SQL_NTS);              // Password
```

`PostgresDSN` 名前付けのデータソースで定義された Postgres のインスタンスに接続するには、データソース名前を変更します。

```
result = SQLConnect(conHandle,          // Connection handle (returned)
*      "PostgresDSN", SQL_NTS,          // Data source name
*      username, SQL_NTS,               // User name
*      password, SQL_NTS);              // Password
```

アプリケーションとサーバー間の接続を確立した後、アプリケーションをテストして、アプリケーションと移行されたスキーマ間の互換性の問題を見つけます。ほとんどの場合、シンプル変更により、アプリケーションで発生する非互換性が解決されます。機能がサポートされていない場合は、回避策またはサードパーティのツールを使用して、アプリケーションに必要な機能を提供してください。一般的な問題とその回避策については、[Migration errors](#) を参照してください。

第3章 Functionality overview

Migration Toolkit は、移行プロセスをきめ細かく制御できる強力なコマンドラインツール。Migration Toolkit の使用は 2 段階のプロセスです。

1. `toolkit.properties` ファイルを編集して、ソースデータベースとターゲットデータベースを指定します。
2. 移行オプションを指定して、コマンドラインで Migration Toolkit を起動します。

Migration Toolkit は、データベースオブジェクトとデータを以下から EDB Postgres Advanced Server または PostgreSQL データベースに移行するのに役立ちます。

- オラクル
- MySQL SQL Server

Migration Toolkit を使用すると、データベースオブジェクトとデータを Sybase から EDB Postgres Advanced Server データベースに移行することもできます。Migration Toolkit を使用して、EDB Postgres Advanced Server と PostgreSQL の間で移行することもできます。Migration Toolkit には多数のオプションが含まれており、移行プロセスをきめ細かく制御できます。

- `-safeMode` オプションを使用して、移行される行をコミットします。
- `-fastCopy` オプションを使用して、WAL ロギングをバイパスし、移行を最適化します。
- `-batchSize` オプションを使用して、一括挿入のバッチサイズを制御します。
- `-cpBatchSize` オプションを使用して、COPY コマンドで使用するバッチサイズを指定します。
- `-lobBatchSize` オプションを使用して、ラージオブジェクトデータ型に使用されるバッチサイズを指定します。
- `-filterProp` オプションを使用して、指定した条件を満たす行のみを移行します。
- `-customColTypeMapping` オプションを使用して、選択した列のデータタイプを変更します。
- `-dropSchema` オプションを使用して、移行前に既存のスキーマを削除し、新しいスキーマを作成します。
- EDB EDB Postgres Advanced Server では、`-allDBLinks` オプションを使用して、すべてのオラクルデータベースリンクを移行します。
- EDB EDB Postgres Advanced Server では、`-copyViaDBLinkOra` オプションを使用して `dblink_ora` モジュールを有効にします。

3.1 オブジェクト移行のサポート

Migration Toolkit は、オブジェクト定義（DDL）、テーブルデータ、またはその両方を移行します。次の表には、Migration Toolkit が移行できるデータベースオブジェクトのタイプのプラットフォーム固有のリストが含まれています。

Object	Oracle	Sybase	SQL Server	MySQL
Schemas	X	X	X	X
Tables	X	X	X	X
List-partitioned table	X			
Range-partitioned table	X			
Hash partitioned table	X			
Constraints	X	X	X	X
Indexes	X	X	X	X
Triggers	X			
Table data	X	X	X	X
Views	X		X	
Materialized views	X			
Packages	X			
Procedures	X			
Functions	X			
Sequences	X			
Users/Roles	X			
Profiles	X			
Object types	X			
Object type methods	X			
Database links	X			
Queues	X			

次のページに続く

表 1 – 前のページからの続き

Object	Oracle	Sybase	SQL Server	MySQL

インポートされたオブジェクトの詳細な制御を提供するコマンドの詳細については、[Schema object selection options](#) を参照してください。

3.2 オンライン移行とオフライン移行

Migration Toolkit は、すぐに直接 Postgres データベースに移行できます（**オンライン移行**）。また、後で Postgres データベースでオブジェクト定義を再作成するために使用するスクリプトを生成することもできます（**オフライン移行**）。

デフォルトでは、Migration Toolkit は Postgres データベースにオブジェクトを直接作成します。または、`-offlineMigration` オプションを含めて、後で使用して新しいデータベースに移行したオブジェクトまたはデータを再現できる SQL スクリプトを生成します。Migration Toolkit で生成された移行スクリプトをカスタマイズしてから実行ことで、移行したオブジェクトを変更できます。`-offlineMigration` オプションを使用すると、システムのロードに最適な時間に実際の移行をスケジュールできます。

`-offlineMigration` オプションの詳細については、[Offline migration options](#) を参照してください。

第4章 Installing Migration Toolkit

Migration Toolkit をインストールする前に、Java（バージョン 1.8.0 以降）をインストールする必要があります。

Migration Toolkit をインストールするには、EnterpriseDB リポジトリへのアクセスを許可する資格情報が必要です。リポジトリの資格証明をリクエストするには、次の EnterpriseDB ウェブサイトにアクセスしてください。

<https://www.enterprisedb.com/repository-access-request>

以下は、さまざまなインストール方法です。

- *Installing Migration Toolkit on a CentOS or RHEL Host*
- *Installing Migration Toolkit on a CentOS, or RHEL 7 ppc64le Host*
- *Installing Migration Toolkit on a Debian or Ubuntu Host*
- *Installing Migration Toolkit on a SLES Host*
- *Installing Migration Toolkit on Windows*
- *Installing Migration Toolkit on Mac OS X*

</ div>

4.1 CentOS または RHEL 7 および 8 ホストへの Migration Toolkit のインストール

RPM パッケージを使用して、CentOS または RHEL ホストに Migration Toolkit をインストールできます。次の手順では、EnterpriseDB リポジトリへのアクセスと Migration Toolkit のインストールに関する詳細情報を提供します。

リポジトリ設定ファイルの作成

1. リポジトリ構成ファイルを作成するには、スーパーユーザ特権を引き受けて、次のコマンドを呼び出します。

- RHEL または CentOS 7 の場合：

```
* * yum -y install https://yum.enterprisedb.com/edb-repo-rpms/
* * edb-repo-latest.noarch.rpm
```

- RHEL または CentOS 8 の場合：

```
* dnf -y install https://yum.enterprisedb.com/edb-repo-rpms/  
* edb-repo-latest.noarch.rpm
```

リポジトリ構成ファイルの名前付けは `edb.repo` です。ファイルは `/etc/yum.repos.d` にあります。

2. `edb.repo` ファイルを作成した後、エディタの選択を使用して `enabled` パラメータの値が 1 であることを保証し、`baseurl` 仕様の `username` および `password` プレースホルダーを登録済みの EnterpriseDB ユーザの名前とパスワードに置き換えます。

```
* [edb]  
* name=EnterpriseDB RPMs $releasever - $basearch  
* baseurl=https://<username>:<password>@yum.enterprisedb.com/  
* edb/  
* redhat/rhel-$releasever-$basearch  
* enabled=1  
* gpgcheck=1  
* gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

3. 変更を構成ファイルに保存した後、EPEL リポジトリをインストールします。

- RHEL または CentOS 7 の場合：

```
* * yum -y install https://dl.fedoraproject.org/pub/epel/  
* * epel-release-latest-7.noarch.rpm
```

- RHEL または CentOS 8 の場合：

```
* dnf -y install https://dl.fedoraproject.org/pub/epel/  
* epel-release-latest-8.noarch.rpm
```

4. RHEL または CentOS 8.x を使用している場合、依存関係を解決するには追加のリポジトリを有効にする必要があります。

- CentOS 8 (Powertools リポジトリ)：

```
* dnf config-manager --set-enabled PowerTools
```

- RHEL 8 (codeready-builder-for-rhel-8- * -rpms repo)：

```
* ARCH=$( /bin/arch )  
* subscription-manager repos --enable "codeready-builder-for  
* -rhel-8-${ARCH}-rpms"
```

**** Migration Toolkit のインストール ****

- RHEL または CentOS 7 の場合：

```
* * yum -y install edb-migrationtoolkit
```

- RHEL または CentOS 8 の場合：

```
* dnf -y install edb-migrationtoolkit
```

システムで認識されないソースによって署名された RPM パッケージをインストールすると、yum はローカルサーバにキーをインポートするパーミッションを求める場合があります。プロンプトが表示され、パッケージソフトが信頼できるソースからのものであることに満足したら、y を入力し、Return を押して続行します。

インストール中に、yum は解決できない依存関係に遭遇する場合があります。存在する場合、手動で解決する必要がある依存関係のリストが提供されます。

Migration Toolkit をインストールした後、インストールを構成する必要があります。Migration Toolkit を呼び出す前に、次の手順を実行します。

**** IDENT 認証で Migration Toolkit を使用****

デフォルトでは、RPM インストーラの `pg_hba.conf` ファイルは、リモートクライアントに対して IDENT 認証を強制します。Migration Toolkit を起動する前に、`pg_hba.conf` ファイルを変更し、認証メソッドを IDENT 以外のフォームに変更（およびサーバーを再起動）するか、次の手順を実行して保証サーバーにアクセスできるようにする必要があります。

1. `identd` サーバーがインストールされ、実行されていることを確認します。次のコマンドを呼び出して、yum パッケージマネージャを使用して `identd` サーバーをインストールできます。

```
• `yum install xinetd authd`
```

- コマンドは、次を含む `/etc/xinetd.d/authd` という名前付けのファイルを作成する必要があります。

```
* service auth
* {
*   disable      = yes
*   socket_type  = stream
*   wait         = no
*   user         = ident
*   cps          = 4096 10
*   instances    = UNLIMITED
*   server       = /usr/sbin/in.authd
*   server_args  = -t60 --xerror --os
* }
```

- **!!! Note**
- ファイルの `server_args` の最後に `-E` 引数が含まれている場合は、`-E` を消去してください。
- 次に、`identd` サーバーをスタートするには、コマンドを呼び出します。

```
* systemctl enable xinetd
* systemctl start xinetd
```

2. `pg_ident.conf` ファイルを開き、ユーザマッピングを作成します。

```
* # map_name system_username postgres_username
* edbas enterprisedb enterprisedb
```

- どこで：
- `-map_name` 列で指定される名前は、`pg_hba.conf` ファイル内のマッピングを識別するユーザー定義の名前です。`-system_username` 列に指定されている名前は `enterprisedb` です。`-postgres_username` 列に指定されている名前は `enterprisedb` です。

3. `pg_hba.conf` ファイルを開き、`IDENT` エントリを変更します。

- IPv4 ホストの場合：
- `host all all 127.0.0.0/0 ident map=edbas`
- IPv6 ホストの場合：
- `host all all ::1/128 ident map=edbas`

4. Migration Toolkit を呼び出す前に、EDB Postgres Advanced Server サービスを再起動します。

4.1.1 RPM インストールのアップグレード

既存の RPM インストールがある場合は、`yum` を使用してリポジトリ構成ファイルをアップグレードし、より新しい製品バージョンに更新できます。`edb.repo` ファイルを更新するには、スーパーユーザ権限を引き受けて次のように入力します。

```
yum upgrade edb-repo
```

`yum` は `edb.repo` ファイルを更新して、`edb.repo` ファイルで指定された資格情報で接続するように構成された現在の EnterpriseDB リポジトリにアクセスできるようにします。次に、`yum` を使用して移行ツールキットをアップグレードできます。

```
yum upgrade edb-migrationtoolkit
```

</ div>

4.2 CentOS または RHEL 7 ppc64le ホストへの Migration Toolkit のインストール

RPM パッケージを使用して、CentOS または RHEL ppc64le ホストに Migration Toolkit をインストールできます。

移行ツールキットをインストールする前に、以下を行う必要があります。

1. Advance Toolchain リポジトリ構成ファイルを作成します。スーパーユーザ特権を想定して、次のコマンドを呼び出します。

```
* rpm --import https://public.dhe.ibm.com/software/server/POWER/
* Linux/toolchain/at/redhat/RHEL7/gpg-pubkey-6976a827-5164221b
```

2. `advance-toolchain.repo` ファイルを作成した後、エディタの選択を使用して `enabled` パラメータの値が 1 であることを保証し、`baseurl` 仕様の `username` および `password` プレースホルダーを登録済みの EnterpriseDB ユーザの名前とパスワードに置き換えます。

```
* [advance-toolchain]
* name=Advance Toolchain IBM FTP
* baseurl=https://public.dhe.ibm.com/software/server/POWER/
* Linux/toolchain/at/redhat/RHEL7
* failovermethod=priority
* enabled=1
* gpgcheck=1
* gpgkey=ftp://public.dhe.ibm.com/software/server/POWER/Linux/
* toolchain/at/redhat/RHELX/gpg-pubkey-6976a827-5164221b
```

次の手順では、EnterpriseDB リポジトリへのアクセスと Migration Toolkit のインストールに関する詳細情報を提供します。

リポジトリ設定ファイルの作成

1. EDB リポジトリ構成ファイルを作成するには、スーパーユーザ特権を引き受けて、次のコマンドを呼び出します。

```
* * yum -y install https://yum.enterprisedb.com/edb-repo-rpms/
* * edb-repo-latest.noarch.rpm
```

リポジトリ構成ファイルの名前付けは `edb.repo` です。ファイルは `/etc/yum.repos.d` にあります。

2. `edb.repo` ファイルを作成した後、エディタの選択を使用して `enabled` パラメータの値が 1 であることを保証し、`baseurl` 仕様の `username` および `password` プレースホルダーを登録済みの EnterpriseDB ユーザの名前とパスワードに置き換えます。

```
* [edb]
* name=EnterpriseDB RPMs $releasever - $basearch
* baseurl=https://<username>:<password>@yum.enterprisedb.com/edb/
* redhat/rhel-$releasever-$basearch
* enabled=1
* gpgcheck=1
* gpgkey=file:///etc/pki/rpm-gpg/ENTERPRISEDB-GPG-KEY
```

3. 変更を構成ファイルに保存した後、EPEL パッケージをインストールします。

```
* * yum -y install https://dl.fedoraproject.org/pub/epel/
* * epel-release-latest-7.noarch.rpm
```

4. RHEL 7 ppc64le を使用している場合、追加のリポジトリを有効にして依存関係を解決する必要があります。

```
* subscription-manager repos --enable "rhel-*-optional-rpms"
* --enable "rhel-*-extras-rpms" --enable "rhel-ha-for-rhel-*
* -server-rpms"
```

5. Migration Toolkit をインストールするには、次のコマンドを実行します。

```
* * yum -y install edb-migrationtoolkit
```

</div>

4.3 Debian または Ubuntu ホストへの Migration Toolkit のインストール

apt でパッケージをインストールするには、次の手順を実行します。コマンドを呼び出すときに、`username` と `password` を EnterpriseDB が提供するユーザ資格情報に置き換えます。

1. スーパーユーザ特権を想定します。

```
* sudo su -
```

2. EnterpriseDB リポジトリを構成します。

- -Debian 9 では、次の手順を実行して EnterpriseDB リポジトリを構成します。

```
* sh -c 'echo "deb https://username:password@apt.
* enterprisedb.com/$(lsb_release -cs)-edb/ $(lsb_release
* -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release
* -cs).list'
```

- -Debian 10 では、次の手順を実行して EnterpriseDB リポジトリを構成します。

- a。 EnterpriseDB リポジトリをセットアップします。

```
* sh -c 'echo "deb [arch=amd64] https://
* apt.enterprisedb.com/
* $(lsb_release -cs)-edb/ $(lsb_release -cs) main"
* > /etc/
* apt/sources.list.d/edb-$(lsb_release -cs).list'
```

- b。 次のコマンドで、``username`` および ``password`` プレースホルダーを EnterpriseDB 資格情報に置き換えます。

```
* sh -c 'echo "machine apt.enterprisedb.com login <username>
* password <password>" > /etc/apt/auth.conf.d/edb.conf'
```

- c。安全な APT リポジトリのサポートをシステムに追加します。

```
* apt-get install apt-transport-https
```

- d。 EBD 署名キーを追加します。

```
* wget -q -O - https://username:password
* @apt.enterprisedb.com/edb-deb.gpg.key | apt-key add -
```

- e。 リポジトリのメタデータを更新します。

```
* apt-get update
```

- f。 `edb-migrationtoolkit` パッケージをインストールします。

```
* apt-get -y install edb-migrationtoolkit
```

</div>

4.4 SLES ホストへの Migration Toolkit のインストール

zypper パッケージマネージャを使用して、移行ツールキットを SLES 12 ホストにインストールできます。zypper は、パッケージをインストールするときにパッケージの依存関係を満たそうとしますが、EnterpriseDB でホストされていない特定のリポジトリへのアクセスが必要です。

1. Migration Toolkit をインストールする前に、スーパーユーザ権限を引き受け、ファイアウォールを停止する必要があります。次に、以下のコマンドを使用して、EnterpriseDB リポジトリをシステムに追加します。

```
* zypper addrepo https://zypp.enterprisedb.com/suse/
* edb-sles.repo
```

2. このコマンドは、`/etc/zypp/repos.d` ディレクトリにリポジトリ構成ファイルを作成します。次のコマンドを使用して、SLES ホストのメタデータをリフレッシュし、EnterpriseDB リポジトリを含めます。

```
* * zypper refresh
```

- プロンプトが表示されたら、リポジトリの資格情報を提供します。a を指定して、提供されたキーを常に信頼し、メタデータを更新して EnterpriseDB リポジトリを含めます。
3. また、SUSE Connect と SUSE Package Hub 拡張を SLES ホストに追加し、ホストを SUSE に登録して、SUSE リポジトリへのアクセスを許可する必要があります。次のコマンドを使用します。

```
* SUSEConnect -r 'REGISTRATION_CODE' -e 'EMAIL'
* SUSEConnect -p PackageHub/12.4/x86_64
* SUSEConnect -p sle-sdk/12.4/x86_64
```

4. SUSEConnect をインストールしてホストを SUSE に登録し、SUSE リポジトリへのアクセスを許可します。

```
* zypper addrepo https://download.opensuse.org/repositories/
* Apache:/Modules/SLE_12_SP4/Apache:Modules.repo
```

5. 次のコマンドを使用して、メタデータをリフレッシュします。

```
* * zypper refresh
```

6. Java ベースのコンポーネント用の OpenJDK (バージョン 1.8) をインストールします。

```
* zypper -n install java-1_8_0-openjdk
```

7. これで、zypper ユーティリティを使用して Migration Toolkit をインストールできます。

```
* zypper -n install edb-migrationtoolkit
```

- SUSE ホストの登録に関する詳細については、次を参照してください。
- <https://www.suse.com/support/kb/doc/?id=7016626>

</div>

4.5 Windows に Migration Toolkit をインストールする

グラフィカルインストーラを使用して、Windows に Migration Toolkit をインストールできます。インストーラは www.enterprisedb.com からダウンロードできます。

インストーラをダウンロードしたら、インストーラアイコンを右クリックし、コンテキストメニューから **Run As Administrator** を選択します。Language Selection ポップアップが開いたら、インストール言語を選択し、OK をクリックして Setup ウィンドウに進みます。

!!! Note * このガイドでは、Stack Builder という用語を使用して、StackBuilder Plus (EDB Postgres Advanced Server で配布) または Stack Builder (EnterpriseDB からの PostgreSQL ワンクリックインストーラで配布) を指します。

Stack Builder で Migration Toolkit のインストールを実行する前に、Java JVM (バージョン 1.8.0 以降) が必要です。

Java 実行可能ファイルは検索パスに含まれている必要があります (%PATH% on Windows) 次のコマンドを使用して、検索パスを設定します (javadir の Java 実行可能ファイルを保持するディレクトリの名前を置き換えます)。

Windows では、次のコマンドを使用します。

```
SET PATH=javadir;%PATH%
```

Linux では、次のコマンドを使用します。

```
PATH=javadir:$PATH
```

検索パスを設定したら、Stack Builder インストールウィザードを使用して Migration Toolkit を EDB Postgres Advanced Server または PostgreSQL にインストールできます。

**** EDB Postgres Advanced Server から StackBuilder Plus を起動します****

既存の EDB Postgres Advanced Server インストールから StackBuilder Plus を起動するには、**Start**（または **Applications**）メニューから EDB Postgres メニューに移動します。EDB Add-ons メニューを開き、StackBuilder Plus メニューオプションを選択します。

**** PostgreSQL から Stack Builder を起動します****

PostgreSQL インストールから Stack Builder を起動するには、**Start**（または **Applications**）メニューから PostgreSQL メニューに移動し、Application StackBuilder Plus メニューオプションを選択します。

以下に示すように、Stack Builder が開きます。

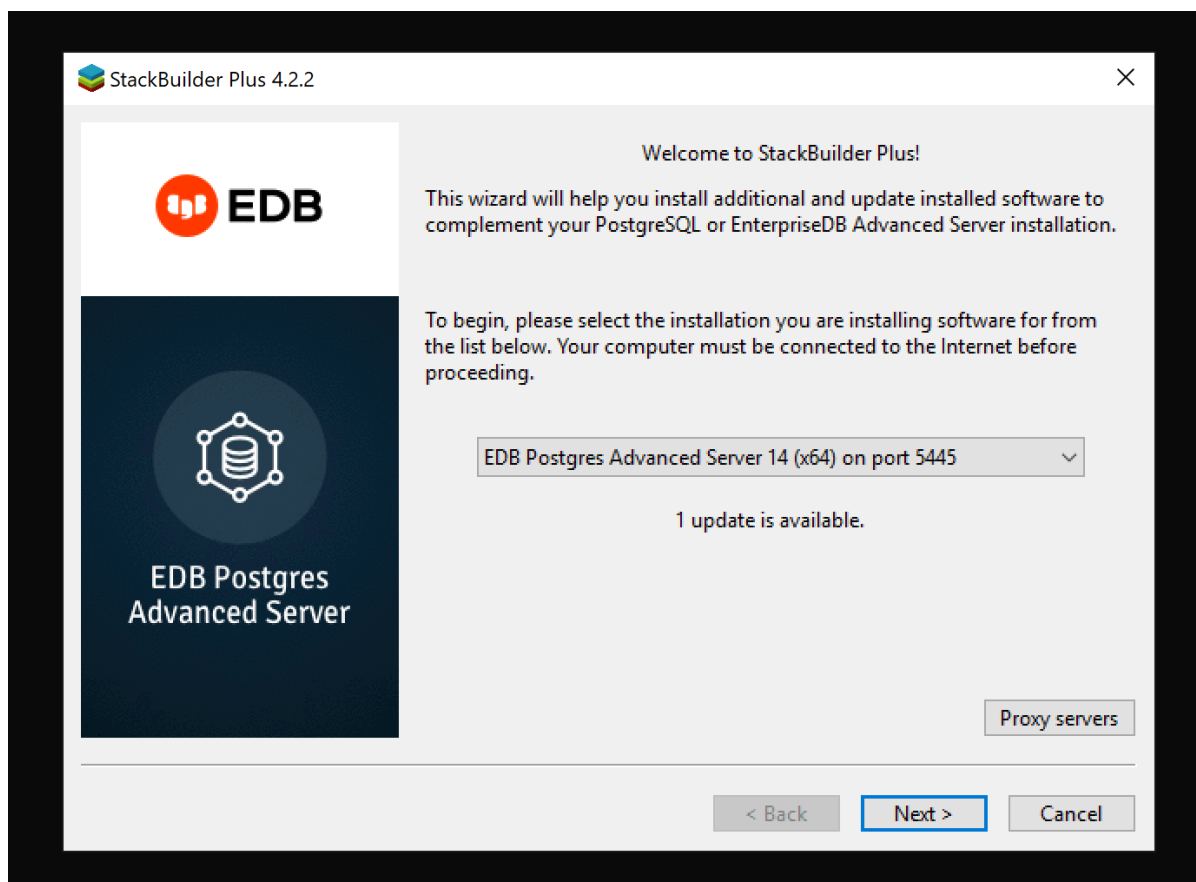


図 1: The Stack Builder welcome window.

- Stack Builder のウェルカムウィンドウ*

ドロップダウンリストボックスを使用して、使用可能なサーバーのリストからターゲットサーバーのインストールを選択します。ネットワークでプロキシサーバーを使用してインターネットにアクセスする必要がある場合は、Proxy servers ボタンを使用して Proxy servers ダイアログを開き、サーバーを指定し

ます。プロキシサーバーを使用する必要がない場合は、**Next** をクリックして、アプリケーション選択ウィンドウを開きます。

StackBuilder Plus を使用して Migration Toolkit を EDB Postgres Advanced Server インストールに追加する場合は、ツリーコントロールの **Add-ons, tools and utilities** ノードを展開し、**EnterpriseDB Migration Toolkit** の横のチェックボックスをオンにします。**Next** をクリックして続行します。

!!! Note * Stack Builder を使用して PostgreSQL インストールに Migration Toolkit を追加する場合、ツリーコントロール (located under the Registration-required and trial products node) の **EnterpriseDB Tools** ノードを展開し、**Migration Toolkit** チェックボックスを選択します。**Next** をクリックして続行します。

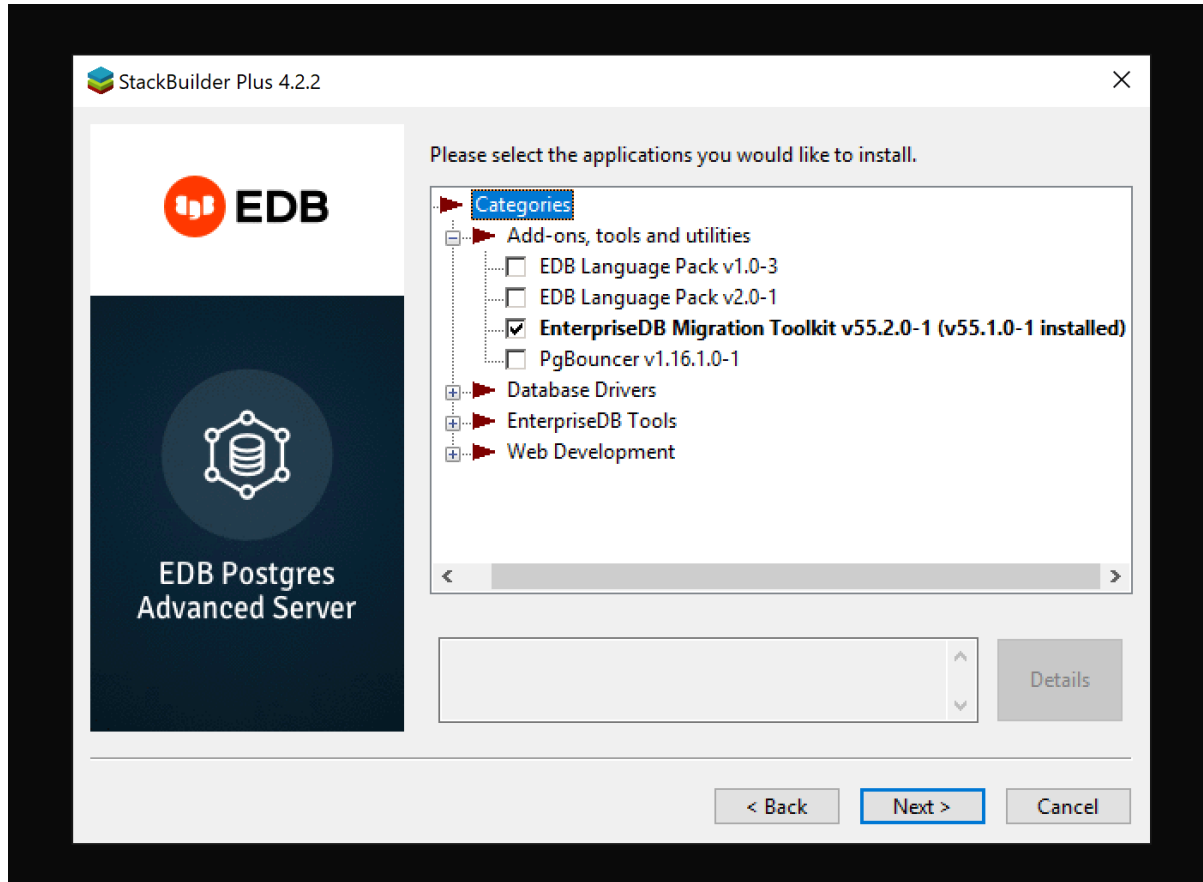


図 2: The Stack Builder application selection window.

- Stack Builder アプリケーション選択ウィンドウ*

Migration Toolkit が **Selected Packages** リストに含まれていること、および **Download directory** フィールドに許容されるダウンロード場所が含まれていることを確認します。

- Stack Builder の選択確認ウィンドウ*

Next をクリックして、Migration Toolkit のダウンロードをスタートします。

- Stack Builder はファイルのダウンロードを確認します*

ダウンロードが完了すると、Stack Builder はインストールファイルが正常にダウンロードされたことを確認します。**Next** を選択して、Migration Toolkit インストールウィザードを開きます。

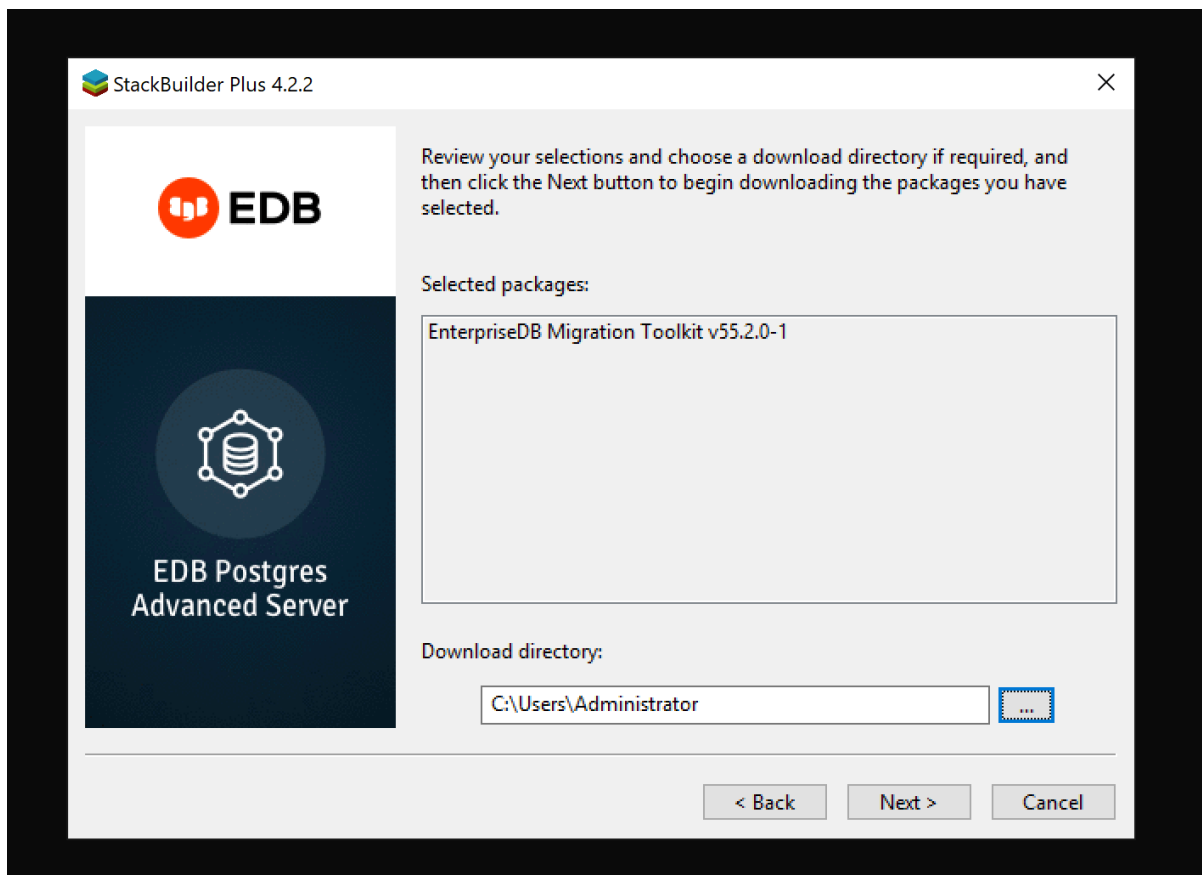


図 3: The Stack Builder selection confirmation window.

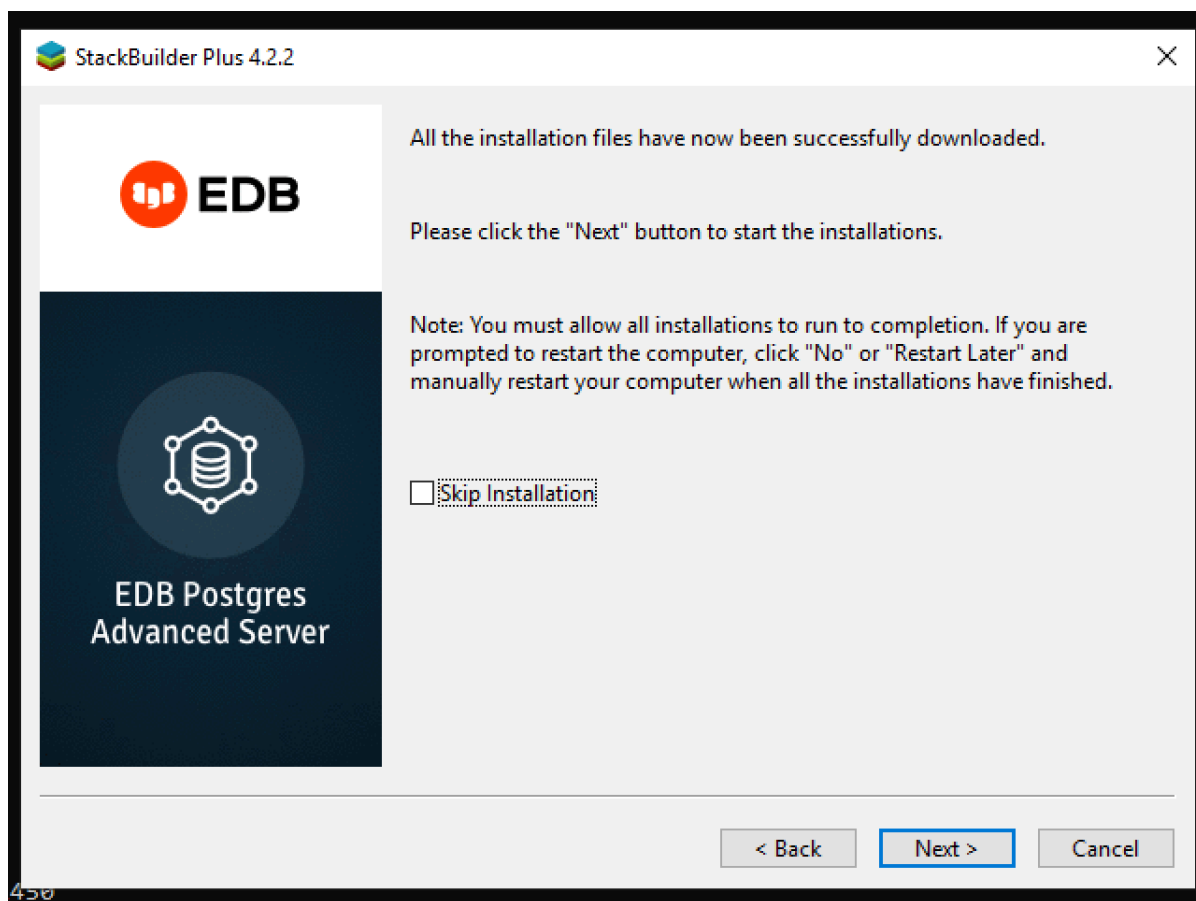


図 4: Stack Builder confirms the file download.

Migration Toolkit インストールウィザードでプロンプトが表示されたら、インストールの言語を指定し、OK をクリックして続行します。

Migration Toolkit インストールウィザードが開きます。Next をクリックして続行します。

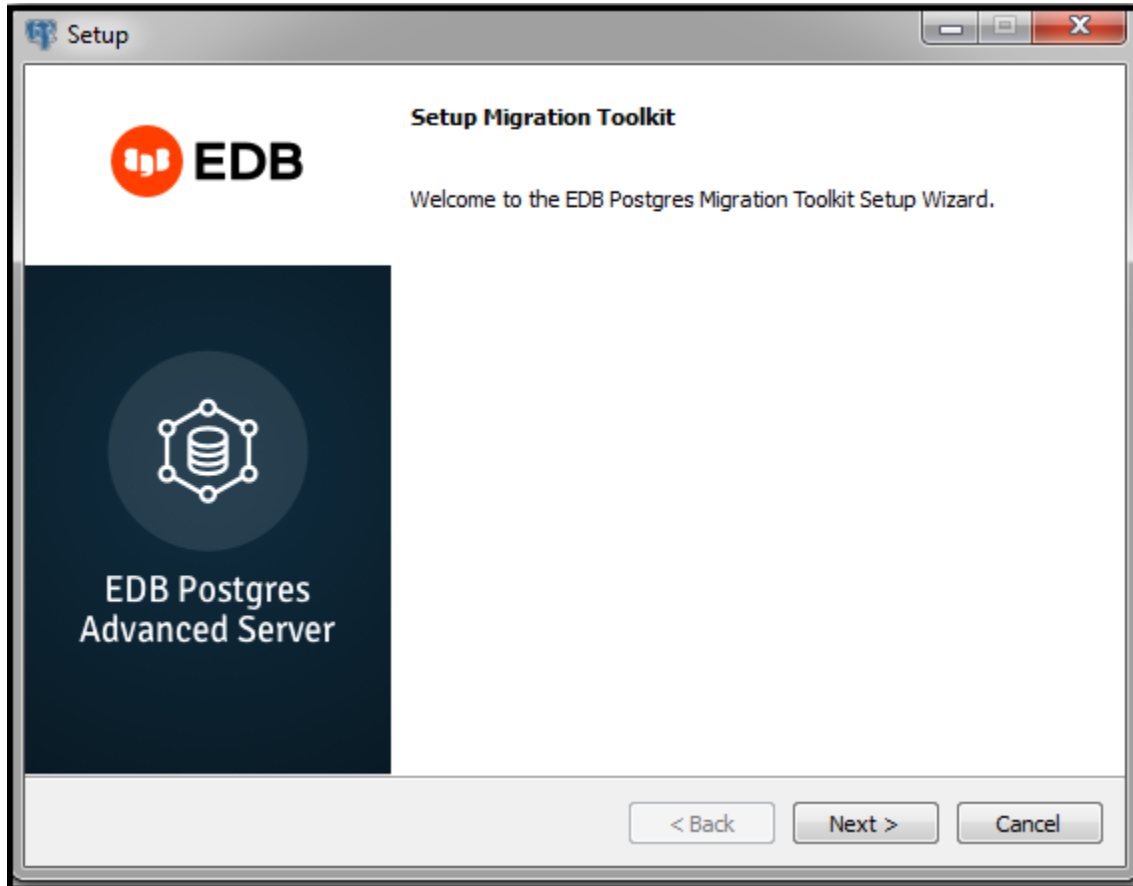


図 5: The Migration Toolkit installation wizard.

- Migration Toolkit インストールウィザード*

適切なラジオボタンをハイライトする前に、ライセンス契約を注意深く確認してください。Next をクリックして続行します。

- EnterpriseDB ライセンス契約*

デフォルトでは、Migration Toolkit は `mtk` ディレクトリにインストールされます。表示されているデフォルトのインストールディレクトリを受け入れるか、ディレクトリを変更し、Next をクリックして続行します。

- Migration Toolkit のインストールディレクトリを指定*

インストールウィザードは、Setup プログラムが Migration Toolkit をインストールする準備ができていることを確認します。Next をクリックしてインストールをスタートします。

- セットアップで Migration Toolkit のインストールをスタートする準備ができました*

ダイアログで、Migration Toolkit のインストールが完了したことを確認します。Finish をクリックして、Migration Toolkit インストーラを終了します。

- Migration Toolkit ウィザードが終了しました*



図 6: The EnterpriseDB license agreement.

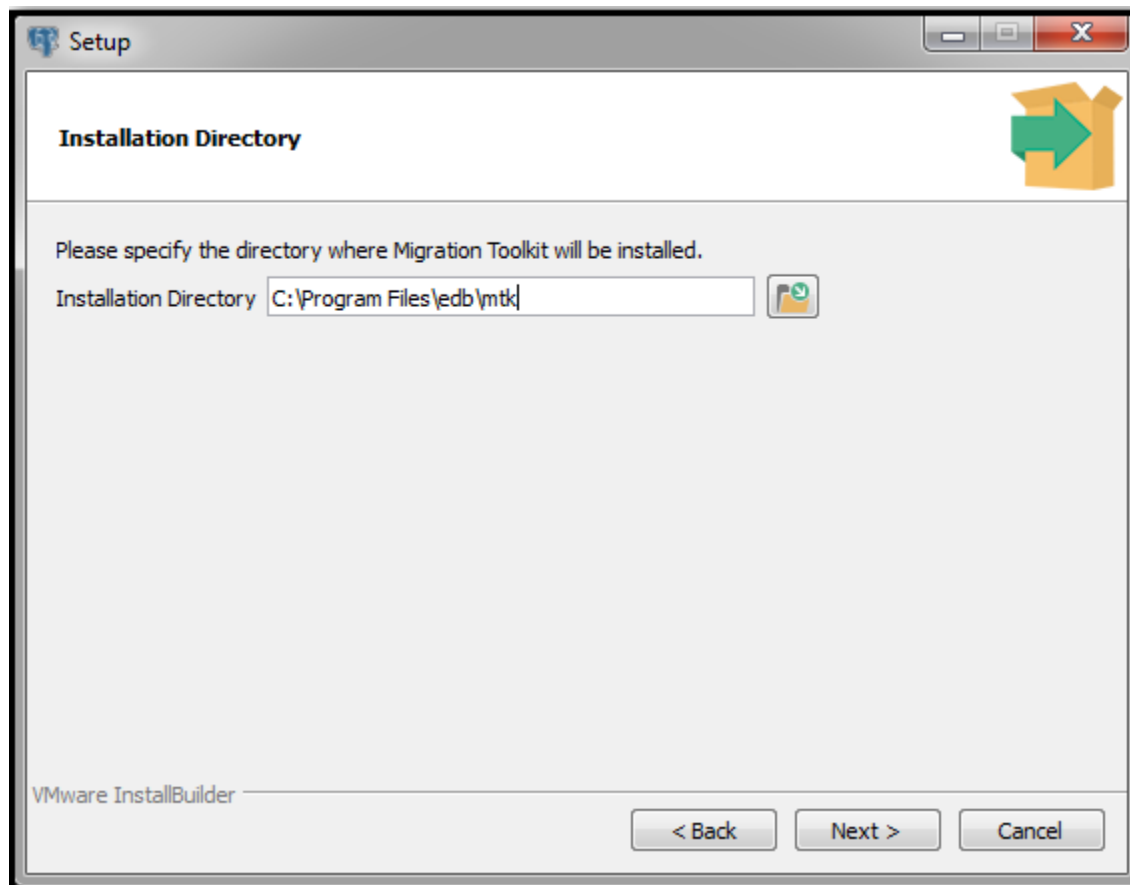


図 7: Specify a Migration Toolkit installation directory.

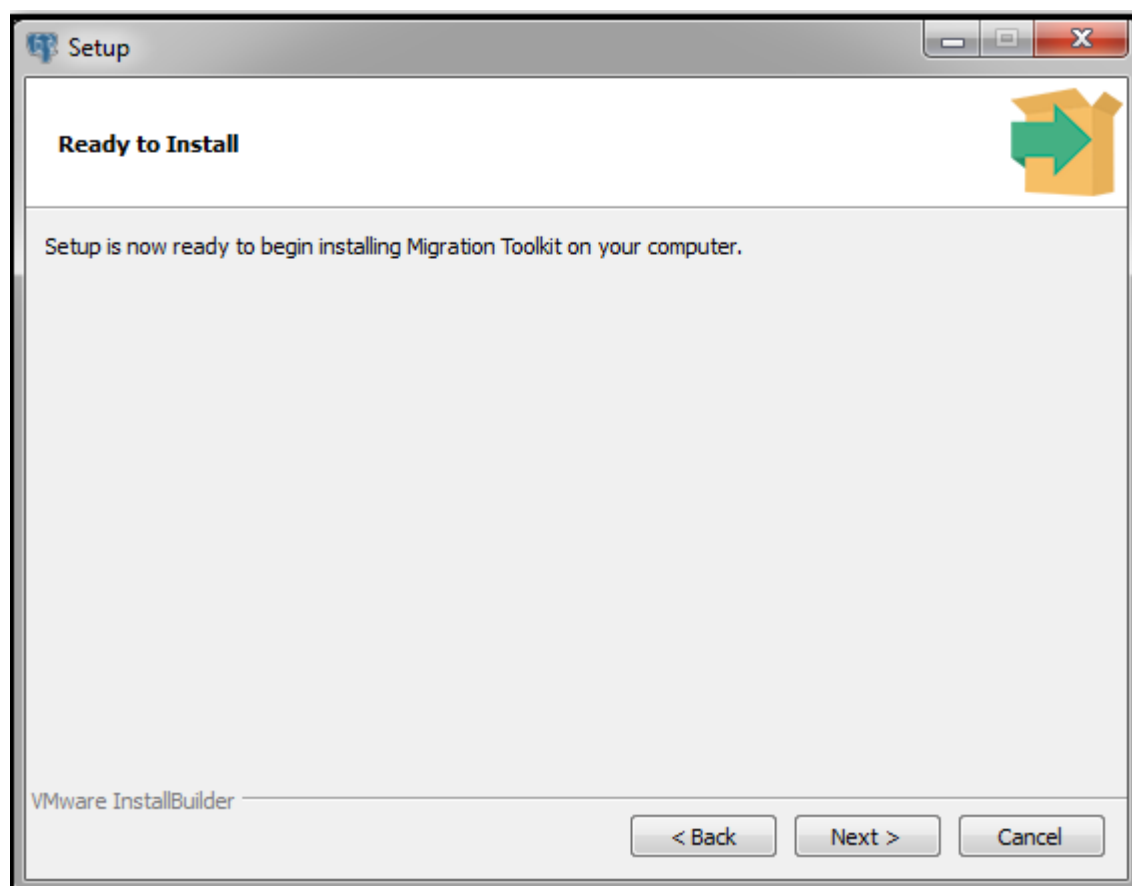


図 8: Setup is ready to start the Migration Toolkit installation.

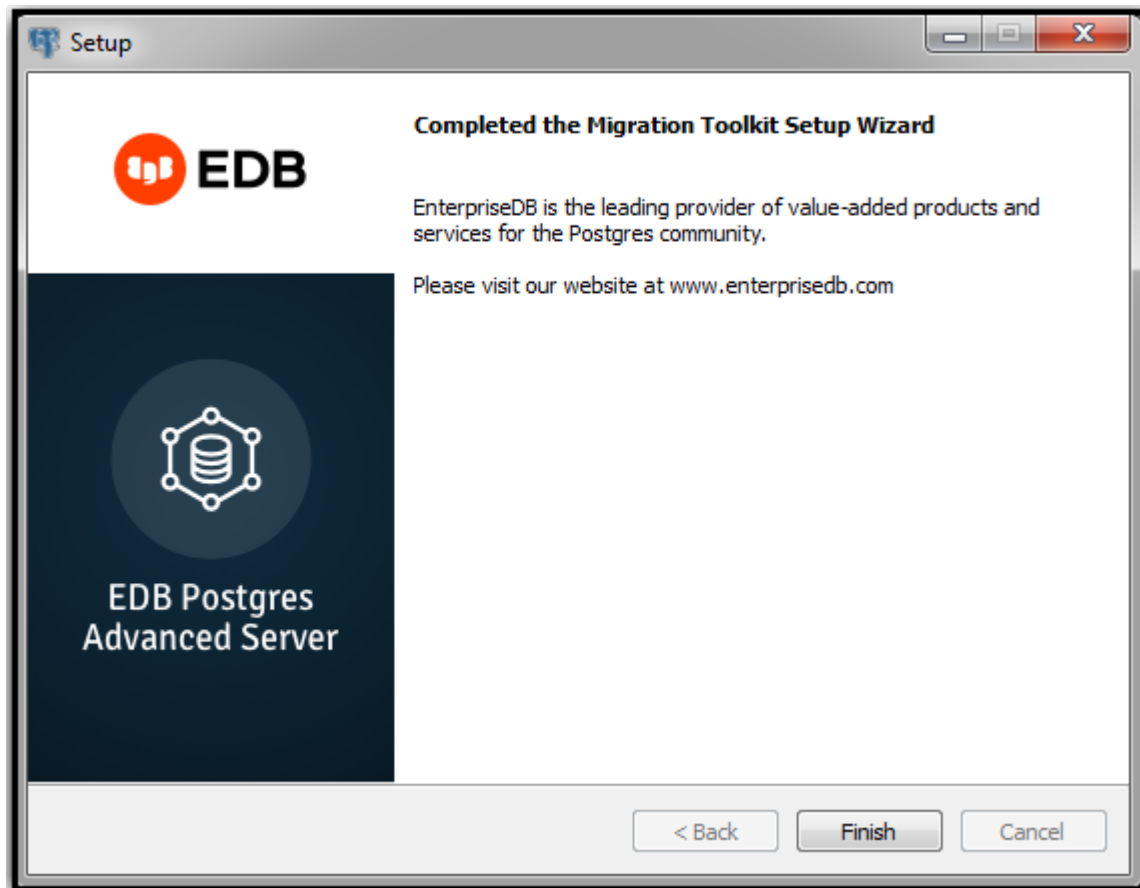


図 9: The Migration Toolkit wizard has finished.

Stack Builder が最後に選択したコンポーネントのインストールを完了すると、Installation Completed ウィンドウが表示されます。Finish をクリックして Stack Builder を閉じます。

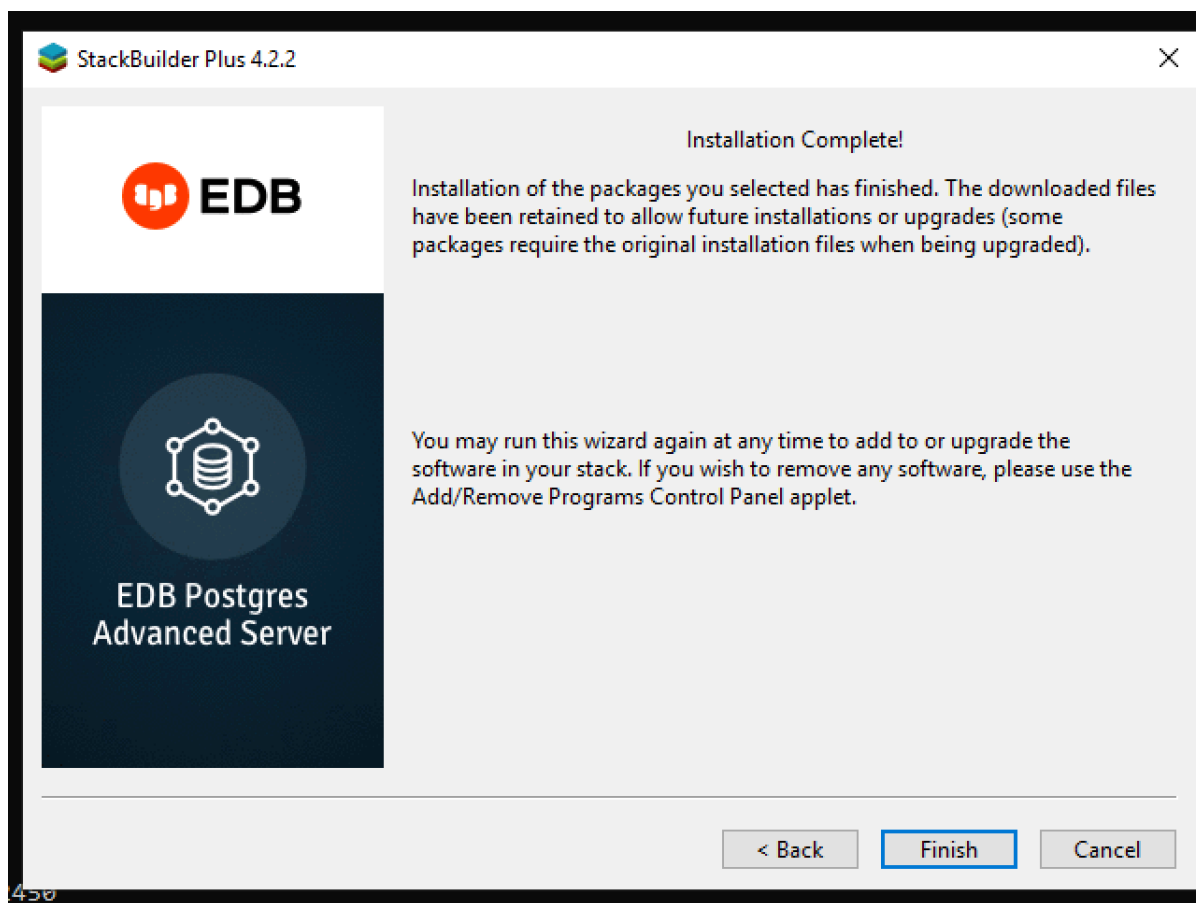


図 10: Stack Builder confirms the installation is complete.

- Stack Builder はインストールが完了したことを確認します*

Stack Builder で Migration Toolkit をインストールした後、移行を実行する前に適切なソース固有のドライバーをインストールする必要があります。詳細については、*Installing Source-Specific Drivers* を参照してください。

</ div>

4.6 Mac OS X に Migration Toolkit をインストールする

グラフィカルインストーラを使用して、Mac OS X に Migration Toolkit をインストールできます。インストーラをダウンロードするには、www.enterprisedb.com にアクセスしてください。

インストーラを起動するには：

- ターミナルでファイルパスに移動します。
 - /User/Directory>/Downloads/edb-migrationtoolkit-55.2.0-1-osx.app/Contents/MacOS

- スーパーユーザ特権を想定して、次のコマンドを呼び出します。

```
* ./installbuilder.sh
```

1. Language Selection ポップアップが開いたら、インストール言語を選択し、OK をクリックして Setup ウィンドウに進みます。
2. Migration Toolkit インストールウィザードが開きます。Next をクリックして続行します。

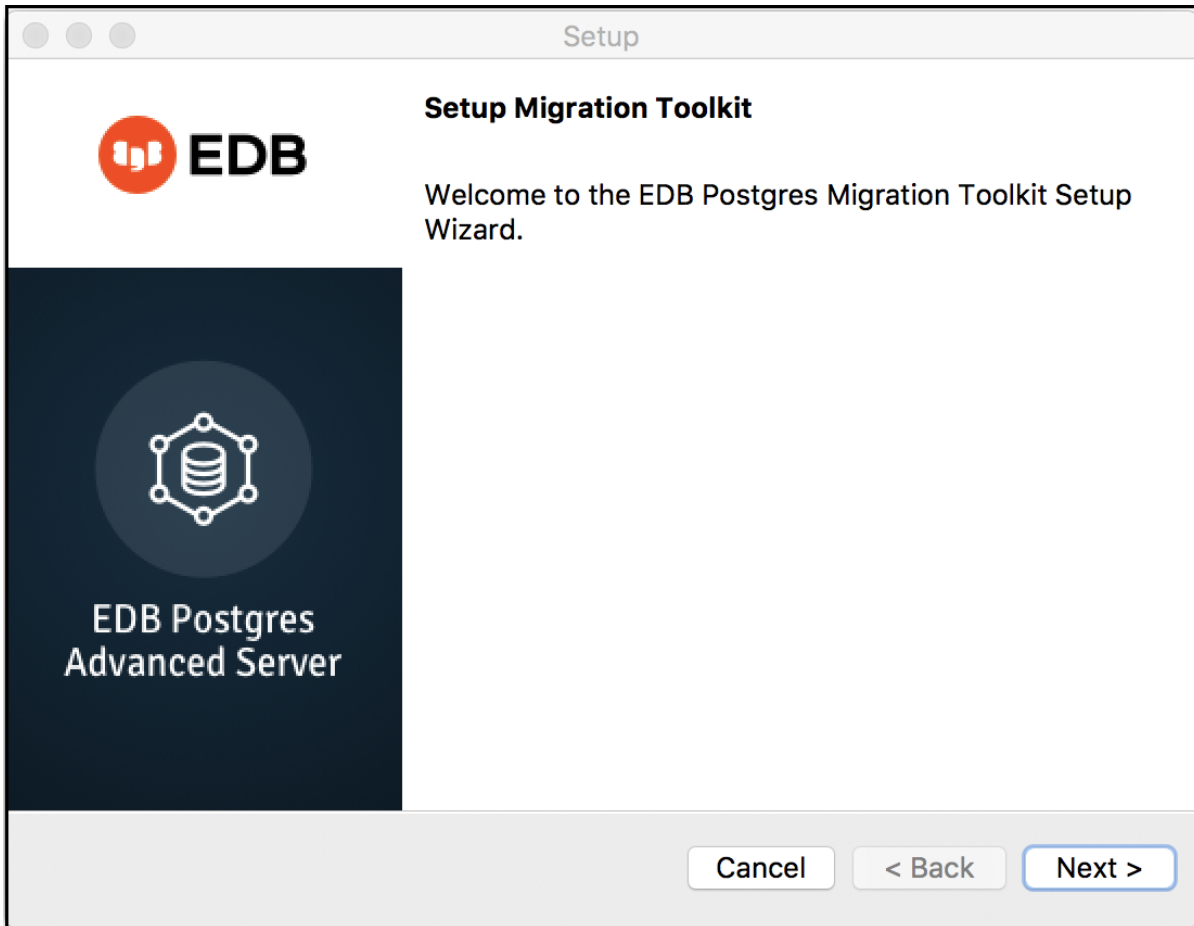


図 11: The Migration Toolkit installation wizard.

- Migration Toolkit インストールウィザード*

3. 適切なラジオボタンをハイライトする前に、ライセンス契約を注意深く確認します。Next をクリックして続行します。

- EnterpriseDB ライセンス契約*

デフォルトでは、Migration Toolkit は mtk ディレクトリにインストールされます。表示されているデフォルトのインストールディレクトリを受け入れるか、ディレクトリを変更し、Next をクリックして続行します。

- Migration Toolkit のインストールディレクトリを指定*

インストールウィザードは、Setup プログラムが Migration Toolkit をインストールする準備ができていることを確認します。Next をクリックしてインストールをスタートします。

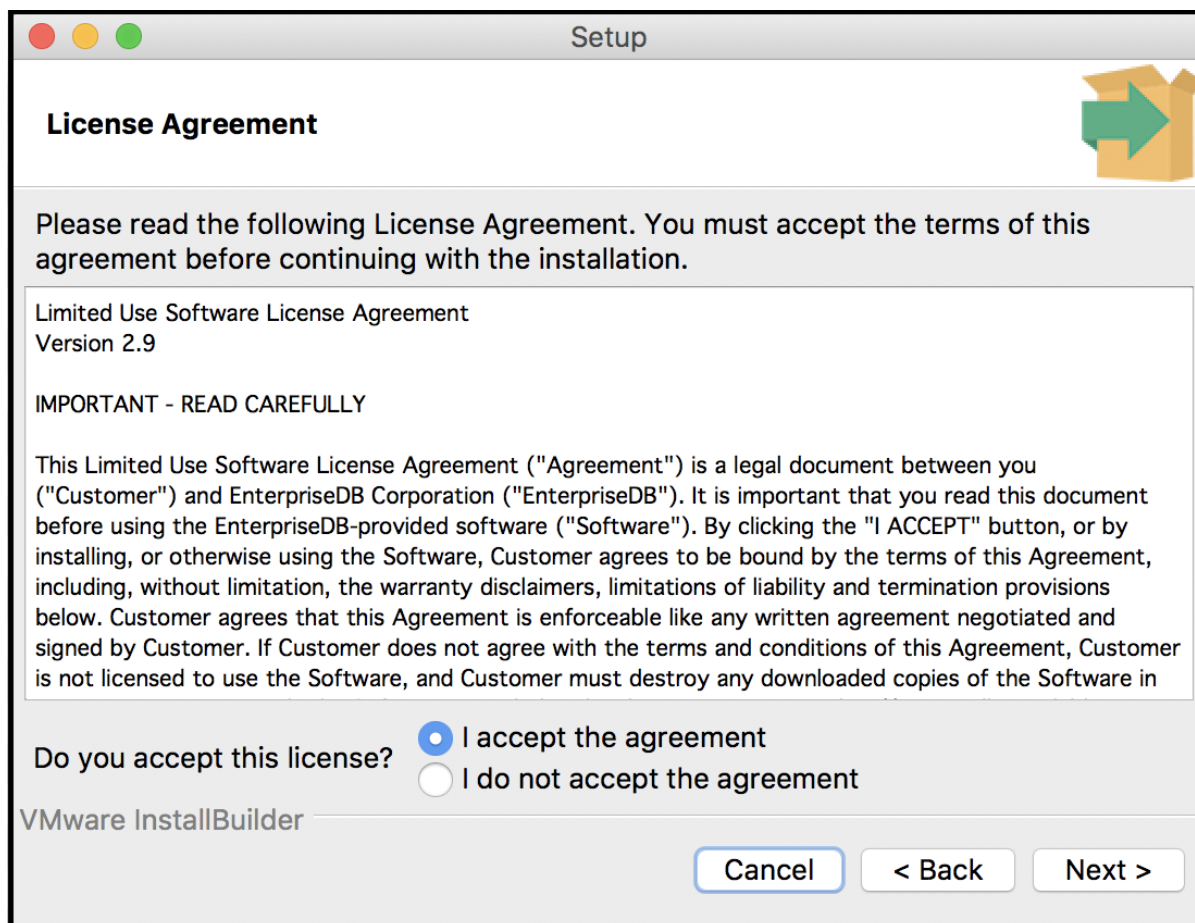


図 12: The EnterpriseDB license agreement.

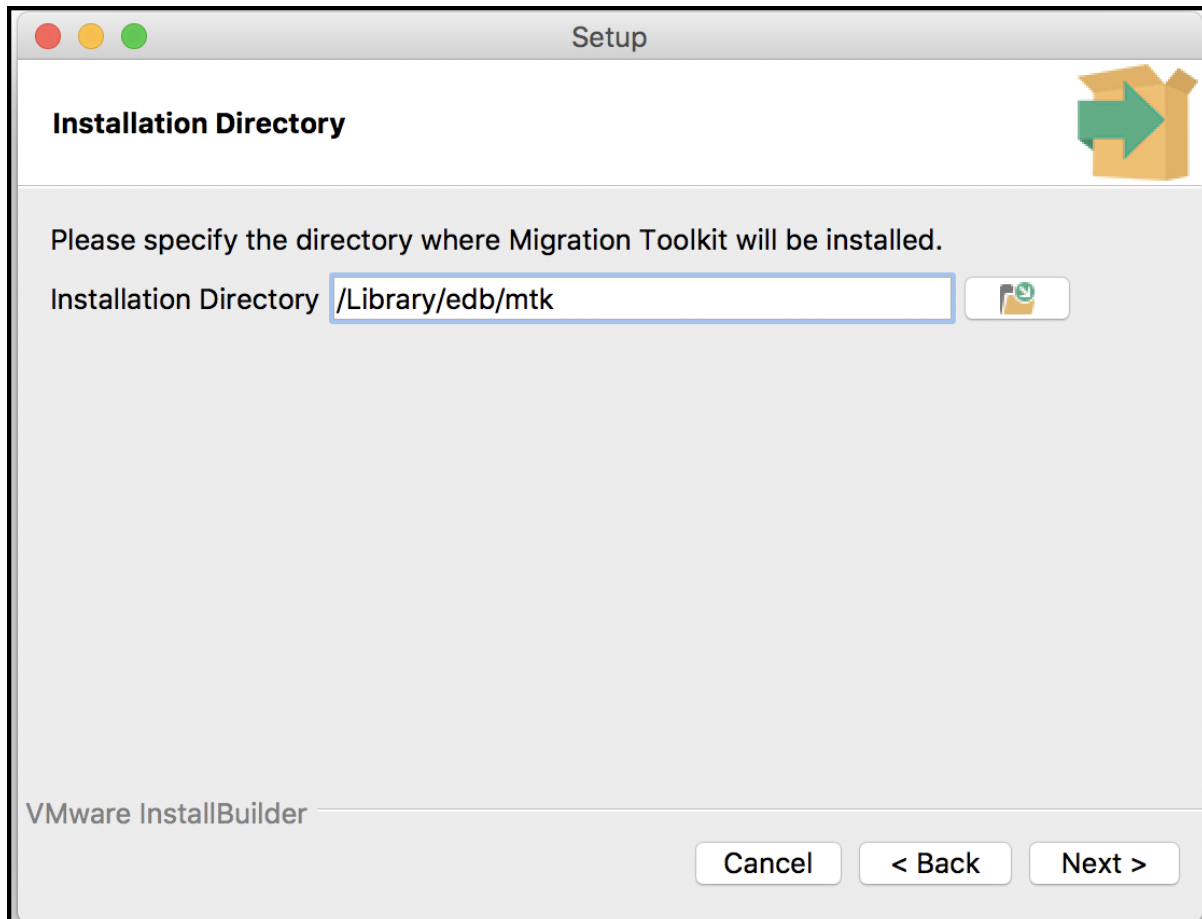


図 13: Specify a Migration Toolkit installation directory.

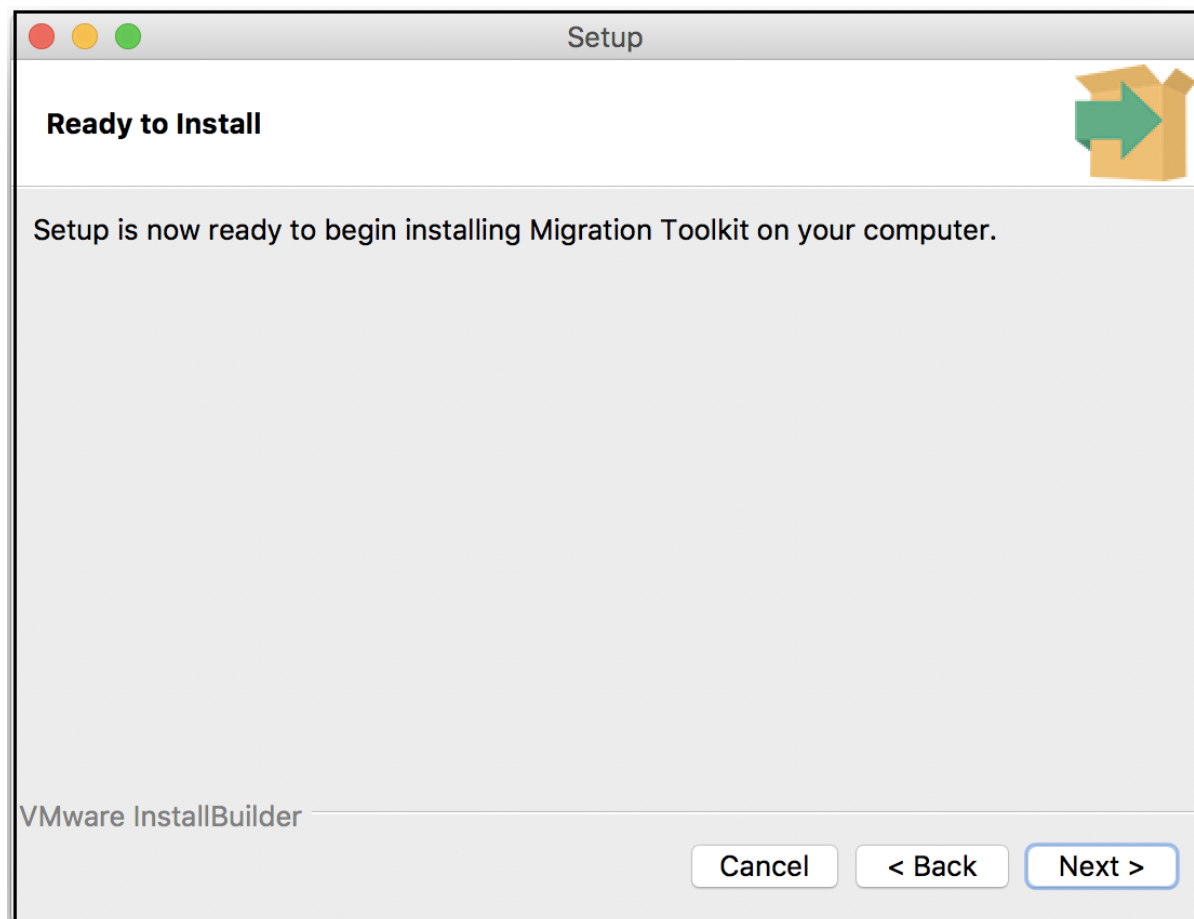


図 14: Setup is ready to start the Migration Toolkit installation.

- セットアップで Migration Toolkit のインストールをスタートする準備ができました*

ダイアログで、Migration Toolkit のインストールが完了したことを確認します。Finish をクリックして、Migration Toolkit インストーラを終了します。

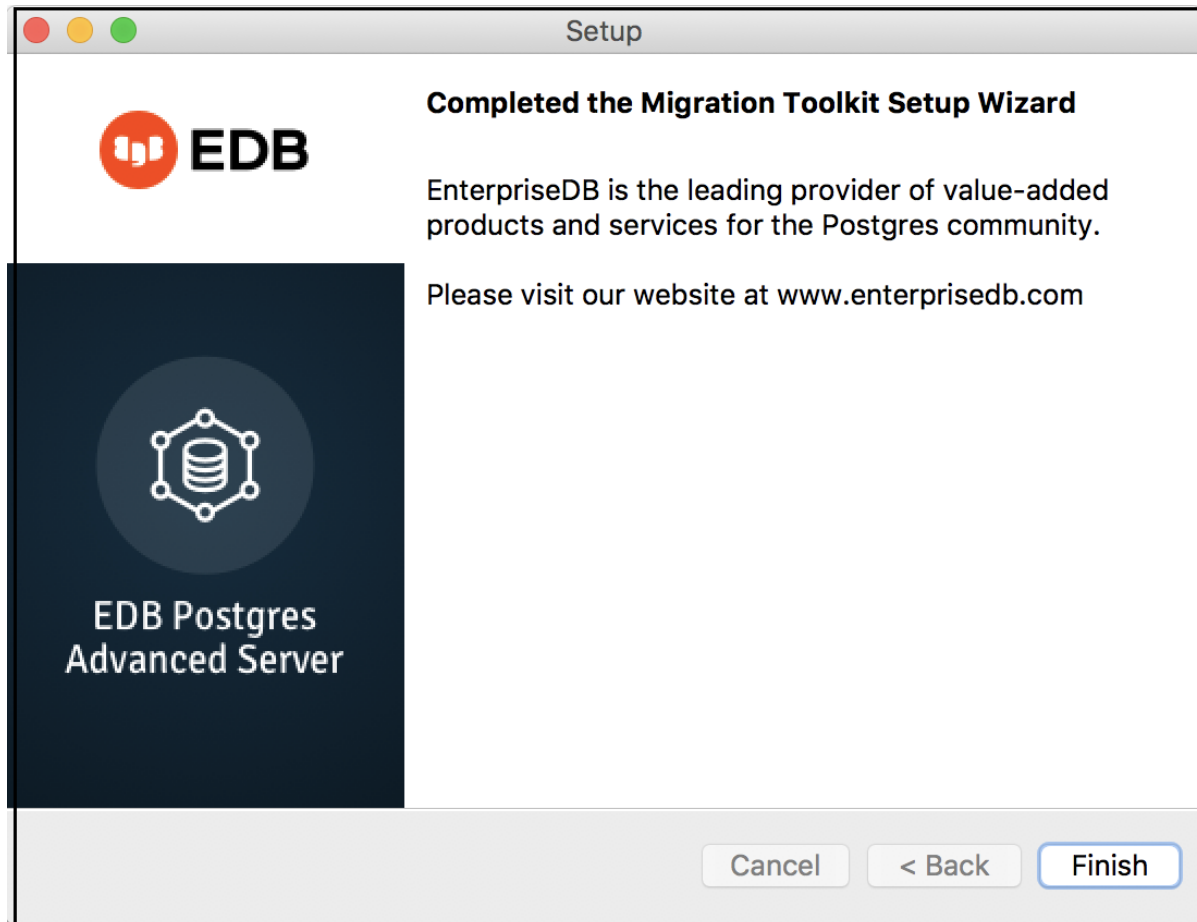


図 15: The Migration Toolkit wizard has finished.

- Migration Toolkit ウィザードが終了しました。*

</div>

4.7 ソース固有のドライバーのインストール

Migration Toolkit を呼び出す前に、無料で入手可能なソース固有のドライバをダウンロードしてインストールする必要があります。ドライバをダウンロードするには、またはベンダのダウンロードサイトへのリンクについては、EnterpriseDB ウェブサイトの Advanced Downloads ページの Third Party JDBC Drivers セクションにアクセスしてください。

<https://www.enterprisedb.com/software-downloads-postgres#third-party-jdbc-drivers>

ソース固有のドライバをダウンロードした後、ドライバファイルを<mtk_install_dir>/lib ディレクトリに移動します。

第5章 Building the toolkit.properties file

Migration Toolkit は、移行プロセス中に `toolkit.properties` ファイルに保存されている構成および接続情報を使用して、ソースデータベースとターゲットデータベースを識別して接続します。

Linux では、`toolkit.properties` ファイルは `/usr/edb/migrationtoolkit/etc` にあります。

Windows では、ファイルは `C:\Program Files\edb\mtk\etc` にあります。

図は、サンプルの `toolkit.properties` ファイルを示しています。

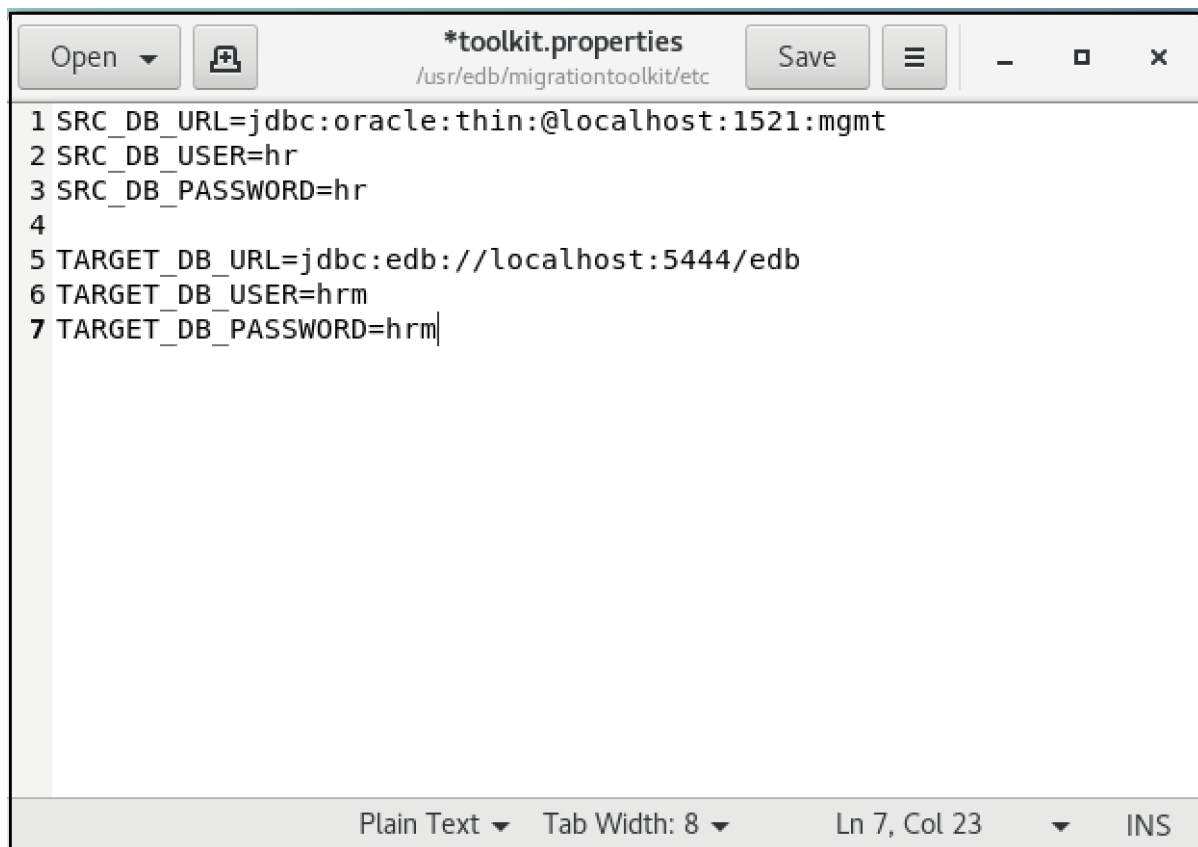


図 1: A typical toolkit.properties file.

Migration Toolkit コマンドを実行する前に、選択したエディタで `toolkit.properties` ファイルを変更します。ファイルを更新して、次の情報を含めます。

- `SRC_DB_URL` は、Migration Toolkit がソースデータベースに接続する方法を指定します。各ソースデータベースの URL の形成の詳細は次のとおりです。
- `SRC_DB_USER` は、ソースデータベースで（十分な特権を持つ）ユーザ名前を指定します。
- `SRC_DB_PASSWORD` は、ソースデータベースユーザのパスワードを指定します。

- TARGET_DB_URL は、ターゲットデータベースの JDBC URL を指定します。
- TARGET_DB_USER は、特権ターゲットデータベースユーザの名前を指定します。
- TARGET_DB_PASSWORD は、ターゲットデータベースユーザのパスワードを指定します。

BigAnimal でターゲットデータベースを指定するには、*Defining target information for BigAnimal* を参照してください。

</div>

5.1 Advanced Server URL の定義

Migration Toolkit は、次のプラットフォームから Advanced Server への移行に役立ちます。

- オラクル PostgreSQL
- MySQL
- Sybase SQL Server

各データベースタイプから移行されたオブジェクトの最終的なリストについては、[Functionality overview](#) を参照してください。

Migration Toolkit は、`toolkit.properties` ファイルからソースデータベースとターゲットデータベースの接続仕様を読み取ります。それぞれの接続情報には以下が含まれている必要があります。

- データベースの URL
- 特権ユーザの名前
- 指定されたユーザに関連付けられたパスワード

URL は JDBC 標準に準拠し、次のフォームを取ります。

`{TARGET_DB_URL|SRC_DB_URL}=jdbc:edb://<host>:<port><database_id>`

Advanced Server URL には次の情報が含まれています。

- `jdbc` & mdash; プロトコルは常に `jdbc` です。
- `edb` & mdash; Advanced Server を使用している場合、サブプロトコル値に `edb` を指定します。
- `<host>` & mdash; Postgres インスタンスが実行されているホストの名前または IP アドレス。
- `<port>` & mdash; Advanced Server データベースリスナーがモニタリングしているポート番号。デフォルトポート番号は 5444 です。
- `<database_id>` & mdash; ソースデータベースまたはターゲットデータベースの名前。
- `{TARGET_DB_USER|SRC_DB_USER}` & mdash; 移行される各タイプのオブジェクトを作成する権限を持つユーザを指定します。データをテーブルに移行する場合、指定されたユーザは、各ターゲットテーブルの挿入、切り捨て、および参照の特権も必要とする場合があります。
- `{TARGET_DB_PASSWORD|SRC_DB_PASSWORD}` & mdash; 特権 Advanced Server ユーザのパスワードに設定します。

</div>

5.2 PostgreSQL URL の定義

Migration Toolkit は、次のプラットフォームから PostgreSQL への移行を支援します。

- オラクル
- Advanced Server
- MySQL SQL Server

各データベースタイプから移行されたオブジェクトの最終的なリストについては、[Functionality overview](#) を参照してください。

Migration Toolkit は、`toolkit.properties` ファイルからソースデータベースとターゲットデータベースの接続仕様を読み取ります。それぞれの接続情報には以下が含まれている必要があります。

- データベースの URL
- 特権ユーザの名前
- 指定されたユーザに関連付けられたパスワード

PostgreSQL URL は JDBC 標準に準拠し、次のフォームを取ります。

```
{SRC_DB_URL|TARGET_DB_URL}=jdbc:postgresql://<host>:<port>/<database_id>
```

URL には次の情報が含まれます。

- `jdbc` & mdash; プロトコルは常に `jdbc` です。
- `postgresql` & mdash; PostgreSQL を使用している場合、サブプロトコル値に `postgresql` を指定します。
- `<host>` & mdash; Postgres インスタンスが実行されているホストの名前または IP アドレス。
- `<port>` & mdash; Postgres データベースリスナーがモニタリングしているポート番号。デフォルトポート番号は 5432 です。
- `<<database_id>` & mdash; ソースデータベースまたはターゲットデータベースの名前。
- `{SRC_DB_USER|TARGET_DB_USER}` & mdash; 移行される各タイプのオブジェクトを作成する権限を持つユーザを指定します。データをテーブルに移行する場合、指定されたユーザは、各ターゲットテーブルの挿入、切り捨て、および参照権限も必要になる場合があります。
- `{SRC_DB_PASSWORD|TARGET_DB_PASSWORD}` & mdash; 特権 PostgreSQL ユーザのパスワードに設定します。

</div>

5.3 BigAnimal URL の定義

Migration Toolkit は、次のプラットフォームから BigAnimal への移行を支援します。

- オラクル
- Advanced Server PostgreSQL
- MySQL SQL Server

Migration Toolkit は、`toolkit.properties` ファイルからソースデータベースとターゲットデータベースの接続仕様を読み取ります。それぞれの接続情報には以下が含まれている必要があります。

- データベースの URL
- 特権ユーザの名前
- 指定されたユーザに関連付けられたパスワード

BigAnimal に移行する場合、`TARGET_DB_URL` は JDBC URL のフォームを取ります。例：

URL には次の情報が含まれます。

- `jdbc & mdash;` プロトコルは常に `jdbc` です。
- `postgres_type & mdash;` サブプロトコルは `Postgres` タイプです。Advanced Server を使用している場合は `edb` を、PostgreSQL を使用している場合は `postgresql` を指定します。
- `<host_name> & mdash;` クラスターのホスト名前。BigAnimal ポータルの**接続タブ**の**ホストフィールド**からコピーできます。
- `<port> & mdash;` データベースリスナーがモニタリングしているポート番号。BigAnimal ポータルの**接続タブ**の**ポートフィールド**からコピーできます。
- `<database_id> & mdash;` ターゲットデータベースの名前。これを、移行ターゲットデータベースとして使用するクラスター内のデータベースの名前に設定します。クラスターのデフォルトデータベースの名前は、BigAnimal ポータルの**接続タブ**の**** Dbname ****フィールドに表示されます。多くの場合、移行ターゲットとして使用するために個別のデータベースが作成されます。
- `TARGET_DB_USER & mdash;` 特権データベースユーザの名前を指定します。BigAnimal ポータルの**接続タブ**の**ユーザーフィールド**からコピーできます。
- `TARGET_DB_PASSWORD & mdash;` 指定されたユーザのパスワードが含まれます。
- `sslmode & mdash;` 「require」または「verify-full」のいずれか。[Recommended settings for SSL mode](#)を参照してください。BigAnimal ポータルの**接続タブ**の**サービス URI** 値の最後にリストされます。

5.4 オラクル URL の定義

Migration Toolkit は、オラクルデータベースから PostgreSQL または Advanced Server データベースへの移行を支援します。オラクルから移行する場合、`toolkit.properties` ファイルでオラクルソースデータベースの接続仕様を指定する必要があります。接続情報には以下を含める必要があります。

オラクルデータベースの URL - 特権ユーザの名前 - 指定されたユーザに関連付けられたパスワード

オラクルデータベースから移行する場合、`SRC_DB_URL` には 2 つの形式のいずれかで指定された JDBC URL が含まれている必要があります。最初のフォームは次のとおりです。

```
jdbc:oracle:thin:@<host_name>:<port>:<database_id>
```

2 番目のフォームは次のとおりです。

```
jdbc:oracle:thin:@//<host_name>:<port>{<database_id|service_name>}
```

オラクル URL には次の情報が含まれます。

- `jdbc` & mdash; プロトコルは常に `jdbc` です。
- `oracle` & mdash; サブプロトコルは常に `oracle` です。
- `thin` & mdash; ドライバのタイプ。 `thin` のドライバタイプを指定します。
- `<host_name>` & mdash; オラクルサーバーが実行されているホストの名前または IP アドレス。
- `<port>` & mdash; オラクルデータベースリスナーがモニタリングしているポート番号。
- `<database_id>` & mdash; オラクルデータベースのデータベース SID。
- `<service_name>` & mdash; オラクルサービスの名前。
- `SRC_DB_USER` & mdash; 特権オラクルユーザの名前を指定します。オラクルユーザがオブジェクトをオラクルから Advanced Server に移行するには、DBA 権限がニーズです。オラクル `GRANT DBA TO user` コマンドを使用して DBA 権限をオラクルユーザに付与し、目的のデータベースオブジェクトを保証移行することができます。
- `SRC_DB_PASSWORD` & mdash; 指定されたユーザのパスワードが含まれます。

</div>

5.5 MySQL URL の定義

Migration Toolkit は、MySQL データベースから Advanced Server または PostgreSQL データベースへの移行を支援します。MySQL から移行する場合、`toolkit.properties` ファイルで MySQL ソースデータベースの接続仕様を指定する必要があります。接続情報には以下を含める必要があります。

- ソースデータベースの URL
- 特権ユーザの名前
- 指定されたユーザに関連付けられたパスワード

MySQL から移行する場合、SRC_DB_URL は JDBC URL のフォームを取ります。例：

```
jdbc:mysql://<host_name>[:<port>]/<database_id>
```

•

URL には次の情報が含まれます。

- jdbc & mdash; プロトコルは常に jdbc です。
- mysql & mdash; サブプロトコルは常に mysql です。
- <host_name> & mdash; ソースサーバーが実行されているホストの名前または IP アドレス。
- [<port>] & mdash; MySQL データベースリスナーがモニタリングしているポート番号。
- <database_id> & mdash; ソースデータベースの名前。
- SRC_DB_USER & mdash; 特権 MySQL ユーザの名前を指定します。
- SRC_DB_PASSWORD & mdash; 指定されたユーザのパスワードが含まれます。

!!! Note * -データソース tinyInt(1) を使用して、0 および 1 以外のバイト値を MySQL ソースデータベースに格納 make 場合、MySQL Connector / J JDBC Driver URL にオプションのパラメータ? tinyInt1isBit=false を追加してください。

- -MySQL Connector / J JDBC Driver 8.0.26 のバグにより、特定の場合に外部キーコンストレインの移行が失敗します。EDB は、Migration Toolkit を使用したデータの移行にこのドライバを使用しないことをお勧めします。代わりに MySQL Connector / J JDBC Driver 8.0.25 ドライバを使用できます。
- このバグの詳細については、[MySQL bug report] (<https://bugs.mysql.com/bug.php?id=95280>) を参照してください。

TINYINT(1) は、PostgreSQL/ EDB Postgres Advanced Server の BIT(1) にマップされます（場合によっては予期されない場合があります）。ただし、MySQL JDBC ドライバが BIT(1) として報告するため、Migration Toolkit はそれを PostgreSQL/ EDB Postgres Advanced Server の BIT(1) にマッピングします。

値-128～127 を保持できる TINYINT(1) とは異なり、BIT(1) は 0 と 1 のみを保持できます。したがって、このフィールドに 0 と 1 以外の値を格納する場合は、MySQL JDBC ドライバのデフォルトの動作を変更する必要がありますドライバがそれを TINYINT として報告するようにします。

この動作を変更するには、接続文字列で次の URL オプションを設定します。

```
jdbc:mysql://localhost:3306/world?tinyInt1isBit=false
```

この場合、JDBC ドライバは TINYINT(1) を TINYINT としてレポートし、ターゲット PostgreSQL/ EDB Postgres Advanced Server の SMALLINT にマップされます。

</div>

5.6 Sybase URL の定義

Migration Toolkit は、Sybase データベースから Advanced Server データベースへの移行を支援します。Sybase から移行する場合、`toolkit.properties` ファイルで Sybase ソースデータベースの接続仕様を指定する必要があります。接続情報には以下を含める必要があります。

- ソースデータベースの URL
- 特権ユーザの名前
- 指定されたユーザに関連付けられたパスワード

Sybase から移行する場合、`SRC_DB_URL` は JTDS URL のフォームを取ります。例：

```
jdbc:jtds:sybase://<host_name>[:<port>]/<database_id>
```

Sybase URL には次の情報が含まれます。

- `jdbc` & mdash; プロトコルは常に `jdbc` です。
- `jtds` & mdash; ドライバ名前は常に `jtds` です。
- `sybase` & mdash; サーバタイプは常に `sybase` です。
- `<host_name>` & mdash; ソースサーバが実行されているホストの名前または IP アドレス。
- `<port>` & mdash; Sybase データベースリスナがモニタリングしているポート番号。
- `<database_id>` & mdash; ソースデータベースの名前。
- `SRC_DB_USER` & mdash; 特権 Sybase ユーザの名前を指定します。
- `SRC_DB_PASSWORD` & mdash; 指定されたユーザのパスワードが含まれます。

5.7 SQL Server URL の定義

Migration Toolkit は、SQL Server データベースから PostgreSQL または Advanced Server データベースへの移行を支援します。Migration Toolkit は、次のオブジェクト定義の移行をサポートしています。

- スキーマ
- テーブル
- テーブルデータ
- Constraints
- インデックス

Migration Toolkit は、`toolkit.properties` ファイルからソースデータベースの接続仕様を読み取ります。接続情報には以下を含める必要があります。

- ソースデータベースの URL
- 特権ユーザの名前

- 指定されたユーザに関連付けられたパスワード

SQL Server データベースに接続している場合、SRC_DB_URL は JTDS URL のフォームを取ります。例：

`jdbc:jtds:sqlserver:\<server_name>\[:<port>]\<database_id>`

SQL サーバーの URL には次の情報が含まれます。

- `jdbc` & mdash; プロトコルは常に `jdbc` です。
- `jtds` & mdash; ドライバ名前は常に `jtds` です。
- `sqlserver` & mdash; サーバータイプは常に `sqlserver` です。
- `<server_name>` & mdash; ソースサーバーが実行されているホストの名前または IP アドレス。
- `<port>` & mdash; ソースデータベースリスナーがモニタリングしているポート番号。
- `<database_id>` & mdash; ソースデータベースの名前。
- `SRC_DB_USER` & mdash; 特権 SQL Server ユーザの名前を指定します。
- `SRC_DB_PASSWORD` & mdash; 指定されたユーザのパスワードが含まれます。

第6章 Invoking Migration Toolkit

Migration Toolkit をインストールし、`toolkit.properties` file でソースデータベースとターゲットデータベースの接続プロパティを指定すると、Migration Toolkit は移行を実行する準備が整います。

Migration Toolkit の実行可能ファイルの名前付けは、Linux システムでは `runMTK.sh`、Windows システムでは `runMTK.bat` です。Linux システムでは、実行可能ファイルは次の場所にあります。

```
/usr/edb/migrationtoolkit/bin
```

Windows では、実行可能ファイルは次の場所にあります。

```
C:\Program Files\edb\mtk\bin
```

!!! Note * Migration Toolkit の起動時に次のエラーが表示される場合は、`toolkit.properties` ファイルのファイル権限を確認してください。

```
MTK-11015: The connection credentials file ../etc/toolkit.properties is
not secure and accessible to group/others users. This file contains
plain passwords and should be restricted to Migration Toolkit owner user
only.
```

Migration Toolkit を実行するオペレーティングシステムユーザアカウントは、ファイルの最小読み取りパーミッションを持つ `toolkit.properties` ファイルの所有者でなければなりません。さらに、グループおよび他のユーザーに対して、いかなる種類のアクセス許可も存在してはなりません。以下は、ユーザ `enterprisedb` が Migration Toolkit を実行している場合の推奨ファイル許可の例です。

```
-rw----- 1 enterprisedb enterprisedb 191 Aug 1 09:59 toolkit.properties
```

6.1 埋め込みたバイナリゼロ（NULL 文字）を含む文字データのインポート

Migration Toolkit は、値が NULL の列のインポートをサポートしています。ただし、Migration Toolkit は、JDBC 接続プロトコルでの NULL 文字値（埋め込みバイナリゼロ 0x00）のインポートをサポートしていません。NULL 文字を含むデータをインポートする場合は、`-replaceNullChar` オプションを使用して、NULL 文字を単一の非 NULL の置換文字に置き換えます。

!!! Note * `-MTK` は NULL 文字を暗黙的に空の文字列に置き換えます。`--replaceNullChar` オプションは、`-copyViaDBLinkOra` オプションでは機能しません。

データを移行したら、SQL ステートメントを使用して、`-replaceNullChar` で指定された文字をバイナリのゼロに置き換えます。

6.2 オラクルからスキーマを移行する

コマンドラインで指定されていない限り、Migration Toolkit は、ソースデータベースがオラクルで、ターゲットデータベースが EDB Postgres Advanced Server であると想定しています。Linux で完全なスキーマを移行するには、実行可能ファイルに移動し、次のコマンドを呼び出します。

```
$ ./runMTK.sh <schema_name>
```

Windows で完全なスキーマを移行するには、実行可能ファイルに移動し、次のコマンドを呼び出します。

```
> .\runMTK.bat <schema_name>
```

どこで：

<schema_name>は、移行する toolkit.properties ファイルで指定されたソースデータベース内のスキーマの名前です。少なくとも 1 つの schema_name 値を含める必要があります。

!!! Note * -移行されたスキーマのデフォルトデータベースユーザが自動的に移行される場合、カスタムプロファイルが存在する場合、デフォルトデータベースユーザのカスタムプロファイルも移行されます。カスタムプロファイルは、ユーザーが作成したプロファイルです。例、カスタムプロファイルはオラクルプロファイル DEFAULT および MONITORING_PROFILE を除外します。PostgreSQL のデフォルト行のサイズは 8 KB に制限されています。これは、各テーブル行が単一の 8 KB ブロックに収まる必要があることを意味します。それ以外の場合、例、1600 列の INT を持つテーブルを作成し、すべての列にデータを挿入できることを示すエラーが発生します。ただし、INT は 4 バイトとして格納されますが、各 BIGINT にはより多くのスペース（8 バイト）が必要であるため、BIGINT 列で同じことはできません。詳細については、PostgreSQL 文書の [PostgreSQL Limits](#) を参照してください。

コマンド名前の後にスキーマ名のカンマ区切りリストを続けることで、マルチプルのスキーマを移行できます。

Linux では、次のコマンドを実行します。

```
$ ./runMTK.sh <schema_name1>,<schema_name2>,<schema_name3>
```

Windows では、次のコマンドを実行します。

```
> .\runMTK.bat <schema_name1>,<schema_name2>,<schema_name3>
```

</div>

6.3 Oracle 以外のソースデータベースからの移行

ソースデータベースタイプとターゲットデータベースタイプを指定しない場合、Postgres はソースデータベースがオラクルであり、ターゲットデータベースが EDB Postgres Advanced Server であると想定します。

Migration Toolkit を呼び出すには、コマンドウィンドウを開き、実行可能ファイルに移動して、次のコマンドを呼び出します。

```
$ ./runMTK.sh -sourcedbtype <source_type> -targetdbtype <target_type> [options, ...]  
<schema_name>;
```

```
-sourcedbtype <source_type>
```

<source_type>は、ソースデータベースのサーバータイプを指定します。<source_type>は大文字と小文字を区別しません。デフォルトでは、<source_type>は oracle です。source_type は、次の値のいずれかです。

To migrate from:	Specify:
Oracle	oracle (the default value)
MySQL	mysql
SQL Server	sqlserver
Sybase	sybase
PostgreSQL	postgres or postgresql
EDB Postgres Advanced Server	enterprisedb

-targetdbtype <target_type>

<target_type>は、ターゲットデータベースのサーバータイプを指定します。<target_type>は大文字と小文字を区別しません。デフォルトでは、<target_type>は enterprisedb です。<target_type>は次の値のいずれかです。

To migrate to:	Specify:
EDB Postgres Advanced Server	enterprisedb
PostgreSQL	postgres or postgresql

<schema_name>

<schema_name>は、移行する toolkit.properties ファイルで指定されたソースデータベース内のスキーマの名前です。少なくとも 1 つの<schema_name>値を含める必要があります。

次の例では、HR 名前付けのスキーマ（テーブル定義とテーブルコンテンツ）を Linux システム上の MySQL データベースから EDB Postgres Advanced Server ホストに移行します。このコマンドには、- sourcedbtype および -targetdbtype オプションが含まれています。

Linux では、次のコマンドを使用します。

```
$ ./runMTK.sh -sourcedbtype mysql -targetdbtype enterprisedb HR
```

Windows では、次のコマンドを使用します。

```
> .\runMTK.bat -sourcedbtype mysql -targetdbtype enterprisedb HR
```

Migration Toolkit コマンドの最後にスキーマのコンマ区切りリストを含めることにより、ソースデータベースからマルチプルのスキーマを移行できます。次の例では、マルチプルのスキーマ（HR および ACCTG 名前付け）を MySQL データベースから PostgreSQL データベースに移行します。

Linux では、次のコマンドを使用します。

```
$ ./runMTK.sh -sourcedbtype mysql -targetdbtype postgres HR,ACCTG
```

Windows では、次のコマンドを使用します。

```
> .\runMTK.bat -sourcedbtype mysql -targetdbtype postgres HR,ACCTG
```

第7章 Migration Toolkit command options

Migration Toolkit を実行するときに移行オプションを追加して、移行の詳細を制御します。例、データベース内のすべてのスキーマを移行するには、コマンドに `-allSchemas` オプションを追加します。

```
$ ./runMTK.sh -allSchemas
```

!!! Note * `--allSchemas` パラメータは、オラクル、EDB Postgres Advanced Server、および PostgreSQL ソースデータベースでのみサポートされます。Sybase、MS SQL Server、および MySQL ソースデータベースではサポートされていません。Migration Toolkit は、データの移行中にユーザーが作成したトリガーを無効にします。ただし、PostgreSQL/ EDB Postgres Advanced Server では、`ALTER TABLE` の `DISABLE TRIGGER USER` 句を使用して継承された子パーティショントリガーを無効にできないため、パーティションテーブルのデータ移行中にトリガーがアクティブになり、予期しない結果が生じる可能性があります。

Migration Toolkit で機能するコマンドオプションは、表に示すように、動作によってグループ化されています。

Feature	Relevant options
[Offline migration options](#offline-migration-options)	-offlineMigration
[Import options](#import-options)	-sourcedbtype, -targetdbtype, -schemaOnly, -dataOnly
[Schema creation options](#schema-creation-options)	-dropSchema, -targetSchema
[Schema object selection options](#schema-object-selection-options)	-allTables, -tables, -constraints, -ignoreCheckConstFilter, -skipCKConst, -skipFKConst, -skipColDefaultClause, -indexes, -triggers, -allViews, -views, -allSequences, -sequences, -allProcs, -procs, -allFuncs, -funcs, -checkFunctionBodies, -allPackages, -packages, -allDomains, -allQueues, -queues, -allRules,

次のページに続く

表 1 – 前のページからの続き

Feature	Relevant options
[Migration options](#migration-options)	-truncLoad, -enableConstBeforeDataLoad, -retryCount, -safeMode, -fastCopy, -analyze, vacuumAnalyze, -replaceNullChar, -copyDelimiter, -batchSize, -cpBatchSize, -lobBatchSize, -fetchSize, -filterProp -customColTypeMapping, -customColTypeMappingFile
[Oracle-specific options](#oracle-specific-options)	-allUsers, -users, -allProfiles, -profiles, -importPartitionAsTable, -objectTypes, -copyViaDBLinkOra, -allDBLinks -allSynonyms, -allPublicSynonyms, -allPrivateSynonyms, -useOraCase, -skipUserSchemaCreation
[Miscellaneous options](#miscellaneous-options)	-help, -logDir, -logFileCount, -logFileSize, -logBadSQL -verbose, -version
[Parallel data loading options](#parallel-loading-options)	-loaderCount, -parallelLoadRowLimit, -tableLoaderLimit

</ div>

7.1 オフライン移行オプション

コマンドラインで**offlineMigration** オプションを指定すると、Migration Toolkit はオフライン移行を実行します。オフライン移行中、Migration Toolkit は選択された各オブジェクトの定義を読み取り、後で実行されるときに Postgres の各オブジェクトを複製する SQL スクリプトを作成します。

!!! Note * 次の例は、Linux で Migration Toolkit を呼び出します。Windows で Migration Toolkit を呼び出すには、**runMTK.sh** コマンドの代わりに **runMTK.bat** コマンドを使用します。

スキーマとデータの両方のオフライン移行を実行するには、**offlineMigration** キーワードに続けてスキーマ名前を指定します。

```
$ ./runMTK.sh -offlineMigration <schema_name>
```

各データベースオブジェクト定義は、ホームフォルダー内のスキーマ名前とオブジェクトタイプから派生した名前を持つ個別のファイルに保存されます。代替ファイルの宛先を指定するには、**offlineMigration** オプションの後にディレクトリ名前を含めます。

```
$ ./runMTK.sh -offlineMigration <file_dest> <schema_name>
```

スキーマオブジェクトのみのオフライン移行（空のテーブルの作成）を実行するには、Migration Toolkit を呼び出すときに- `offlineMigration` キーワードに加えて- `schemaOnly` キーワードを指定します。

```
$ ./runMTK.sh -offlineMigration -schemaOnly <schema_name>
```

スキーマオブジェクト定義を省略してデータのみをオフライン移行を実行するには、Migration Toolkit を呼び出すときに- `dataOnly` キーワードと- `offlineMigration` キーワードを指定します。

```
$ ./runMTK.sh -offlineMigration -dataOnly <schema_name>
```

デフォルトでは、データは COPY フォーマットで書き込まれます。データをプレーンな SQL フォーマットで書き込むには、- `safeMode` キーワードを含めます。

```
$ ./runMTK.sh -offlineMigration -dataOnly -safeMode <schema_name>
```

デフォルトでは、テーブルデータを含むオフライン移行を実行すると、テーブルごとに個別のファイルが作成されます。マルチプルのテーブルからのデータを含む単一のファイルを作成するには、- `singleDataFile` キーワードを指定します。

```
./runMTK.sh -offlineMigration -dataOnly -singleDataFile -safeMode <schema_name>
```

!!! Note * -`singleDataFile` オプションは、プレーン SQL フォーマットのデータを移行する場合にのみ使用できます。- `singleDataFile` オプションを含める場合は、-`safeMode` キーワードを含める必要があります。

</div>

7.1.1 オフライン移行スクリプトの実行

edb-psql または psql コマンドラインを使用して、オフライン移行中に生成されたスクリプトを実行します。次の例では、スキーマ（hr 名前付け）を EDB Postgres Advanced Server に保存されている新しいデータベース（acctg 名前付け）に復元する方法について説明します。

1. `createdb` コマンドを使用して acctg データベースを作成し、そこに移行したデータベースオブジェクトを復元します。

- `createdb -U enterprisedb acctg`

2. edb-psql を使用して新しいデータベースに接続します。

- `edb-psql -U enterprisedb acctg`

3. \i メタコマンドを使用して、オブジェクト定義を作成する移行スクリプトを呼び出します。

- `acctg=# \i ./mtk_hr_ddl.sql`

4. -`offlineMigration` コマンドに- `singleDataFile` キーワードが含まれていた場合、`mtk_hr_data.sql` スクリプトには、新しいターゲットデータベース内のすべてのオブジェクトを再作成するために必要なコマンドが含まれます。次のコマンドでデータベースを作成します。

- `acctg=# \i ./mtk_hr_data.sql`

</div>

7.2 インポートオプション

デフォルトでは、Migration Toolkit はソースデータベースがオラクル、ターゲットデータベースが EDB Postgres Advanced Server であると想定します。- `sourcedbtype` および -`targetdbtype` キーワードを含めて、デフォルト以外のソースまたはターゲットデータベースを指定します。

デフォルトでは、Migration Toolkit はスキーマの移行時にデータとオブジェクト定義の両方をインポートします。または、データまたはオブジェクト定義のいずれかをインポートすることもできます。

-`sourcedbtype` <source_type>

-`sourcedbtype` オプションは、ソースデータベースタイプを指定します。`source_type` には、`mysql`、`oracle`、`sqlserver`、`sybase`、`postgresql`、または `enterprisedb` のいずれかの値を使用します。`source_type` は大文字と小文字を区別しません。デフォルトでは、`source_type` は `oracle` です。

-`targetdbtype` <target_type>

-`targetdbtype` オプションは、ターゲットデータベースタイプを指定します。`target_type` の場合は、`enterprisedb`、`postgres`、または `postgresql` のいずれかの値を使用します。`target_type` は大文字と小文字を区別しません。デフォルトでは、`target_type` は `enterprisedb` です。

-`schemaOnly`

このオプションは、スキーマ定義をインポートし、選択したすべてのスキーマオブジェクトをターゲットデータベースに作成します。このオプションを- `dataOnly` オプションと併用することはできません。

-`dataOnly`

このオプションはデータのみをコピーします。-`tables` オプションと併用すると、Migration Toolkit は選択したテーブルのデータのみをインポートします。このオプションを-`schemaOnly` オプションと併用することはできません。

</div>

7.3 スキーマ作成オプション

デフォルトでは、Migration Toolkit はソーススキーマオブジェクトまたはデータを同じ名前のスキーマにインポートします。ターゲットスキーマが存在しない場合、Migration Toolkit は新しいスキーマを作成します。または、- `targetSchema option` を使用してカスタムスキーマ名前を指定できます。次のオプションを使用して、既存のスキーマを削除し、新しいスキーマを作成することを選択できます。

-`dropSchema` [true|false]

このオプションを `true` に設定すると、Migration Toolkit は既存のスキーマとそのスキーマ内のオブジェクトをすべて削除し、新しいスキーマを作成します。デフォルトでは、-`dropSchema` は `false` です。

-`targetSchema` <schema_name>

-`targetSchema` オプションを使用して、移行されたスキーマの名前を指定します。マルチプルのスキーマを移行する場合は、各スキーマの名前を、間にスペース文字を入れずにコンマ区切りリストで指定します。-`targetSchema` オプションを使用しない場合、新しいスキーマの名前はソーススキーマの名前と同じです。

ターゲットスキーマ名として `information-schema`、`dbo`、`sys`、または `pg_catalog` を指定することはできません。これらのスキーマ名は、EDB Postgres Advanced Server のメタデータストレージ用に予約されています。

</div>

7.4 スキーマオブジェクト選択オプション

次のオプションを使用して、移行する特定のスキーマオブジェクトを選択します。

-allTables

ソーススキーマからすべてのテーブルをインポートします。

-tables <table_list>

選択したテーブルをソーススキーマからインポートします。`table_list` は、スペース文字を含まないコンマ区切りのテーブル名のリストです (`-tables emp, dept, acctg` など)。

-constraints

テーブルコンストレインをインポートします。このオプションは、スキーマ全体をインポートするとき、または `-allTables` または `-tables <table_list>` オプションを指定するときのみ有効です。

-ignoreCheckConstFilter

デフォルトでは、Migration Toolkit はチェックコンストレインとデフォルト句の Sybase データベースからの移行を実装していません。`-constraints` パラメータを指定して Sybase データベースからコンストレインとデフォルト句を移行する場合は、`- ignoreCheckConstFilter` パラメータを含めます。

-skipCKConst

チェックコンストレインの移行を省略します。このオプションは、ターゲットデータベースでサポートされていないソースデータベースのビルトイン関数に基づくチェックコンストレインを移行するときに役立ちます。

このオプションは、スキーマ全体をインポートする場合、または `-allTables` または `-tables <table_list>` オプションが指定されている場合にのみ有効です。

-skipFKConst

外部キーコンストレインの移行を省略します。このオプションは、スキーマ全体をインポートする場合、または `-allTables` または `-tables <table_list>` オプションが指定されている場合のみ有効です。

-skipColDefaultClause

列の `DEFAULT` 句の移行を省略します。

-indexes

テーブルインデックスをインポートします。このオプションは、スキーマ全体をインポートするとき、または `-allTables` または `-tables <table_list>` オプションが指定されているときに有効です。

-triggers

テーブルトリガーをインポートします。このオプションは、スキーマ全体をインポートするとき、または `allTables` または `-tables <table_list>` オプションが指定されているときに有効です。

`-allViews`

ソーススキーマからビューをインポートします。このオプションは、ソースから動的ビューとマテリアライズドビューを移行します。オラクルおよび Postgres のマテリアライズドビューがサポートされています。

`-views <view_list>`

指定されたマテリアライズドビューまたは動的ビューをソーススキーマからインポートします。オラクルおよび Postgres のマテリアライズドビューがサポートされています。`view_list` は、間にスペース文字のないビュー名のコンマ区切りリストです（例：`-views all_emp, mgmt_list, acct_list`）。

`-allSequences`

ソーススキーマからすべてのシーケンスをインポートします。

`-sequences <sequence_list>`

選択したシーケンスをソーススキーマからインポートします。`<sequence_list>` は、間にスペース文字を入れないシーケンス名のコンマ区切りリストです。

`-allProcs`

ソーススキーマからすべてのストアードプロシージャをインポートします。

`-procs <procedures_list>`

選択したストアードプロシージャをソーススキーマからインポートします。`procedures_list` は、間にスペース文字を入れないプロシージャ名のコンマ区切りリストです。

`-allFuncs`

ソーススキーマからすべての関数をインポートします。

`-funcs <function_list>`

選択した関数をソーススキーマからインポートします。`function_list` は、間にスペース文字を入れないファンクション名のコンマ区切りリストです。

`-checkFunctionBodies [true/false]`

`false` の場合、ファンクションの作成中にファンクション本体のバリデーションを無効にします。このバリデーションを無効にすると、ファンクションに前方参照が含まれている場合のエラーを回避できます。デフォルト値は `true` です。

`-allPackages`

ソーススキーマからすべてのパッケージソフトをインポートします。

`-packages <package_list>`

選択したパッケージソフトをソーススキーマからインポートします。`package_list` は、間にスペース文字を入れないパッケージ名のコンマ区切りリストです。

`-allDomains`

すべてのドメイン、列挙、および複合型をソースデータベースからインポートします。このオプションは、ソースとターゲットの両方が Postgres ホストに保存されている場合にのみ有効です。

-allQueues

ソーススキーマからすべてのキューをインポートします。これらは、DBMS_AQ および DBMS_AQADM ビルトインパッケージソフトによって作成および管理されるキューです。オラクルがソースデータベースの場合、**-objectTypes** オプションも指定する必要があります。EDB Postgres Advanced Server がソースデータベースの場合、**-allDomains** および **-allTables** オプションも指定する必要があります。オラクルおよび EDB Postgres Advanced Server キューがサポートされています。

-queues <queue_list>

選択したキューをソーススキーマからインポートします。**queue_list** は、間にスペース文字を入れないキュー名のコンマ区切りリストです。これらは、DBMS_AQ および DBMS_AQADM ビルトインパッケージソフトによって作成および管理されるキューです。オラクルがソースデータベースの場合、**-objectTypes** も指定する必要があります。EDB Postgres Advanced Server がソースデータベースの場合、**-allDomains** および **-allTables** も指定する必要があります。オラクルおよび EDB Postgres Advanced Server キューがサポートされています。

-allRules

ソースデータベースからすべてのルールをインポートします。このオプションは、ソースとターゲットの両方が Postgres ホストに保存されている場合にのみ有効です。

</div>

7.5 移行オプション

移行オプションを使用して、移行プロセスの詳細を制御します。

-truncLoad

新しいデータをインポートする前に、テーブルからデータを切り捨てます。このオプションは、**-dataOnly** オプションとともに使用します。

-enableConstBeforeDataLoad

パーティション化されていないソーステーブルがパーティション化されたテーブルにマップされる場合は、**-enableConstBeforeDataLoad** オプションを含めます。このオプションは、データ移行前に、個々のパーティションにデータをリダイレクトするトリガーを含む、ターゲットテーブルのすべてのトリガーを有効にします。**-enableConstBeforeDataLoad** は、**-truncLoad** パラメータも指定した場合にのみ有効です。

-retryCount [<value>]

複数スキーマの移行を実行している場合、スキーマ間の依存関係が原因で最初の移行試行中に移行に失敗したオブジェクトは、後の移行中に正常に移行する可能性があります。**-retryCount** オプションを使用して、Migration Toolkit が最初の移行試行中に make したオブジェクトの移行を試行する回数を指定します。0 より大きい値を指定します。デフォルト値は 2 です。

-safeMode

`-safeMode` オプションを含めると、Migration Toolkit は移行された行をコミットします。移行がすべてのレコードの転送に失敗した場合、障害点の前に挿入された行はターゲットデータベースに残ります。

`-fastCopy`

`-fastCopy` オプションを含めると、Migration Toolkit が WAL ロギングをバイパスして、最適化された方法で COPY オペレーションを実行するように指定されます。デフォルトでは無効になっています。`-fastCopy` オプションの使用を選択した場合、移行が中断された場合、ターゲットデータベース内の移行されたデータを回復できない可能性があります。

`-replaceNullChar <value>`

Migration Toolkit は、値が NULL の列のインポートをサポートしています。ただし、Migration Toolkit は、JDBC 接続プロトコルでの NULL 文字値（埋め込みバイナリゼロ 0x00）のインポートをサポートしていません。NULL 文字を含むデータをインポートする場合は、`-replaceNullChar` オプションを使用して、NULL 文字を単一の非 NULL 置換文字に置き換えます。置換文字を引用符またはアポストロフィで囲まないでください。

データを移行したら、SQL ステートメントを使用して、`-replaceNullChar` で指定された文字をバイナリのゼロに置き換えます。

`-analyze`

`-analyze` オプションを含めて、ターゲットデータベースに対して Postgres `ANALYZE` オペレーションを呼び出します。オプティマイザは、効率的なクエリープランを構築するための情報を使用して、`ANALYZE` オペレーションによって収集された統計を調べます。

`-vacuumAnalyze`

`-vacuumAnalyze` オプションを含めて、ターゲットデータベースに対して `VACUUM` 操作と `ANALYZE` 操作の両方を呼び出します。オプティマイザは、効率的なクエリープランを構築するための情報を使用して、`ANALYZE` オペレーションによって収集された統計を調べます。`VACUUM` オペレーションは、ターゲットデータベース内の無効なタプルによって占有されているストレージスペースを再利用します。

`-copyDelimiter`

テーブルデータをローディングするときに `COPY` コマンドでデリミタとして使用する単一の文字を指定します。デフォルト値は `'\t'`（タブ）です。

`-batchSize`

一括挿入のバッチサイズを指定します。有効な値は 1～1000 です。デフォルトのバッチサイズは 1000 です。メモリ不足例外が発生する場合は、`-batchSize` の値を減らします。

`-cpBatchSize`

`COPY` コマンドで使用するバッチサイズをメガバイト単位で指定します。0 より大きい値はすべて有効です。デフォルトのバッチサイズは 8 メガバイトです。

`-lobBatchSize`

LOB データ型のバッチでロードする行数を指定します。`BYTEA`、`BLOB`、`CLOB` などのラージオブジェクトタイプ（LOB）列を含むテーブルのデータ移行は、デフォルトで一度に 1 行ずつ実行されデフォルト。これは、個々の LOB 列が数百メガバイトのデータを保持している場合に、ヒープ領域不足エラーを回避するた

めです。LOB 列の平均データサイズが下限にある場合、各バッチの行数を 0 より大きい値で指定することにより、LOB バッチサイズをカスタマイズします。

-fetchSize

-fetchSize オプションを使用して、結果セットでフェッチされる行の数を指定します。指定された **-fetchSize** がラージすぎる場合、メモリ不足例外が発生する可能性があります。ラージテーブルを移行するときにこの落とし穴を避けるために、**-fetchSize** オプションを含めます。デフォルトのフェッチサイズは JDBC ドライバの実装に固有であり、データベースによって異なります。

MySQL ユーザーのノート：デフォルトでは、MySQL JDBC ドライバは 1 回のネットワークラウンドトリップでテーブル内のすべての行を Migration Toolkit にフェッチします。この動作は、ラージテーブルで使用可能なメモリを簡単に超える可能性があります。ヒープ領域不足エラーが発生した場合は、コマンドライン引数として **-fetchSize 1** を指定して、Migration Toolkit にテーブルデータを一度に 1 行ずつロードさせます。

-filterProp <file_name>

<file_name> は、キー=値のペアでコンストレインを含むファイルの名前を指定します。データベースから読み取られた各レコードは、コンストレインに対して評価されます。コンストレインを満たすものは移行されます。ペアの左側には、テーブル名前がリストされます。テーブル名前はスキーマ修飾されていません。右側は、移行される行に対して真でなければならない条件を指定します。例、プロパティファイルに次のコンストレインを含める

```
countries=country_id<>'AR'
```

country_id の値が **AR** と等しくない国のみを移行します。この制約は、国テーブルに適用されます。

マルチプルのテーブルの条件を指定することもできます。ただし、各テーブルの条件は、プロパティファイルの新しい行にある必要があります。

例：

プロパティファイルの次のエントリは、EMPLOYEES および DEPARTMENTS テーブルから関連データのみを移行します。

```
EMPLOYEES=(LAST_NAME IN ('Grant','Weiss') AND PHONE_NUMBER LIKE '650%')
```

```
DEPARTMENTS=(DEPARTMENT_ID BETWEEN 10 AND 30)
```

-customColTypeMapping <column_list>

カスタム型マッピングを使用して、移行された列のデータタイプを変更します。各ペアの左側は、正規表現で列を指定します。各ペアの右側には、その列が想定するデータタイプの名前が付けられます。**<column_list>** のセミコロン区切りリストにマルチプルのペアを含めることができます。例、名前が ID で終わる列をマップと入力するには、次のカスタムマッピングエントリーを使用します。

```
.*ID=INTEGER
```

列がテーブル修飾されていない限り、条件にマッチするすべてのテーブル列にカスタムマッピングが適用されます。

'\\' 文字はエスケープ文字列として機能します。'.' は正規表現の予約文字であるため、Linux では '\\.' を使用して '.' 文字を表します。例、カスタムマッピングを使用して EMP テーブルの EMP_ID 列から行を選択するには、次のカスタムマッピングエントリーを指定します。

```
EMP\.\.EMP_ID=INTEGER
```

Windows、'\'.' を使用して'\'.' 文字を表します。

```
EMP\.\.EMP_ID=INTEGER
```

```
-customColTypeMappingFile <property_file>
```

<property_file>にはマルチプルのカスタムタイプマッピングを含めることができます。ファイルの各エントリーを、キー=値のペアで別々の行に指定します。各ペアの左側は、正規表現で列を選択します。各ペアの右側には、その列が想定するデータタイプの名前が付けられます。

</div>

7.6 並列データローディングの移行オプション

比較的ラージのテーブルまたはラージサイズのテーブルを処理する場合、通常のデータ移行には長い時間がかかります。この問題の解決策として、Migration Toolkit には、スキーマレベルおよびテーブルレベルでデータを並列に移行できる並列データローディングアルゴリズムがあります。

スキーマレベルでの並列データ移行では、マルチプルのスレッドを使用して複数のテーブルを同時に移行します。テーブルレベルでの並列データローディングでは、マルチプルのスレッドを使用して、ソースデータベースからターゲットデータベースに単一のテーブルをより効率的かつ短時間で移行します。

!!! Note * -並列ローディングはラージテーブルにのみ役立ちます。小さなテーブルの場合、並列スレッドには時間がかかる場合があります。したがって、比較的ラージテーブルに対しては、テーブルレベルで並列データローディングを使用することをお勧めします。

- -並列データのローディングは、プライマリキー/一意キーのないテーブルには適用されません。
- -fastCopy パラメータは、並列データのローディングでは使用できません。
- -並列ローディング機能は、ソースオラクル、EDB Postgres Advanced Server、および PostgreSQL データベースにのみ適用されます。

次のオプションは、MTK がデータを並行して移行する方法を制御します。

```
-loaderCount [<value>]
```

-loaderCount オプションを使用して、プールで使用可能な並列スレッドの数を指定して、データロードを並列に実行します。このオプションは、Migration Toolkit を実行しているホストシステムにハイエンドの CPU および RAM リソースがある場合に特に役立ちます。value は非ゼロの正の数にすることができますが、CPU コアの数を超えないことをお勧めします。例、デュアルコア CPU の最適値は 2 です。デフォルトは 1 です。

!!! Note * マルチプルのスレッドを使用する場合、実際のスレッド数とテーブルデータサイズに応じて、Migration Toolkit のメモリヒープサイズを調整する必要があります。

```
-parallelLoadRowLimit [<value>]
```

-parallelLoadRowLimit オプションを使用して、テーブルレベルでデータロードを並行して実行するためにテーブルに必要な最小行数を指定します。<value>パラメータは 0 より大きいなければなりません。デフォルトは 100,000 です。

例、`-parallelLoadRowLimit` の値が 10,000 の場合、行数が 10,000 以上のテーブルのみがテーブルレベルで並列にロードされます。他のテーブルは、シーケンシャルモードまたはスキーマレベルで移行します。

`-tableLoaderLimit` [<value>]

`-tableLoaderLimit` オプションを使用して、スレッドプールから割り当てるスレッドの最大数を指定し、テーブルレベルで単一のテーブルデータを並列チャンクでロードします。<value>パラメータは、`-loaderCount` の値をより大きいとはなりません。デフォルト値は、`-loaderCount` オプションの値と同じです。このオプションは、`-loaderCount` のデフォルト以外のカスタム値に基づいて暗黙的に変更されます。

例、`-loaderCount` が 4 に設定され、`-tableLoaderLimit` が指定されていない場合、`-tableLoaderLimit` も 4 に設定されます。したがって、すべてのスレッドが単一のテーブルロードに使用されることを意図していない場合、`-tableLoaderLimit` をより低い値に減らします。

7.6.1 並列データ読み込みアルゴリズムの概要

アルゴリズムは 3 つのステップで機能します。

1. テーブルとテーブルチャンクのリストは、移行されるテーブルの順序付きリストから作成され、テーブル名前でソートされます。リストは、`-parallelLoadRowLimit` および `-tableLoaderLimit` オプションの値に基づいて作成されます。

1. `-loaderCount` オプション値に基づいてスレッドプールが作成されます。

1. 並列データ読み込みアルゴリズムは、テーブルおよびテーブルチャンクのリストから各テーブル/チャンクのタスクを作成し、スレッドプールで実行します。スレッド数と同じ数のタスクが並行して実行されます。タスクの実行が完了するとすぐに、そのスレッドはリストから次のタスクの実行を開始します。リストの最後のタスクが完了するとすぐに、並列データのローディングが完了します。

例：

このシナリオでは、4 つのテーブル `tab1`, `tab2`, `tab3`, `tab4` を並列データローディングで移行します。

```
# ./runMTK.sh -sourcedbtype enterprisedb -targetdbtype enterprisedb -loaderCount 4 -
  ↪parallelLoadRowLimit 10000 -tableLoaderLimit 2 -tables tab1,tab2,tab3,tab4 public
```

テーブル `tab1` には 50,000 行、テーブル `tab2` には 9,500 行、テーブル `tab3` には 50,001 行、テーブル `tab4` には 9,999 行があります。ステップ 1 のテーブルのリストは、`tab1`, `tab2`, `tab3` および `tab4` です。

テーブル `tab1` および `tab3` のみが、テーブルレベル (`-parallelLoadRowLimit 10000`) での並列ローディングの修飾となります。

`-tableLoaderLimit` オプションの値は 2 であるため、テーブル `tab1` は `tab1_chunk1` と `tab1_chunk2` の 2 つのチャンクでロードされます。各チャンクの行数は、`totalRowCount/tableLoaderLimit` に基づいて決定されます。この場合、両方のチャンクには 25,000 行 (50,000 / 2) があります。

テーブル `tab3` も、`tab3_chunk1` と `tab3_chunk2` の 2 つのチャンクでロードされます。テーブル `tab3` の行の総数が奇数 (50,001) であるため、最初のチャンクは追加の行を収容するため、`tab3_chunk1` には 25,001 行、`tab3_chunk2` には 25,000 行が含まれます。

作成されるテーブルとテーブルチャンクのリストは次のとおりです。

```
*   tab1_chunk1      (25,000 rows)
*   tab1_chunk2      (25,000 rows)
*   tab2              (9500 rows)
*   tab3_chunk1      (25,001 rows)
*   tab3_chunk2      (25,000 rows)
*   tab4              (9999 rows)
```

4つのスレッド T1, T2, T3, T4 を持つスレッドプールが作成されます。ステップ 1 で作成されたリストからテーブル/チャンクを移行するタスクは、スレッドプールで実行されます。

```
T1 → tab1_chunk1      (25,000 rows)
T2 → tab1_chunk2      (25,000 rows)
T3 → tab2              (9500 rows)
T4 → tab3_chunk1      (25,001 rows)
```

スレッド T3 は最初に終了し、リストの次のチャンクである `tab3_chunk2` の実行を開始します。

```
T1 → tab1_chunk1      (25,000 rows)
T2 → tab1_chunk2      (25,000 rows)
T3 → tab3_chunk2      (25,000 rows)
T4 → tab3_chunk1      (25,001 rows)
```

スレッド T1 は 2 番目に終了し、`tab4` の実行を開始します。

```
T1 → tab4              (9999 rows)
T2 → tab1_chunk2      (25,000 rows)
T3 → tab3_chunk2      (25,000 rows)
T4 → tab3_chunk1      (25,001 rows)
```

このように、4つのスレッドを使用して、スキーマとテーブルの混在テーブルレベルで4つのテーブルを並行して移行します。コマンドオプションを変更して `-tableLoaderLimit` を 1 に設定する場合：

```
# ./runMTK.sh -sourcedbtype enterprisedb -targetdbtype enterprisedb -loaderCount 4 -
→parallelLoadRowLimit 10000 -tableLoaderLimit 1 -tables tab1,tab2,tab3,tab4 public
```

純粋なスキーマレベルの並列処理が可能です。各テーブルは1つのスレッドで同時に処理されます。

```
T1 → tab1      (50,000 rows)
T2 → tab2      (9500 rows)
T3 → tab3      (50,001 rows)
T4 → tab4      (9999 rows)
```

7.6.2 システムリソースの推奨事項とコンストレイン**

MTK を実行しているホストシステムで使用可能な CPU および RAM リソースに応じて、スレッドの数を選択します。CPU、IO、およびディスク使用率に関してソースデータベースに特定のオーバーヘッドがあるため、特定のしきい値を超える並列スレッド数を超えると、MTK のパフォーマンスが低下します。

</div>

7.7 Oracle 固有のオプション

次のオプションは、ソースデータベースがオラクルの場合にのみ適用されます。

-objectTypes

runMTK.sh コマンドの最後に指定されたスキーマリストからユーザー定義のオブジェクトタイプをインポートします。

-allUsers

ソースデータベースからすべてのユーザーとロールをインポートします。- allUsers オプションは、オラクルデータベースから EDB Postgres Advanced Server データベースに移行する場合にのみサポートされます。

-users <user_list>

選択したユーザーまたはロールをソースオラクルデータベースからインポートします。<user_list>は、スペース文字を挟まないユーザ/ロール名のコンマ区切りリストです (-users MTK, SAMPLE, acctg など)。-users オプションは、オラクルデータベースから EDB Postgres Advanced Server データベースに移行する場合にのみサポートされます。

-allProfiles

ソースデータベースからすべてのカスタム（つまり、ユーザー作成）プロファイルをインポートします。DEFAULT や MONITORING_PROFILE などの他のオラクル非カスタムプロファイルはインポートされません。

インポートされたプロファイルに関連付けられている次のパスワードパラメータのみがインポートされます。

FAILED_LOGIN_ATTEMPTS

PASSWORD_LIFE_TIME

PASSWORD_REUSE_TIME

PASSWORD_REUSE_MAX

PASSWORD_LOCK_TIME

PASSWORD_GRACE_TIME

PASSWORD_VERIFY_FUNCTION

オラクルリソースパラメータなど、他のすべてのプロファイルパラメータはインポートされません。SRC_DB_USER で指定されたオラクルデータベースユーザには、オラクルデータディクショナリビュー DBA_PROFILES に対する SELECT 権限が必要です。

!!! Note * - allProfiles オプションは、オラクルデータベースから EDB Postgres Advanced Server データベースに移行する場合にのみサポートされます。

-profiles <profile_list>

選択したカスタム（つまり、ユーザー作成）プロファイルをソースオラクルデータベースからインポートします。profile_list は、スペース文字を含まないコンマ区切りのプロファイル名のリストです（-profiles ADMIN_PROFILE,USER_PROFILE など）。DEFAULT や MONITORING_PROFILE などのオラクル非カスタムプロファイルはインポートされません。

-allProfiles オプションと同様に、パスワードパラメータのみがインポートされます。SRC_DB_USER で指定されたオラクルデータベースユーザには、オラクルデータディクショナリビュー DBA_PROFILES に対する SELECT 権限が必要です。

!!! Note * -profiles オプションは、オラクルデータベースから EDB Postgres Advanced Server データベースに移行する場合にのみサポートされます。

-importPartitionAsTable <table_list>

-importPartitionAsTable パラメータを含めて、オラクルホスト上にあるパーティションテーブルの内容を単一の非パーティションテーブルにインポートします。table_list は、間にスペース文字が入っていないテーブル名のコンマ区切りリストです（-importPartitionAsTable emp,dept,acctg など）。

-copyViaDBLinkOra

dblink_ora モジュールは、SQL レベルで EDB Postgres Advanced Server から Oracle への接続を提供します。dblink_ora はバンドルされ、EDB Postgres Advanced Server データベースインストールのパートとしてインストールされます。dblink_ora は、COPY API メソッドを使用してデータベース間でデータを転送します。このメソッドは、JDBC COPY メソッドよりかなり高速です。

この例では、dblink_ora COPY API を使用して、HR スキーマからすべてのテーブルを移行します。

```
./runMTK.sh -copyViaDBLinkOra -allTables HR
```

ターゲット EDB Postgres Advanced Server データベースに dblink_ora がインストールおよび構成されている必要があります。dblink_ora については、[Database Compatibility for Oracle Developers](#) を参照してください。

-allDBLinks [link_Name_1=password_1,link_Name_2=password_2,...]

オラクルデータベースリンクを移行するには、このオプションを選択します。ソースデータベース内の各リンク接続のパスワード情報は暗号化されているため、指定しない限り、ダミーパスワード edb が代わりに使用されます。

接続ユーザのダミーパスワードとして edb を使用してすべてのデータベースリンクを移行するには：

```
./runMTK.sh -allDBLinks HR
```

または、スペース文字を介在させずに、名前=値のペアのコンマ区切りリストを使用して、各データベースリンクのパスワードを指定することもできます。ペアの左側にリンク名前を、右側にパスワード値を指定します。

すべてのデータベースリンクをコマンドラインで指定された実際のパスワードで移行するには：

```
$. /runMTK.sh -allDBLinks LINK_NAME1=abc, LINK_NAME2=xyz HR
```

Migration Toolkit は、EnterpriseDB で現在サポートされているデータベースリンクタイプのみを移行します。これには、パブリックおよびプライベートタイプの固定ユーザリンクが含まれます。

-allSynonyms

すべてのパブリックシノニムとプライベートシノニムをオラクルデータベースから EDB Postgres Advanced Server データベースに移行するには、**-allSynonyms** オプションを含めます。同じ名前の同義語がターゲットデータベースに既に存在する場合、既存の同義語は移行されたバージョンに置き換えられます。

-allPublicSynonyms

すべてのパブリックシノニムをオラクルデータベースから EDB Postgres Advanced Server データベースに移行するには、**-allPublicSynonyms** オプションを含めます。同じ名前の同義語がターゲットデータベースに既に存在する場合、既存の同義語は移行されたバージョンに置き換えられます。

-allPrivateSynonyms

すべてのプライベートシノニムをオラクルデータベースから EDB Postgres Advanced Server データベースに移行するには、**-allPrivateSynonyms** オプションを含めます。同じ名前の同義語がターゲットデータベースに既に存在する場合、既存の同義語は移行されたバージョンに置き換えられます。

-useOraCase

-useOraCase オプションを含めて、オラクルデータベースから EDB Postgres Advanced Server データベースに移行するときに、すべてのデータベースオブジェクトのオラクルのデフォルトの大文字の命名規則を保持します。

大文字の命名規則は、テーブル、ビュー、シーケンス、プロシージャ、関数、トリガー、パッケージソフトなどで保持されます。これらのデータベースオブジェクトでは、大文字の命名規則が適用されます：-データベースオブジェクトの名前-テーブルおよびビューの列名、キー名、インデックス名、制約名など-aの SELECT 列リストビュープロシージャまたはファンクションヘッダーのパートであるパラメータ名

!!! Note * プロシージャ、ファンクション、trigger または package の手続きコード本体では、エラーなしで実行プログラムの識別子参照を手動で編集する必要がある場合があります。このような修正は、発生した可能性のある識別子参照の適切な大文字小文字変換に関するものです。

!!! Note * **-useOraCase** オプションを指定する場合、**-skipUserSchemaCreation** オプションも指定する必要があります。詳細については、**-skipUserSchemaCreation** オプションの説明を参照してください。

-useOraCase オプションを使用しない Migration Toolkit のデフォルトの動作では、データベース引用符オラクルは引用符で囲まらずにオラクルから抽出されます。以下は、**-offlineMigration** オプションを使用して Migration Toolkit によって生成された TABLE コマンドの一部です。

```
CREATE TABLE DEPT (
  *   DEPTNO NUMBER(2) NOT NULL,
  *   DNAME VARCHAR2(14),
  *   LOC VARCHAR2(13)
);
```

(次のページに続く)

(前のページからの続き)

```
ALTER TABLE DEPT ADD CONSTRAINT DEPT_PK PRIMARY KEY (DEPTNO);
ALTER TABLE DEPT ADD CONSTRAINT DEPT_DNAME_UQ UNIQUE (DNAME);
```

その後、このテーブルを移行して EDB Postgres Advanced Server で作成すると、引用されていないオブジェクト名はすべて小文字に変換されるため、テーブルは次のように EDB Postgres Advanced Server に表示されます。

```
Table "edb.dept"
Column | Type                | Modifiers
-----+-----+-----
deptno | numeric(2,0)         | not null
dname  | character varying(14) |
loc    | character varying(13) |
Indexes:
* "dept_pk" PRIMARY KEY, btree (deptno)
* "dept_dname_uq" UNIQUE CONSTRAINT, btree (dname)
```

EDB Postgres Advanced Server アプリケーションが、引用符で囲まれた大文字の識別子を使用して移行されたデータベースオブジェクトを参照している場合、データベースオブジェクト名は小文字であるため、アプリケーションは失敗します。

```
usepostcase=# SELECT * FROM "DEPT";
ERROR: relation "DEPT" does not exist
LINE 1: SELECT * FROM "DEPT";
```

アプリケーションが引用符付きの大文字の識別子を使用する場合は、`-useOraCase` オプションを使用して移行を実行します。DDL は、すべてのデータベースオブジェクト名を引用符で囲みます。

```
CREATE TABLE "DEPT" (
*   "DEPTNO" NUMBER(2) NOT NULL,
*   "DNAME" VARCHAR2(14),
*   "LOC" VARCHAR2(13)
);
ALTER TABLE "DEPT" ADD CONSTRAINT "DEPT_PK" PRIMARY KEY ("DEPTNO");
ALTER TABLE "DEPT" ADD CONSTRAINT "DEPT_DNAME_UQ" UNIQUE ("DNAME");
```

その後、このテーブルを移行して EDB Postgres Advanced Server で作成すると、すべてのオブジェクト名が大文字で維持されるため、テーブルは次のように EDB Postgres Advanced Server に表示されます。

```
Table "EDB.DEPT"
Column | Type                | Modifiers
-----+-----+-----
DEPTNO | numeric(2,0)         | not null
DNAME  | character varying(14) |
LOC    | character varying(13) |
Indexes:
```

(次のページに続く)

(前のページからの続き)

```
* "DEPT_PK" PRIMARY KEY, btree ("DEPTNO")
* "DEPT_DNAME_UQ" UNIQUE CONSTRAINT, btree ("DNAME")
```

アプリケーションは、引用符で囲まれた大文字の名前を使用してオブジェクトにアクセスできます。

```
useoracase=# SELECT * FROM "DEPT";
DEPTNO | DNAME          | LOC
-----+-----+-----
10      | ACCOUNTING     | NEW YORK
20      | RESEARCH       | DALLAS
30      | SALES          | CHICAGO
40      | OPERATIONS     | BOSTON
(4 rows)
```

-skipUserSchemaCreation

オラクルユーザが移行されると、そのロールがまだ存在しない場合、オラクルユーザのターゲットデータベースサーバにロール（ユーザ名前）が作成されます。ロール名前は小文字で作成されます。新しいロールが作成されると、同じ名前のスキーマも小文字で作成されます。

-skipUserSchemaCreation オプションを指定すると、移行されたオラクルユーザ名前の自動スキーマ作成が防止されます。このオプションは、**-useOraCase** オプションを指定して、大文字と小文字が異なるだけで同じ名前の2つのスキーマを作成しないようにする場合に特に便利です。**-useOraCase** オプションを指定すると、Migration Toolkit の起動時にオプションリストに従って指定されたソーススキーマの大文字のオラクル命名規則でスキーマが作成されます。

したがって、**-skipUserSchemaCreation** オプションを指定せずに**-useOraCase** オプションを指定すると、ターゲットデータベースには、一方が小文字でもう一方が大文字の2つの同じ名前付けのスキーマが作成されます。**-useOraCase** オプションが**-skipUserSchemaCreation** オプションとともに指定されている場合、ターゲットデータベースには大文字のスキーマのみが含まれます。

</div>

7.8 その他のオプション

これらの移行オプションを使用すると、Migration Toolkit のヘルプとバージョン情報がビューされます。これらのオプションを使用して、Migration Toolkit のフィードバックとロギングオプションを制御することもできます。

-help

アプリケーションのコマンドラインの使用情報を表示します。

-logDir <log_path>

ログファイルを書き込む場所を指定するには、このオプションを含めます。**<log_path>**は、ログファイルを保存する場所です。デフォルトでは、Linux ではログファイルは次の場所書き込まれます。

`$HOME/.enterprisedb/migration-toolkit/logs`

Windows では、ログファイルは次の場所に保存されます。

```
%HOMEDRIVE%%HOMEPATH%\enterprisedb\migration-toolkit\logs
```

```
-logFileCount <file_count>
```

ログファイルのローテーションで使用するファイルの数を指定するには、このオプションを含めます。0 の値を指定して、ログファイルのローテーションを無効にし、単一のログファイルを作成します。このファイルは、logFileSize オプションを使用して指定された値に達すると切り捨てられます。<file_count>は 0 以上でなければなりません。デフォルトは 20 です。

```
-logFileSize <file_size>
```

新しいログファイルにローテーションする前に、メガバイト単位で最大ファイルサイズ制限を指定するには、このオプションを含めます。file_size は 0 より大きいなければなりません。デフォルトは 50 です。

```
-logBadSQL
```

失敗したオブジェクトのスキーマ定義（DDL スクリプト）をファイルに保存するには、このオプションを含めます。ファイルはエラーログに使用されたのと同じパスの下に保存され、次のフォーマットで名前が名前付けます。

```
mtk_bad_sql_<schema_name_timestamp>.sql
```

ここで、schema_name はスキーマの名前であり、timestamp は Migration Toolkit 実行のタイムスタンプです。

```
-verbose [on|off]
```

標準出力にアプリケーションログメッセージを表示します。デフォルトでは、詳細表示はオンになっています。

```
-version
```

Migration Toolkit バージョンを表示します。

</ div>

7.9 例

この例では、オラクルから EDB Postgres Advanced Server への移行を実行します。

以下は、toolkit.properties ファイルのコンテンツです。

```
SRC_DB_URL=jdbc:oracle:thin:@192.168.2.6:1521:xe
SRC_DB_USER=edb
SRC_DB_PASSWORD=password

TARGET_DB_URL=jdbc:edb://localhost:5444/edb
TARGET_DB_USER=enterprisedb
TARGET_DB_PASSWORD=password
```

次のコマンドは、Migration Toolkit を呼び出します。

```
$ ./runMTK.sh EDB
```

```
Running EnterpriseDB Migration Toolkit (Build 48.0.0) ...
Source database connectivity info...
conn =jdbc:oracle:thin:@192.168.2.6:1521:xe
user =edb
password=*****\*
Target database connectivity info...
conn =jdbc:edb://localhost:5444/edb
user =enterprisedb
password=*****\*
Connecting with source Oracle database server...
Connected to Oracle, version 'Oracle Database 10g Express Edition
Release 10.2.0.1.0 - Production'
Connecting with target EnterpriseDB database server...
Connected to EnterpriseDB, version '9.4.0.0'
Importing redwood schema EDB...
Creating Schema...edb
Creating Sequence: NEXT_EMPNO
Creating Tables...
Creating Table: BAD_TABLE
MTK-15013: Error Creating Table BAD_TABLE
DB-42704: ERROR: type "binary_double" does not exist at position 58
-- CREATE TABLE BAD_TABLE (
-- F1 NUMBER NOT NULL,
-- Line 3: F2 BINARY_DOUBLE
-- ^
Creating Table: DEPT
Creating Table: EMP
Creating Table: JOBHIST
Creating Table: "MixedCase"
Creating Table: "lowercase"
Created 5 tables.
Loading Table Data in 8 MB batches...
Loading Table: DEPT ...
[DEPT] Migrated 4 rows.
[DEPT] Table Data Load Summary: Total Time(s): 0.147 Total Rows: 4
Loading Table: EMP ...
[EMP] Migrated 14 rows.
[EMP] Table Data Load Summary: Total Time(s): 0.077 Total Rows: 14
Loading Table: JOBHIST ...
[JOBHIST] Migrated 17 rows.
[JOBHIST] Table Data Load Summary: Total Time(s): 0.042 Total Rows: 17
Total Size(MB): 9.765625E-4
Loading Table: "MixedCase" ...
```

(次のページに続く)

(前のページからの続き)

```

["MixedCase"] Table Data Load Summary: Total Time(s): 0.098 Total Rows:0
Loading Table: "lowercase" ...
["lowercase"] Table Data Load Summary: Total Time(s): 0.066 Total Rows:0
Data Load Summary: Total Time (sec): 0.806 Total Rows: 35 Total
Size(MB): 0.001
Creating Constraint: DEPT_PK
Creating Constraint: DEPT_DNAME_UQ
Creating Constraint: EMP_PK
Creating Constraint: JOBHIST_PK
Creating Constraint: SYS_C008958
MTK-15001: Error Creating Constraint SYS_C008958
DB-42P01: com.edb.util.PSQLException: ERROR: relation "bad_table" does
not exist
Creating Constraint: EMP_REF_DEPT_FK
Creating Constraint: EMP_SAL_CHK
Creating Constraint: JOBHIST_REF_DEPT_FK
Creating Constraint: JOBHIST_REF_EMP_FK
Creating Constraint: JOBHIST_DATE_CHK
Creating Trigger: USER_AUDIT_TRIG
Creating Trigger: EMP_SAL_TRIG
MTK-13009:Warning! Skipping migration of trigger DROP_TRIGGER, currently
non-table triggers are not supported in target database.
Creating View: SALESEMP
Creating Function: EMP_COMP
Creating Package: EMP_ADMIN
MTK-16005:Package Body is Invalid, Skipping...
Schema EDB imported with errors.
MTK-12001: The user/role migration failed due to insufficient
privileges.
Grant the user SELECT privilege on the following Oracle catalogs:
DBA_ROLES
DBA_USERS
DBA_TAB_PRIVS
DBA_PROFILES
DBA_ROLE_PRIVS
ROLE_ROLE_PRIVS
DBA_SYS_PRIVS
One or more schema objects could not be imported during the migration
process. Please review the migration output for more details.
Migration logs have been saved to
/home/user/.enterprisedb/migration-toolkit/logs
***** Migration Summary *****
Sequences: 1 out of 1
Tables: 5 out of 6
Constraints: 9 out of 10

```

(次のページに続く)

(前のページからの続き)

```

Triggers: 2 out of 3 (skipped 1)
Views: 1 out of 1
Functions: 1 out of 1
Packages: 1 out of 1
Total objects: 30
Successful count: 20
Failed count: 2
Skipped count: 1
Invalid count: 7
List of failed objects
=====

```

Tables

```

-----
1. EDB.BAD_TABLE

```

Constraints

```

-----
1. EDB.BAD_TABLE.SYS_C008958

```

List of invalid objects

```

=====
1. EDB.HIRE_CLERK (FUNCTION)
2. EDB.NEW_EMPNO (FUNCTION)
3. EDB.EMP_ADMIN (PACKAGE BODY)
4. EDB.EMP_QUERY (PROCEDURE)
5. EDB.EMP_QUERY_CALLER (PROCEDURE)
6. EDB.LIST_EMP (PROCEDURE)
7. EDB.SELECT_EMP (PROCEDURE)
*****

```

スキップされ、サポートされていないデータベースオブジェクトは省略されます。移行情報は、実行の最後に移行概要に要約されます。

第8章 Migration errors

移行プロセス中、移行の概要に移行の進行状況が表示されます。エラーが発生すると、概要にエラーに関する情報が表示されます。移行の概要もログファイルに書き込まれます。

Linux では、ログファイルのデフォルトの場所は次のとおりです。

```
$HOME/.enterprisedb/migration-toolkit/logs
```

Windows では、ログファイルは次の場所に保存されます。

```
%HOMEDRIVE%\%HOMEPATH%\enterprisedb\migration-toolkit\logs
```

Migration Toolkit の `-logdir <log_path>` オプションを使用して、代替ログファイルの場所を指定できます。

</ div>

8.1 接続エラー

Migration Toolkit は、`toolkit.properties` file の情報を使用して、ソースデータベースとターゲットデータベースに接続します。Migration Toolkit の使用時に発生する接続エラーのほとんどは、`toolkit.properties` ファイルで指定された情報に関連しています。

8.1.1 無効なユーザー名/パスワード

- Migration Toolkit を使用してオラクルデータベースから移行を実行しようとする、次のエラーが表示されます。*

```
MTK-11009: Error Connecting Database "Oracle"
DB-1017: java.sql.SQLException: ORA-01017: invalid username/password;
logon denied
The user name or password specified in the `toolkit.properties` file is
not valid to use to connect to the Oracle source database.
```

このエラーを解決するには、`SRC_DB_USER` および `SRC_DB_PASSWORD` プロパティで移行を実行する権限を持つ有効なユーザの名前とパスワードを指定して、`toolkit.properties` ファイルを編集します。

8.1.2 接続が拒否されました：致命的：パスワード

- Migration Toolkit で移行を実行しようとする、次のエラーが表示されます。*

```
MTK-11009: Error Connecting Database "EDB Postgres EDB Postgres Advanced Server"
DB-28P01: com.edb.util.PSQLException: FATAL: password authentication
failed for user "name"
```

toolkit.properties ファイルで指定されたユーザ名前またはパスワードは、Postgres データベースへの接続に使用するには有効ではありません。

このエラーを解決するには、TARGET_DB_USER および TARGET_DB_PASSWORD プロパティで移行を実行する権限を持つ有効なユーザの名前とパスワードを指定して、toolkit.properties ファイルを編集します。

8.1.3 例外：ORA-28000：アカウントはロックされています

- Migration Toolkit を使用してオラクルデータベースから移行を実行しようとする、次のエラーメッセージれます。*

```
MTK-11009: Error Connecting Database "Oracle"
DB-28000: java.sql.SQLException: ORA-28000: the account is locked
The Oracle account associated with the user name specified in the
toolkit.properties file is locked.
```

このエラーを解決するには、オラクルサーバーでユーザアカウントのロックを解除するか、SRC_DB_USER および SRC_DB_PASSWORD パラメーターで移行を実行する権限を持つ有効なユーザの名前とパスワードを指定して、toolkit.properties ファイルを編集します。

8.1.4 例外：oracle.jdbc。ドライバ。OracleDriver

- Migration Toolkit で移行を実行しようとする、移行が失敗し、エラーメッセージが表示されます：*

```
MTK-11009: Error Connecting Database "Oracle"
java.lang.ClassNotFoundException: oracle.jdbc.driver.OracleDriver
```

Migration Toolkit を使用する前に、移行元のデータベースに適切な JDBC ドライバをダウンロードしてインストールする必要があります。完全な手順については、[Installing Source-Specific Drivers](#) を参照してください。

8.1.5 I/O 例外：ネットワークアダプターは接続を確立できませんでした

- Migration Toolkit で移行を実行しようとする、次のエラーが表示されます。*

```
MTK-11009: Error Connecting Database "Oracle"
DB-17002: java.sql.SQLException: Io exception: The
Network Adapter could not establish the connection
```

toolkit.properties ファイルで指定されたソースデータベースの JDBC URL に無効な接続プロパティが含まれています。

このエラーを解決するには、を編集し、プロパティでソースデータベースの有効な接続情報を指定します。値はデータベース固有です。ソースデータベースタイプの詳細情報を参照してください。

8.1.6 例外：「ターゲット」データベースに指定された URL は無効です

- Migration Toolkit で移行を実行しようとする、次のエラーが表示されます。*

```
MTK-10045: The URL specified for the "target" database is invalid.
Check the connectivity credentials.
```

toolkit.properties ファイルで指定されているターゲットデータベース（EDB Postgres Advanced Server）の JDBC URL に無効な接続プロパティが含まれています。

このエラーを解決するには、TARGET_DB_URL プロパティでターゲットデータベースの有効な接続情報を指定して、toolkit.properties ファイルを編集します。EDB Postgres Advanced Server の JDBC URL の形成については、[Defining an EDB Postgres Advanced Server URL](#) を参照してください。

</div>

8.2 移行エラー

Migration Toolkit がターゲットおよびソースデータベースサーバーに正常に接続した後、次のエラーが発生する可能性があります。

8.2.1 エラー：最後の予想列の後の追加データ

- テーブルをオンラインで移行すると、エラーメッセージが表示されます：*

```
MTK-17001: Error Loading Data into Table: table_name
DB-22P04: com.edb.util.PSQLException: ERROR: extra data after last
expected column
Where: COPY table_name, line 5: "50|HR|LOS|ANGELES"
```

このエラーは、`table_name` の列のデータにデリミタ文字が含まれている場合に発生します。このエラーを修正するには、デリミタ文字をテーブルの内容にない文字に変更します。

!!! Note * この例では、最後の列への挿入を意図したテキスト `LOS|ANGELES` にパイプ文字 (`|`) **があり、** ```-copyDelimiter '|'``` オプションを使用して Migration Toolkit が実行され、エラーが発生します。

8.2.2 テーブルへのデータローディングエラー：* TABLE_NAME *

- データのみの移行を実行すると、次のエラーが表示されます：*

```
MTK-17001: Error Loading Data into Table: TABLE_NAME
DB-42P01: com.edb.util.PSQLException: ERROR: relation
"schema.\ table_name" does not exist
* I also get the error:*
Trying to reload table: TABLE_NAME through bulk inserts with a batch
size of 100
MTK-17001: Error Loading Data into Table: TABLE_NAME
DB-42P01: com.edb.util.PSQLException: ERROR: relation
"schema.\ table_name" does not exist
```

データを移行する前に、ターゲットデータベースのデータを受信するテーブルを作成する必要があります。TABLE_NAME という名前のテーブルがターゲットデータベースに存在することを確認します。必要に応じてテーブルを作成し、データ移行をリトライします。

8.2.3 制約 CONS_NAME_FK の作成エラー

- インデックスとコンストレインを含むテーブル移行を実行すると、次のエラーメッセージれます。*

```
MTK-15001: Error Creating Constraint EMP_DEPT_FK
DB-42P01: com.edb.util.PSQLException: ERROR: relation "hr.departments"
does not exist
Creating Constraint: EMP_JOB_FK
MTK-15001: Error Creating Constraint EMP_JOB_FK
DB-42P01: com.edb.util.PSQLException: ERROR: relation "hr.jobs" does not exist
Creating Constraint: EMP_MANAGER_FK
Schema HR imported with errors.
One or more schema objects could not be imported during the migration
process. Please review the migration output for more details.
Migration logs have been saved to
/home/user/.enterprisedb/migration-toolkit/logs
***** Migration Summary *****
Tables: 1 out of 1
Constraints: 4 out of 6
Total objects: 7
Successful count: 5
```

(次のページに続く)

(前のページからの続き)

```
Failed count: 2
Invalid count: 0
List of failed objects
=====
```

```
Constraints
-----
```

1. HR.EMPLOYEES.EMP_DEPT_FK
2. HR.EMPLOYEES.EMP_JOB_FK

移行するテーブルには、ターゲットデータベースに存在しないテーブルに対する外部キー制約が含まれています。Migration Toolkit は、外部キー制約を省略してテーブルを作成します。

Migration Toolkit コマンドに `-skipFKConst` オプションを含めることにより、エラーメッセージの生成を回避できます。

8.2.4 テーブルへのデータ読み込みエラー

- テーブル定義は既に移行済みです。データをテーブルに移行しようとすると、エラーが発生します：*

```
MTK-17001: Error Loading Data into Table: DEPARTMENTS
DB-22P04: com.edb.util.PSQLException: ERROR: extra data after last
expected column
Where: COPY departments, line 1: "10 Administration 200 1700"
Trying to reload table: DEPARTMENTS through bulk inserts with a batch
size of 100
MTK-17000: Batch entry 0 INSERT INTO hr.DEPARTMENTS VALUES (10,
'Administration', 200, 1700); was aborted. Call getNextException to see
the cause.
DB-42601: java.sql.BatchUpdateException: Batch entry 0 INSERT INTO
hr.DEPARTMENTS VALUES (10, 'Administration', 200, 1700); was aborted.
Call getNextException to see the cause., Skipping Batch
MTK-17000:ERROR: INSERT has more expressions than target columns
Position: 48
[DEPARTMENTS] Table Data Load Summary: Total Time(s): 0.037 Total Rows:0
Data Load Summary: Total Time (sec): 0.168 Total Rows: 0 Total Size(MB):0.0
Schema HR imported with errors.
```

ターゲットデータベースのテーブル定義が、移行されたデータとマッチしません。ターゲットまたはソーステーブルを変更した場合は、テーブル定義とデータに互換性があることを確認してください。

エラー：タイプに対して値が長すぎます

- テーブル定義は既に移行済みです。データをテーブルに移行しようとすると、次のエラーが表示されます。*

```
MTK-17001: Error Loading Data into Table: DEPARTMENTS
DB-22001: com.edb.util.PSQLException: ERROR: value too long for type
character(1)
Where: COPY departments, line 1, column location_id: "1700"
Trying to reload table: DEPARTMENTS through bulk inserts with a batch
size of 100
MTK-17000: Batch entry 0 INSERT INTO hr.DEPARTMENTS VALUES (10,
'Administration', 200, 1700); was aborted. Call getNextException to see
the cause.
DB-22001: java.sql.BatchUpdateException: Batch entry 0 INSERT INTO
hr.DEPARTMENTS VALUES (10, 'Administration', 200, 1700); was aborted.
Call getNextException to see the cause., Skipping Batch
MTK-17000:ERROR: value too long for type character(1)
```

ターゲットデータベースの列は、移行されたデータを受け取るのに十分なラージさではありません。この問題は、移行後にテーブル定義が変更された場合に発生する可能性があります。列名前（この例では、location_id）は、Where: で始まる行で識別されます。

場所: COPY departments, line 1, column location_id: "1700"

問題を修正するには、列サイズを調整し、移行をリトライします。

エラー：スレッドの例外：OutOfMemoryError

- MySQL データベースから移行すると、次のエラーが発生します。*

```
Loading Table: big_range ...
Exception in thread "dataload_job_1" java.lang.OutOfMemoryError: Java
heap space
at com.mysql.jdbc.MySQLIO.nextRow(MySQLIO.java:1370)
at com.mysql.jdbc.MySQLIO.readSingleRowSet(MySQLIO.java:2369)
at com.mysql.jdbc.MySQLIO.getResultSet(MySQLIO.java:451)
at com.mysql.jdbc.MySQLIO.readResultsForQueryOrUpdate(MySQLIO.java:2076)
at com.mysql.jdbc.MySQLIO.readAllResults(MySQLIO.java:1451)
at com.mysql.jdbc.MySQLIO.sqlQueryDirect(MySQLIO.java:1787)
at com.mysql.jdbc.Connection.execSQL(Connection.java:3277)
at com.mysql.jdbc.Connection.execSQL(Connection.java:3206)
at com.mysql.jdbc.Statement.executeQuery(Statement.java:1232)
at com.edb.dbhandler.mysql.Data.getTableData(Data.java:90)
at com.edb.DataLoader.loadDataInFastMode(DataLoader.java:594)
at com.edb.DataLoader.run(DataLoader.java:186)
at java.lang.Thread.run(Thread.java:722)
```

デフォルトでは、MySQL JDBC ドライバは、1 回のネットワークラウンドトリップで、テーブル内のすべての行を Migration Toolkit にフェッチします。この動作は、ラージテーブルで使用可能なメモリを簡単に超える可能性があります。

問題を修正するには、移行をリトライするときにコマンドライン引数として `-fetchSize 1` を指定します。

第9章 Error codes

Migration Toolkit で問題が発生すると、エラーコードとメッセージのフォームで通知が発行されます。

各エラーコードは、プレフィックス MTK-で始まり、その後に 5 桁が続きます。最初の 2 桁は、エラーの一般的な分類であるエラークラスを示します。最後の 3 桁は特定のエラー状態を表します。

次の表にエラークラスを示します。

Error Class	Description
00	正常に完了
01	情報
02	警告
03	一般的な drror
0x	予約済み
10	無効なユーザ入力
11	構成エラー
12	不十分な特権
13	サポートされていない機能
14	不足しているオブジェクト
15	スキーマの移行
16	手続き型言語の移行
17	データのローディング
	NA

特定のデータベースサーバからエラーが報告された場合、このエラーメッセージの先頭には DB-が付きます。例、Postgres データベースサーバの既存のテーブルが原因でテーブルの作成が失敗した場合、エラーコード 42P07 がデータベースサーバによって返されます。Migration Toolkit ログの特定のエラーは、DB-42P07 として表示されます。

9.1 エラーコードの概要

次の表では、列エラーコードに Migration Toolkit エラーコードがリストされています。 ** Message and resolution **列には、エラーコードとともに表示されるメッセージが含まれます。メッセージには、エラーの原因とその解決方法が説明されています。

** Message and resolution **列の \$NAME は、ランタイムに適切な値に置き換えられる情報のプレースホルダです。

9.1.1 クラス 02-警告

このクラスはワーニングメッセージを表します。

Error code	Message and resolution
MTK-02000	All the warnings that relate to this class and don't have a specific error code binding use this error code.
MTK-02001	Run 'runMTK -help' to see the usage details.
MTK-02002	Warning! The offline migration path <i>\$OFFLINE_PATH</i> does not exist, the scripts will be created under the user home folder.

9.1.2 クラス 10-無効なユーザ入力

このクラスは、無効なユーザ入力値を表します。

9.1.3 クラス 11-構成の問題

このクラスは、`toolkit.properties` ファイルまたは Migration Toolkit で使用される他の構成ファイルの無効な構成設定を表します。

Error code	Message and resolution
MTK-11000	All the errors that relate to this class and don't have a specific error code binding use this error code.
MTK-11001	The properties file containing table filter clause cannot be loaded.
MTK-11002	The custom type mapping file couldn't be loaded. Reason: <i>\$REASON</i> .
MTK-11003	The custom type mapping file contains an invalid mapping entry.
MTK-11004	The custom type mapping file doesn't contain any mapping entry.
MTK-11005	Connectivity information for source database is not available.
MTK-11006	Connectivity information for target database is not available.
MTK-11007	Unable to create the logs directory. <i>\$PATH</i> .
MTK-11008	The properties file containing table columns filter list cannot be loaded.
MTK-11009	Error Connecting Database <i>\$DATABASE</i> .
MTK-11010	Error loading <i>\$DBLINKORA_FILE</i> file.
MTK-11011	Error while loading DBLink Ora module. <i>\$DBLINKORA_MODULE</i> . Verify that <code>dblink_ora</code> is installed/configured on target EnterpriseDB server. Please see the Database Compatibility for Oracle Developer's Guide for more information about installing and configuring the <code>dblink_ora</code> module.

次のページに続く

表 3 – 前のページからの続き

Error code	Message and resolution
MTK-11012	Error while loading given DBLink_Ora module. <i>\$DBLINKORA_MODULE</i> . Verify that dblink_ora is installed/configured on target EnterpriseDB server. Please see the Database Compatibility for Oracle Developer's Guide for more information about installing and configuring the dblink_ora module.
MTK-11013	Could not load DBLinkOra Module.
MTK-11014	Error connecting to DBLinkOra.
MTK-11015	The connection credentials file <i>\$TOOLKIT_PROP_FILE</i> is not secure and accessible to group/others users. This file contains plain passwords and should be restricted to Migration Toolkit owner user only.
MTK-11016	Data loader task failed to initialize. Reason: \$REASON

9.1.4 クラス 12-不十分な特権

このクラスは、データベースユーザ/ロール情報をローディングするための不十分権限エラーを表します。

Error code	Message and resolution
MTK-12000	All the errors that relate to this class and don't have a specific error code binding use this error code.
MTK-12001	The user/role migration failed due to insufficient privileges. Grant the user SELECT privilege on the following Oracle catalogs: DBA_ROLES, DBA_USERS, DBA_TAB_PRIVS, DBA_PROFILES, DBA_ROLE_PRIVS, ROLE_ROLE_PRIVS, DBA_SYS_PRIVS.
MTK-12002	The migration of privileges failed due to insufficient privileges. Grant the user SELECT privilege on the following Oracle catalog: dba_tab_privs.

9.1.5 クラス 13-サポートされていない機能

このクラスは、サポートされていないオブジェクトおよび句の移行に関連するエラーを表します。ターゲットデータベースが指定されたオブジェクトを実装していないか、Migration Toolkit がその移行をハンドリングしていません。

Error code	Message and resolution
MTK-13000	All the errors that relate to this class and don't have a specific error code binding use this error code.
MTK-13001	Skipping index as Oracle does not allow multiple indexes against same column list.

次のページに続く

表 5 – 前のページからの続き

Error code	Message and resolution
MTK-13002	For <i>\$DATABASE</i> views migration is not supported.
MTK-13003	For <i>\$DATABASE</i> roles migration is not supported.
MTK-13004	You cannot migrate triggers, sequences, procedures, functions, packages and synonyms for MySQL, SQL Server, and Sybase databases.
MTK-13005	You cannot migrate sequences and packages for SQL Server database.
MTK-13006	Given trigger is not migrated, the trigger is owned by a different user schema and cross-schema triggers are not supported by EnterpriseDB.
MTK-13007	The given trigger is not migrated, the trigger has WHEN clause which is not supported by EnterpriseDB.
MTK-13008	Skipping Database Link <i>\$DATABASE_LINK</i> . EnterpriseDB currently does not support this type of Database Link.
MTK-13009	Warning! Skipping migration of trigger <i>\$TRIGGER</i> , currently non-table triggers are not supported in target database.
MTK-13010	You cannot migrate procedures, packages, synonyms and database links to PostgreSQL database.
MTK-13011	Domain objects are not supported in target database.
MTK-13012	Rules are not supported in <i>\$DATABASE</i> database.
MTK-13013	The database type is not supported.
MTK-13015	<i>\$TYPE</i> is Not Supported by COPY.
MTK-13016	The migration to PostgreSQL is supported only when Oracle or PostgreSQL is the source database.
MTK-13017	Groups are not supported in <i>\$DATABASE</i> database.
MTK-13018	Profile migration is not supported in <i>\$SRC_DB</i> to <i>\$TARGET_DB</i> permutation.
MTK-13019	Cannot migrate unknown Profile <i>\$PROFILE</i> .
MTK-13020	Profiles cannot be migrated to database version <i>\$VERSION</i> .
MTK-13021	Password Profile verify function <i>\$MY_VERIFICATION_FUNCTION</i> must be explicitly migrated.
MTK-13022	Advanced Queues migration is not supported from <i>\$SRC_DB</i> database to <i>\$TARGET_DB</i> database.
MTK-13023	Advanced Queues cannot be migrated to database version <i>\$VERSION</i> .

次のページに続く

表 5 – 前のページからの続き

Error code	Message and resolution
MTK-13024	The INTERVAL partition is not supported in <i>\$DATABASE</i> , the table will be migrated without INTERVAL definition.
MTK-13025	Warning! User profile migration is not supported in target database version <i>\$VERSION</i> . The profile \$PROFILE for user \$USER will be skipped.
MTK-13026	Object Type migration is not supported from <i>\$SRC_DB</i> to <i>\$TARGET_DB</i> .
MTK-13027	Migration is not supported from <i>\$DATABASE</i> database for PDB Admin schemas/users with PDB_DBA role.

9.1.6 クラス 14-欠落オブジェクト

このクラスは、ソースデータベース内のメタデータ情報の検索の失敗を表します。

Error code	Message and resolution
MTK-14000	All the errors that relate to this class and don't have a specific error code binding use this error code.
MTK-14001	One or more tables are missing from the source <i>\$DATABASE</i> database.
MTK-14002	One or more tables couldn't be found in the source <i>\$DATABASE</i> database. With -tables mode, the table name should be in uppercase unless it is case-sensitive.
MTK-14003	One or more users couldn't be found in the source <i>\$DATABASE</i> database. With -users mode, the user name should be in uppercase unless it is case-sensitive.

9.1.7 クラス 15-スキーマの移行

このクラスは、テーブル、コンストレイン、インデックス、同義語、ビュー、ユーザー、ロールなどの非手続き型データベースオブジェクトに関連する移行の問題を表します。

Error code	Message and resolution
MTK-15000	All the errors that relate to this class and don't have a specific error code binding use this error code.
MTK-15001	Error creating constraint <i>\$CONSTRAINT</i> .

次のページに続く

表 7 – 前のページからの続き

Error code	Message and resolution
MTK-15002	Error creating role <i>\$ROLE</i> .
MTK-15003	Error granting given privilege <i>\$PRIVILEGE</i> on <i>\$OBJECT</i> to <i>\$USER</i> .
MTK-15004	Error granting <i>\$ROLE</i> to <i>\$USER</i> .
MTK-15005	Error granting privilege <i>\$REASON</i> .
MTK-15006	Error creating user <i>\$USER</i> .
MTK-15007	Error creating index <i>\$INDEX</i> .
MTK-15008	Error creating view <i>\$VIEW</i> .
MTK-15009	Error creating materialized view <i>\$MVIEW</i> .
MTK-15010	Error creating public synonym <i>\$SYNONYM</i> .
MTK-15011	Error creating private synonym <i>\$SYNONYM</i> .
MTK-15012	Error creating sequence <i>\$SEQUENCE</i> .
MTK-15013	Error creating table <i>\$TABLE</i> .
MTK-15014	Error creating database link <i>\$DATABASE_LINK</i> .
MTK-15015	Error creating given object type <i>\$OBJECT_TYPE</i> .
MTK-15016	The table <i>\$TABLE</i> could not be created in <i>\$DATABASE</i> database.
MTK-15017	The linked schema <i>\$LINKED_SCHEMA</i> doesn't exist in the target database. Create the schema and then retry.
MTK-15018	Error creating domain <i>\$DOMAIN</i> .
MTK-15019	Error creating custom data type <i>\$DATA_TYPE</i> .
MTK-15020	Error creating membership for group <i>\$GROUP</i> .
MTK-15021	Table name <i>\$TABLE</i> does not have a schema qualifier. With multiple schema migration context, each table should be schema qualified.
MTK-15022	Schema qualifier <i>\$SCHEMA</i> does not match the schema list.
MTK-15023	The table metadata information is not available.
MTK-15024	Tables list is not initialized yet.
MTK-15025	Error while getting database metadata information.
MTK-15026	Error creating group <i>\$GROUP</i> .
MTK-15027	Error creating Profile <i>\$PROFILE</i> .

9.1.8 クラス 16-手続き言語の移行

このクラスは、プロシージャ、関数、パッケージソフト、匿名ブロック、トリガー、ルールなどの手続き言語に基づくデータベースオブジェクトに関連する移行の問題を表します。

Error code	Message and resolution
MTK-16000	All the errors that relate to this class and don't have a specific error code binding use this error code.
MTK-16001	Error Creating Trigger <i>\$TRIGGER</i> .

次のページに続く

表 8 – 前のページからの続き

Error code	Message and resolution
MTK-16002	Error Creating Rule <i>\$RULE</i> .
MTK-16003	Error Creating Package Spec <i>\$PACKAGE</i> .
MTK-16004	Error Creating Package Body <i>\$PACKAGE</i> .
MTK-16005	Package Body is invalid, skipping... Note: This error message also appears when a package specification is successfully migrated, but there is no corresponding package body in the source database. In this case, the package specification should function properly in the target database despite the appearance of the error message.
MTK-16006	Error Creating Procedure <i>\$PROCEDURE</i> .
MTK-16007	Error Creating Function <i>\$FUNCTION</i> .
MTK-16008	Error Creating Anonymous Block <i>\$BLOCK</i> .
MTK-16009	Error creating Queue <i>\$QUEUE</i> .

9.1.9 クラス 17-データのローディング

このクラスは、ソースからターゲットデータベースにデータをコピーするときに発生する可能性のあるエラーを表します。

Error code	Message and resolution
MTK-17000	All the errors that relate to this class and don't have a specific error code binding use this error code.
MTK-17001	Error Loading Data into Table: <i>\$TABLE</i>
MTK-17002	Encoding Conversion encountered some characters that will be loaded using Bulk Inserts instead.
MTK-17003	Error in copy tables <i>\$REASON</i> .
MTK-17004	Invalid DataType.
MTK-17005	This Table Contains CLOB data, Marked for Bulk Insert Loading.
MTK-17006	One of the table loader threads failed while copying data for table <i>\$TABLE</i> .

第10章 Unsupported features

EDB Postgres Advanced Server は、一部のオラクル機能を完全にサポートし、その他の機能を部分的にサポートしています。Migration Toolkit は、サポートされていない機能を使用するオブジェクトを移行できません。

場合によっては、Migration Toolkit は部分的な互換性を提供する機能を使用するオブジェクトを移行できます。その他の場合、EDB Postgres Advanced Server は適切な回避策をサポートします。

テキストサーチは、オラクルと完全に互換性がない機能の例です。EDB Postgres Advanced Server データベースには、かなり長い間フルテキストサーチのサポートが含まれていますが、実装は Oracle とは異なります。Migration Toolkit は、この機能を使用するオブジェクトを移行できません。

EDB Postgres Advanced Server はまだ他の機能をサポートしていません。このカテゴリの機能には、自動ストレージ管理、表圧縮、外部表が含まれます。多くの場合、成功する回避策を実装できます。

- 自動ストレージ管理をシステム固有のボリューム管理ソフトウェアに置き換えることができます。
- 圧縮ファイルシステムにあるテーブルスペースにデータを保存することにより、テーブル圧縮を実装できます。
- 外部テーブルは EDB Postgres Advanced Server にありませんが、フラットテキストファイルをデータベースのステー징テーブルにロードできます。EDB Postgres Advanced Server データベースにデータをすばやくロードするには、EDB * Loader ユーティリティを使用することをお勧めします。
- マルチプルのプロファイルをオラクルデータベースから EDB Postgres Advanced Server に移行する場合、移行の完了時に 1 つまたは複数のユーザにプロファイルを手動で割り当てる必要があります。

</ div>

10.1 サポートされていない Postgres 機能

Migration Toolkit は、次の Postgres 機能の移行をサポートしていません。

- OPERATOR CLASS
- OPERATOR FAMILY

OPERATOR CLASS および OPERATOR FAMILY については、[PostgreSQL core documentation](#) を参照してください。

</ div>

10.2 よくある質問

**** Migration Toolkit はパッケージソフトの移行をサポートしていますか？ ****

Migration Toolkit は、オラクルデータベースから EDB Postgres Advanced Server へのパッケージソフトの移行をサポートしています。EDB Postgres Advanced Server が提供する移行サポートについては、[Functionality overview](#) を参照してください。

ソースデータベースからのデータのみを転送する方法はありますか？

はい。データ転送は、オンラインまたはオフラインの移行のパートとしてサポートされています。

**** Migration Toolkit は、CLOB データタイプのデータを含むテーブルの移行をサポートしていますか？ ****

Migration Toolkit は、CLOB タイプのデータを含むテーブルの移行をサポートします。

**** EDB Postgres Advanced Server は enum データタイプをサポートしていますか？ ****

EDB Postgres Advanced Server は現在 enum データタイプをサポートしていませんが、将来のリリースではサポートする予定です。それまでは、検査制約を使用して、EDB Postgres Advanced Server データベースに追加されるデータを制限できます。検査制約は、列が取ることができる有効な値のリストを定義します。

次のコードサンプルには、列の値を 3 つの dept タイプのいずれかに制限する検査制約のシンプル例が含まれています。sales、admin、または technical。

```
CREATE TABLE emp (
  * emp_id INT NOT NULL PRIMARY KEY,
  * dept VARCHAR(255) NOT NULL,
  * CHECK (dept IN ('sales', 'admin', 'technical'))
);
```

有効な dept タイプを入力して検査制約をテストすると、INSERT ステートメントはエラーなしで機能します。

```
test=# INSERT INTO emp VALUES (7324, 'sales');

INSERT 0 1
```

制約 (support) に含まれていない値を挿入しようとする、EDB Postgres Advanced Server はエラーを返します。

```
test=# INSERT INTO emp VALUES (7325, 'support');

ERROR: new row for relation "emp" violates check constraint "emp_dept_check"
```

**** EDB Postgres Advanced Server はマテリアライズドビューをサポートしていますか？ ****

Postgres は、オラクルデータベースと互換性のあるマテリアライズドビューをサポートしていません。Postgres でマテリアライズドビュー/サマリテーブルを設定するには、サマリテーブルを維持するトリガーを手動で作成する必要があります。自動クエリー書き換えは現在サポートされていません。サマリー表の存在をアプリケーションに認識させる必要があります。

**** TIME データタイプを含む MySQL データベースから移行しようとすると、次のエラーが表示されます。エラーデータをテーブルにロード中：時間のフォーマットが正しくありません。Postgres は MySQL \TIME \データ型をサポートしていますか？****

Postgres には、時間コンポーネントの値が 24 をより大きい値に限り、TIME データ型の保存に問題はありません。

Postgres とは異なり、MySQL TIME データタイプを使用すると、TIME または INTERVAL 値を表す値を格納できます。INTERVAL 値を表す MySQL TIME 列に格納された値は、有効な Postgres TIMESTAMP 値の許容レンジ外になる可能性があります。移行プロセス中に、Postgres が TIME データ列に格納された値を検出し、レンジ外であると認識した場合、エラーを返します。

第11章 Release Notes

Migration Toolkit の文書には、マイナーリリースとパッチを含む Migration Toolkit 55 の最新バージョンが記載されています。このセクションのリリースノートには、各リリースの新機能に関する情報が記載されています。マイナーリリースまたはパッチリリースで導入された新しい機能については、どのリリースが機能を導入したかに関するコンテンツ内のインジケータもあります。

Version	Release Date
[55.2.0](07_mkt_552_rel_note)	21 Dec 2021
[55.1.0](08_mkt_551_rel_note)	20 Sep 2021
[55.0.0](09_mkt_55_rel_note)	19 Mar 2020

11.1 Version 55.2.0

Migration Toolkit 55.2.0 の新機能、機能強化、バグ修正、およびその他の変更には以下が含まれます。

Type	Description
Enhancement	Migration Toolkit は、PostgreSQL をサポートすることが認定されています。
Enhancement	新しいオラクル互換性機能の一部を含む EPAS14 のサポートが追加されました。
Enhancement	LIST BY PARTITION 句の後に AUTOMATIC キーワードが続くパーティションテーブルを、ターゲットの EPAS13 および 14 データベースに移行できるようになりました。
Enhancement	Migration Toolkit は、SUBPARTITION TEMPLATE 句を使用してパーティションテーブルをターゲット EPAS14 データベースに正常に移行できるようになりました。
Bug Fix	1000 人を超えるユーザーの移行を成功させるために ORA-01795 の問題を修正しました。
Bug Fix	1000 を超えるパーティションを正常に移行できるように ORA-01795 の問題を修正しました。

11.2 Version 55.1.0

Migration Toolkit 55.1.0 の新機能、機能強化、バグ修正、およびその他の変更には以下が含まれます。

Type	Description
Enhancement	Migration Toolkit は、MySQL5.7 および 8.0 からの移行をサポートするように拡張されています。
Bug Fix	MySQL から PostgreSQL への移行の場合、ブールデータタイプは LO モードで移行していませんでした。この問題は修正されました。
Bug Fix	番号で始まるテーブル名を MySQL から PostgreSQL に移行できるようにする問題を修正しました。
Bug Fix	MySQL から PostgreSQL に移行するときに同じタイムゾーン内で TIMESTAMPTAMP 値が変更される問題を修正しました。
Bug Fix	MySQL から PostgreSQL への移行中に、fastcopy モードでターゲットデータベースに enum タイプが作成されていませんでした。この問題は修正されました。
Bug Fix	n> 1 の場合の MySQL から PostgreSQL への BIT (n) データタイプの移行に関連する障害を修正しました。
Bug Fix	オラクルから PostgreSQL への移行中に、数値フォーマットの例外が報告される問題を修正しました (72863)。
Bug Fix	関連する OID 値が INTEGER のレンジを超えると、PostgreSQL テーブルの移行は失敗します。この問題は修正されました (72694)。
Bug Fix	JSON データタイプの PostgreSQL からオラクルへの移行に関連する問題を修正しました。
Bug Fix	オラクルから EDB Postgres Advanced Server の場合、特定のシステムユーザー/スキームが移行対象として誤って選択されていました。この問題は修正されました。
Bug Fix	並列ローディングモードで大文字と小文字を区別するスキーマ/テーブル名をオラクルから EDB Postgres Advanced Server に移行できるようにする問題を修正しました。
Bug Fix	カスタム列タイプマッピングオプションの使用に関連していた特定のコーナーケースが修正されました。

11.3 Version 55.0.0

Migration Toolkit 55.0.0 の新機能、機能強化、バグ修正、およびその他の変更には以下が含まれます。

Type	Description
Enhancement	テーブルレベルでの並列データローディングのサポート。これにより、マルチプルのスレッドを使用して、ソースデータベーステーブルからデータを並列にロードし、ターゲットデータベーステーブルに並列に適用できます。これにより、比較的ラージサイズのテーブルを処理する場合のデータロード時間が大幅に改善されます。ラボのベンチマーク結果では、マルチプルの並列スレッドを使用すると、ラージテーブルのデータロード時間が25～60 %短縮されることが明らかになりました。この機能は、ソース PostgreSQL、EDB Postgres Advanced Server、およびオラクルデータベースでサポートされています。 [Migration Options for Parallel Data Loading](http://localhost:8000/migration_toolkit/latest/08_mtk_command_options/#migration-options-for-parallel-data-loading) を参照してください。
Enhancement	テーブルレベルの並列データローディングの機能強化のパートとして、`tableLoaderLimit` および `parallelLoadRowLimit` オプションがコマンドラインオプションリストに追加されました。
Enhancement	オラクルバージョン 18c および 19c からのスキーマおよびデータ移行のサポート。
Enhancement	スキーマおよびデータ移行セッションの合計移行時間は、`Total Elapsed Migration Time` 属性を介して報告されます。
Bug Fix	テーブルのデータ移行が失敗した場合、要約レポートはステータスを正確にレポートしません。
Bug Fix	要約レポートは、冗長な無効なオブジェクトを削除しません。
Bug Fix	指数フォーマットで報告されるテーブルのデータサイズ。
Bug Fix	`-tables` リストを介して重複するテーブル名前が指定された場合の失敗を回避します。
Bug Fix	以前はメガバイトと表示されていた小さなテーブルサイズを反映します。