



EDB

Migration Toolkit Guide

リリース **54.0.0**

Migration Toolkit Guide

2021 年 01 月 24 日

目次

第 1 章	What's New	3
第 2 章	Supported Operating Systems and Database Versions	5
第 3 章	Migration Methodology	7
3.1	移行プロセス	7
3.2	アプリケーションを Postgres に接続する	9
第 4 章	Functionality Overview	11
第 5 章	Installing Migration Toolkit	13
5.1	OEL、CentOS、または RHEL7 または 8 ホストへの移行ツールキットのインストール	14
5.1.1	Upgrading an RPM Installation	16
5.2	CentOS または RHEL7ppc64le ホストへの移行ツールキットのインストール	17
5.3	Debian または Ubuntu ホストへの移行ツールキットのインストール	19
5.4	SLES ホストへの移行ツールキットのインストール	20
5.5	Windows への MigrationToolkit のインストール	21
5.6	MacOSX に MigrationToolkit をインストールする	32
5.7	ソース固有のドライバーのインストール	34
第 6 章	Building the toolkit.properties File	37
6.1	EDBPostgresAdvancedServerURL の定義	39
6.2	PostgreSQLURL の定義	41
6.3	オラクル URL の定義	43
6.4	MySQLURL の定義	44
6.5	SybaseURL の定義	45
6.6	SQLServerURL の定義	46
第 7 章	Invoking Migration Toolkit	47
7.1	オラクルからのスキーマの移行	48
7.2	Oracle 以外のソースデータベースからの移行	49
第 8 章	Migration Toolkit Command Options	51
8.1	オフライン移行オプション	53
8.1.1	オフライン移行スクリプトの実行	54
8.2	インポートオプション	55
8.3	スキーマ作成オプション	56
8.4	スキーマオブジェクト選択オプション	57
8.5	移行オプション	60
8.6	オラクル固有のオプション	64
8.7	その他のオプション	69
8.8	例	70

第 9 章 Migration Errors	73
9.1 接続エラー	73
9.1.1 無効なユーザー名/パスワード	73
9.1.2 接続が拒否されました:致命的:パスワード	74
9.1.3 例外:ORA-28000:アカウントはロックされています	74
9.1.4 例外:oracle.jdbc。ドライバ,OracleDriver	74
9.1.5 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	74
9.1.6 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	75
9.2 移行エラー	76
9.2.1 エラー:予想される最後の列の後の追加データ	76
9.2.2 テーブルへのデータ読み込みエラー:TABLE_NAME	76
9.2.3 制約 CONS_NAME_FK の作成エラー	77
9.2.4 テーブルへのデータ読み込みエラー	77
第 10 章 Error Codes	79
10.1 エラーコードの概要	79
10.1.1 クラス 02-警告	80
10.1.2 クラス 10-無効なユーザー入力	80
10.1.3 クラス 11-構成の問題	82
10.1.4 クラス 12-不十分な特権	83
10.1.5 クラス 13-サポートされていない機能	84
10.1.6 クラス 14-欠落オブジェクト	86
10.1.7 クラス 15-スキーマの移行	86
10.1.8 クラス 16-手続き言語の移行	88
10.1.9 クラス 17-データの読み込み	88
第 11 章 Unsupported Features	89
11.1 サポートされていない Postgres 機能	90
11.2 よくある質問	91
第 12 章 Conclusion	93
索引	95

MigrationToolkit 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。MigrationToolkit は、データベースオブジェクトとデータの EDBPostgres AdvancedServer または PostgreSQL データベースからの移行を容易にします。

- オラクル®
- MySQL®
- SQLServer の®

MigrationToolkit を使用して、データベースオブジェクトとデータを Sybase ® AdaptiveServer ® Enterprise から EDBPostgresAdvancedServer に、または EDBPostgresAdvancedServer と PostgreSQL 間で移行することもできます。

MigrationToolkit は、EDBPostgresAdvancedServer インストーラまたは StackBuilder を使用してインストールできます。StackBuilder は、EDBPostgresAdvancedServer と PostgreSQL のワンクリックインストーラの両方で配布され、EnterpriseDBWeb サイトから入手できます。

<https://www.enterprisedb.com/advanced-downloads>

このガイドでは、移行プロセスに関連する手順の概要と、MigrationToolkit のインストールおよび使用手順について説明します。また、一般的な移行の問題のソリューション、およびサポートされていない機能とその潜在的な回避策の詳細も含まれています。

Note:

このガイドでは、Postgres という用語を使用して、EDBPostgresAdvancedServer または PostgreSQL のインストールを「*」* 参照します。

このガイドでは、StackBuilder という用語を使用して*StackBuilderPlus（EDBPostgresAdvancedServer で配布）または StackBuilder（EnterpriseDB の PostgreSQL ワンクリックインストーラで配布）のいずれかを指します。

さらに、このドキュメントの pdf バージョンを使用している場合、カット/ペーストを使用してコマンドをコピーすると、貼り付けたコマンドに余分なスペースまたは改行が含まれる場合があります。コマンドが失敗した場合は、追加の文字がないかコマンドを慎重に確認してください。

第1章 What's New

54.0.0 リリースの EDBPostgresMigrationToolkit には、次の機能強化が追加されています。

- MigrationToolkit は現在、以下をサポートしています。
 - Postgres および EDBPostgresAdvancedServer バージョン 13
 - Ubuntu バージョン 20
 - Windows (64 ビット)

注釈: 既存の 32 ビット WindowsMigrationToolkit がインストール場合は、Windows64 ビットインストーラで MigrationToolkit をアンインストールして再インストールする必要があります。

サポートされているプラットフォームの完全なリストについては、[サポートされているオペレーティングシステムとデータベースバージョン](#) を参照してください。

第2章 Supported Operating Systems and Database Versions

Database Versions

MigrationToolkit 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 9.6、10、11、12、および 13
- EDBPostgresAdvancedServer バージョン 9.6、10、11、12、および 13
- 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
- SQLServer2008
- SQLServer2012
- SQLServer2014
- MySQL5.5.36
- SybaseAdaptiveServerEnterprise15.7

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: MigrationToolkit はテストされておらず、オラクル RealApplicationClusters (RAC) および Exadata での使用を公式にサポートしていません。ただし、MigrationToolkit 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。

Supported Operating Systems Versions

- Centos または RHEL または OEL7 および 8
- CentOS または RHELPPCLE7
- SLES12SP5
- Debian9 および 10
- Ubuntu18.04 および 20
- Windows (64 ビット)、2016、および 2019
- MacOSX10.12+

Note: ResultSet からのフェッチ中に、ojdbc7.jar により、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3章 Migration Methodology

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理由はたくさんあります。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を拡張するための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移行です。

移行も非常にコスト効果が高くなります。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、ベンダー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りした場合に、適切な移行パスを提供できます。EnterpriseDB は、企業が既存のデータベースシステムを長年 Postgres に移行するのを支援してきました。

移行プロセス で詳述されている方法に従うことをお勧めします。

3.1 移行プロセス

Postgres への移行パスには、次の主な手順が含まれます。

1. どのデータベースオブジェクトとデータを移行に含めるかを決定して、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 EDBPostgresAdvancedServer への移行の場合、詳細については、EnterpriseDB 文書<<ahref="https://www.enterprisedb.com/edb-docs/p/edb-postgres-advanced-server">https://www.enterprisedb.com/edb-docs/p/edb-postgres-advanced-server>_を参照してください。EDBPostgresAdvancedServer でサポートされている互換性機能。このレビューを支援するために、EnterpriseDB の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検討してスキーマ。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。 **移行エラーセクション** このドキュメントの移行問題の解決に関する情報を含みます。
5. データを移行します。小さいデータセットの場合は、MigrationToolkit を使用します。（EDBPostgresAdvancedServer への）オラクル移行であり、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、利用可能な他のデータ移動方法のいずれかを利用してください。

- オラクルデータベースと互換性のある EDBPostgresAdvancedServer データベースリンク機能を使用します。
- 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ではなく 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 CallInterface (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、EDBPostgresAdvancedServer 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 などのオープン標準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の変更と EnterpriseDB ドライバの選択のみが必要です。 [アプリケーションを Postgres に接続する](#) 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。EDBPostgresAdvancedServer データベースに移行する場合、EDBPostgresAdvancedServer 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。
 - 使用して、クエリーのパスを指示する Optimizer Hints。
 - ANALYZE コマンドを使用して、データベース統計を取得します。

EDBPostgresAdvancedServerGuide および *Database**Compatibility**for**オラクル**Developer's**Guide* (どちらも EnterpriseDB ウェブサイトから入手可能) は、EDBPostgresAdvancedServer 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を提供します。

3.2 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`）および JDBCURL 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。

- i. 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 のインストール後、ODBC Data Source Administrator へのリンクについては Administrative Tools メニューを確認してください。Add ボタンをクリックして、Create New Data Source ウィザードをスタートします。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。

- ii. 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`（または場合によっては `SQLDriverConnect`）への呼び出しが含まれます。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、[MigrationErrors](#) を参照してください。

第4章 Functionality Overview

MigrationToolkit 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。MigrationToolkit 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. コマンドラインで MigrationToolkit を起動し、移行オプションを指定します。

MigrationToolkit は、データベースオブジェクトとデータの EDBPostgresAdvancedServer または PostgreSQL データベースからの移行を容易にします。

- オラクル
- MySQL
- SQLServer

MigrationToolkit 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 EDBPostgresAdvancedServer データベースに移行することもできます。MigrationToolkit を使用して、EDBPostgresAdvancedServer と PostgreSQL の間で移行することもできます。MigrationToolkit 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、移行された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、移行を最適化移行。
- -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。
- COPY コマンドで使用されるバッチサイズを指定するには、-cpBatchSize オプションを使用します。
- -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、ユーザー定義の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スキーマ。
- EDBPostgresAdvancedServer 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。
- EDBPostgresAdvancedServer 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。

Object Migration Support

MigrationToolkit 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、MigrationToolkit 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。

対象	オラクル	Sybase	SQLServer	MySQL
スキーマ	X	X	X	X
テーブル	X	X	X	X
リストパーティションテーブル	X			
範囲パーティションテーブル	X			
ハッシュ分割テーブル	X			
Constraints	X	X	X	X
インデックス	X	X	X	X
トリガー	X			
テーブルデータ	X	X	X	X
Views	X		X	
マテリアライズド Views	X			
パッケージ	X			
Procedures	X			
Functions	X			
シーケンス	X			
ユーザー/ロール	X			
プロフィール	X			
オブジェクトタイプ	X			
オブジェクト型メソッド	X			
データベースリンク	X			
Queues	X			

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、[スキーマオブジェクト選択オプション](#) を参照してください。

Online Migration vs. Offline Migration

MigrationToolkit は、すぐに Postgres データベースに直接移行できます（オンライン**移行）。または、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後で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、MigrationToolkit は Postgres データベースにオブジェクトを直接作成します。対照的に、`-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。実行する前に MigrationToolkit によって生成された移行スクリプトをカスタマイズすることにより、移行されたオブジェクトを変更実行します。`-offlineMigration` オプションを使用すると、システム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、[オフライン移行オプション](#) を参照してください。

第5章 Installing Migration Toolkit

MigrationToolkit をインストールする前に、Java（バージョン 1.7.0 以降）をインストールする必要があります。

MigrationToolkit 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 ウェブサイトしてください。

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

5.1 OEL、CentOS、または RHEL7 または 8 ホストへの移行ツールキットのインストール

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

Creating a Repository Configuration File

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想定して、次のコマンドを呼び出します。

- RHEL または CentOS7 の場合:

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

- RHEL または CentOS8 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. 設定ファイルへの変更を保存した後、EPEL リポジトリをインストールします。

- RHEL または CentOS7 の場合:

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

- RHEL または CentOS8 の場合:

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

4. RHEL または CentOS8.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。

- CentOS8 (Powertools リポジトリ) :

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

- RHEL8 (codeready-builder-for-rhel-8-/*-rpmsrepo) の場合:

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

Installing Migration Toolkit

- RHEL または CentOS7 の場合:

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

- RHEL または CentOS8 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、信頼できるソースからのパッケージソフトことに満足したら、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。存在する場合は、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。

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

Using Migration Toolkit with IDENT Authentication

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の IDENT 認証を強制します。MigrationToolkit を呼び出す前に、pg_hba.conf ファイルを変更して、認証メソッドを IDENT 以外のフォームに変更する（およびサーバーを再起動する）か、次の手順を実行して IDENT サーバーにアクセスできる保証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/auth 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
}
```

注釈: 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` 列で指定された名前は `pg_hba.conf` ファイルに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:

```
host all all:::1/128 ident map=edbas
```

4. `MigrationToolkit` を呼び出す前に、`EDBPostgresAdvancedServer` サービスを再起動します。

5.1.1 Upgrading an RPM Installation

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ファイル。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
```

5.2 CentOS または RHEL7ppc64le ホストへの移行ツールキットのインストール

RPM パッケージを使用して、CentOS または RHELppc64le ホストに MigrationToolkit 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. AdvanceToolchain 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
```

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

Creating a Repository Configuration File

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想定して、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. 構成ファイルへの変更を保存した後、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. RHEL7ppc64le 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。

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

5. MigrationToolkit 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
```

5.3 Debian または Ubuntu ホストへの移行ツールキットのインストール

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は、username および 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。

- Debian9 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'
```

- Debian10 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'
```

3. 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
```

4. EBD 署名キーを追加しキー。

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

5. リポジトリのメタデータを更新します。

```
apt-get update
```

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

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

5.4 SLES ホストへの移行ツールキットのインストール

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 SLES12 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. MigrationToolkit 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、SUSEhost および SUSEPackageHub 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 MigrationToolkit 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>

5.5 Windows への MigrationToolkit のインストール

グラフィカルインストーラを使用して、Windows に MigrationToolkit 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。

注釈: このガイドでは、StackBuilder という用語を使用して、StackBuilderPlus (EDBPostgresAdvancedServer で配布) または StackBuilder (EnterpriseDB の PostgreSQL ワンクリックインストーラで配布) のいずれかを指します。

StackBuilder で MigrationToolkit のインストールを実行する前に、JavaJVM (バージョン 1.7.0 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、コマンド `SET PATH=javadir;%PATH%` を使用します

検索パスを設定したら、StackBuilder インストールウィザードを使用して MigrationToolkit を EDBPostgresAdvancedServer または PostgreSQL にインストールできます。

Java 実行可能ファイルは、検索パス (Windows では %PATH%、Linux/Unix では \$PATH) に存在する必要があります。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、StackBuilder インストールウィザードを使用して MigrationToolkit を EDBPostgresAdvancedServer または PostgreSQL にインストールできます。

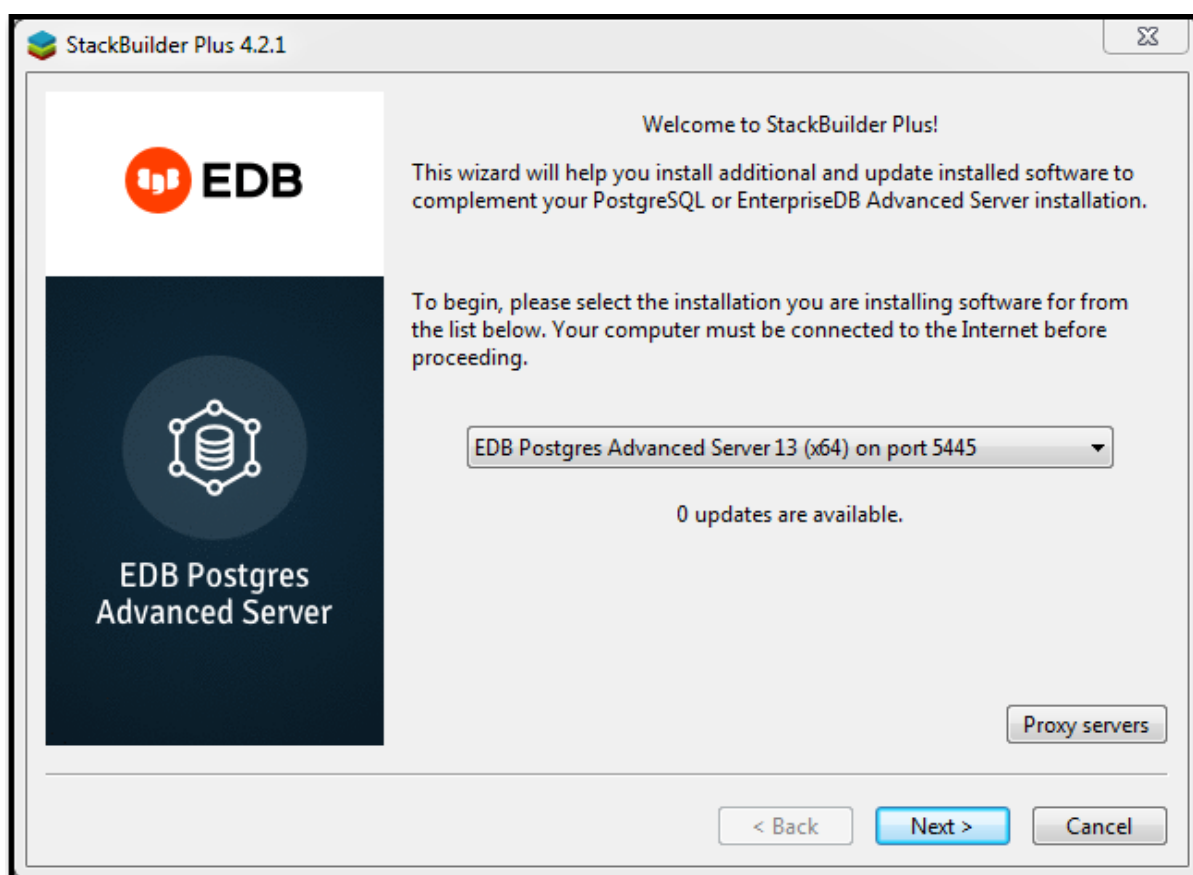
Launching StackBuilder Plus from EDB Postgres Advanced Server

既存の EDBPostgresAdvancedServer インストールから StackBuilderPlus を起動するには、Start (または Applications) メニューから EDBPostgres 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。

Launching Stack Builder from PostgreSQL

PostgreSQL インストールから StackBuilder 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、StackBuilder が開きます。

図 1: *StackBuilder* のようこそウィンドウ。

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。

StackBuilderPlus を使用して MigrationToolkit を EDBPostgresAdvancedServer インストールに追加する場合、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 の隣のボックスをオンにします。Next をクリックして続行します。

注釈: StackBuilder を使用して PostgreSQL インストールに MigrationToolkit を追加する場合は、ツリーコントロール (located under the Registration-required and trial productsnode) 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。

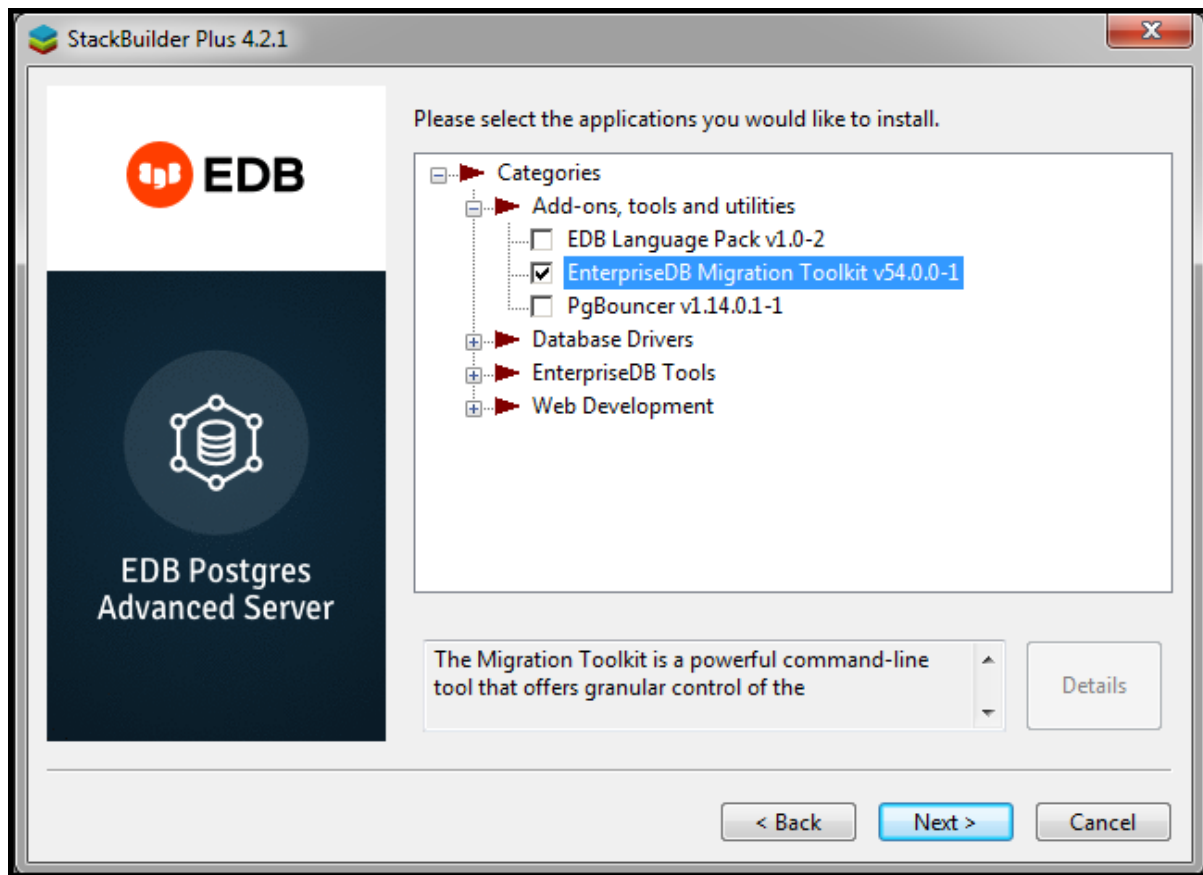
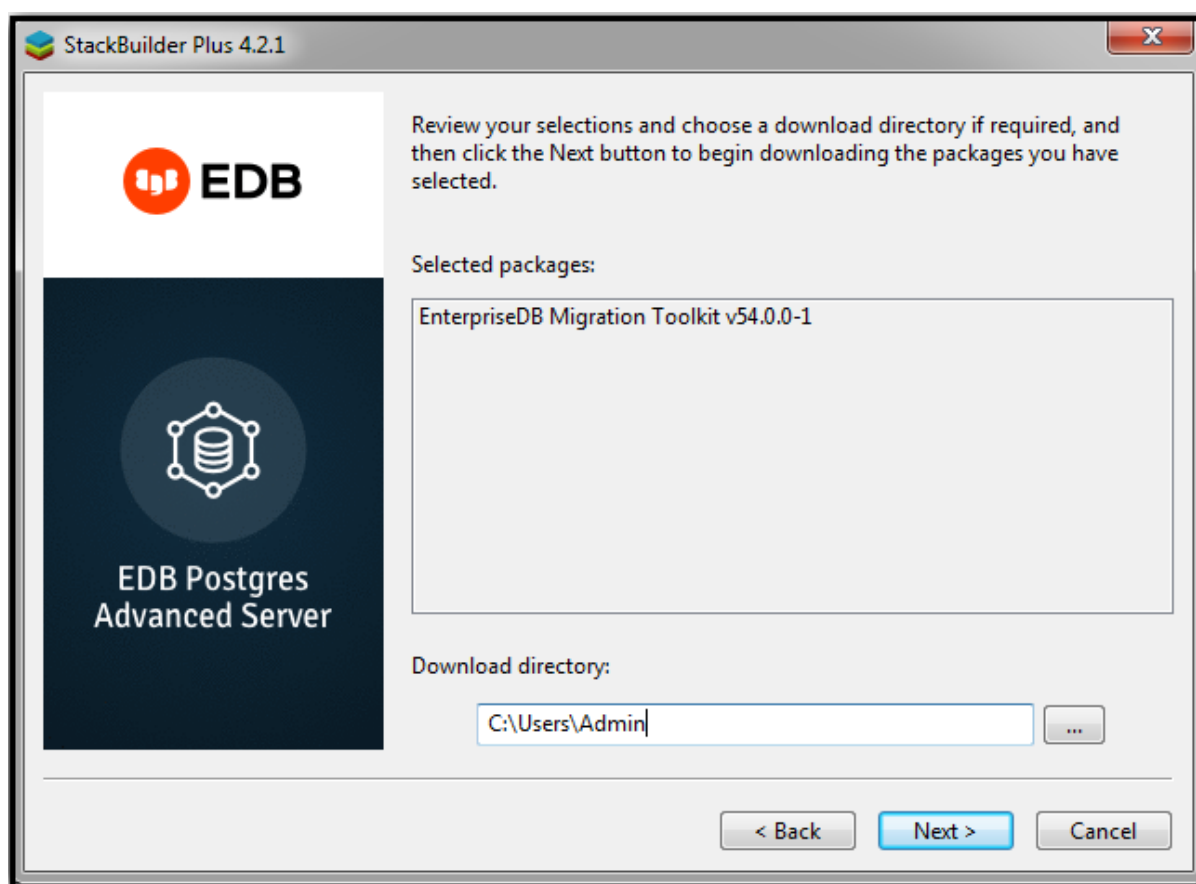


図 2: StackBuilder アプリケーション選択ウィンドウ

MigrationToolkit 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。

Next をクリックして、移行ツールキットのダウンロードをスタートします。

ダウンロードが完了すると、StackBuilder 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、MigrationToolkit インストールウィザードを開きます。

図 3: *StackBuilder* の選択確認ウィンドウ。

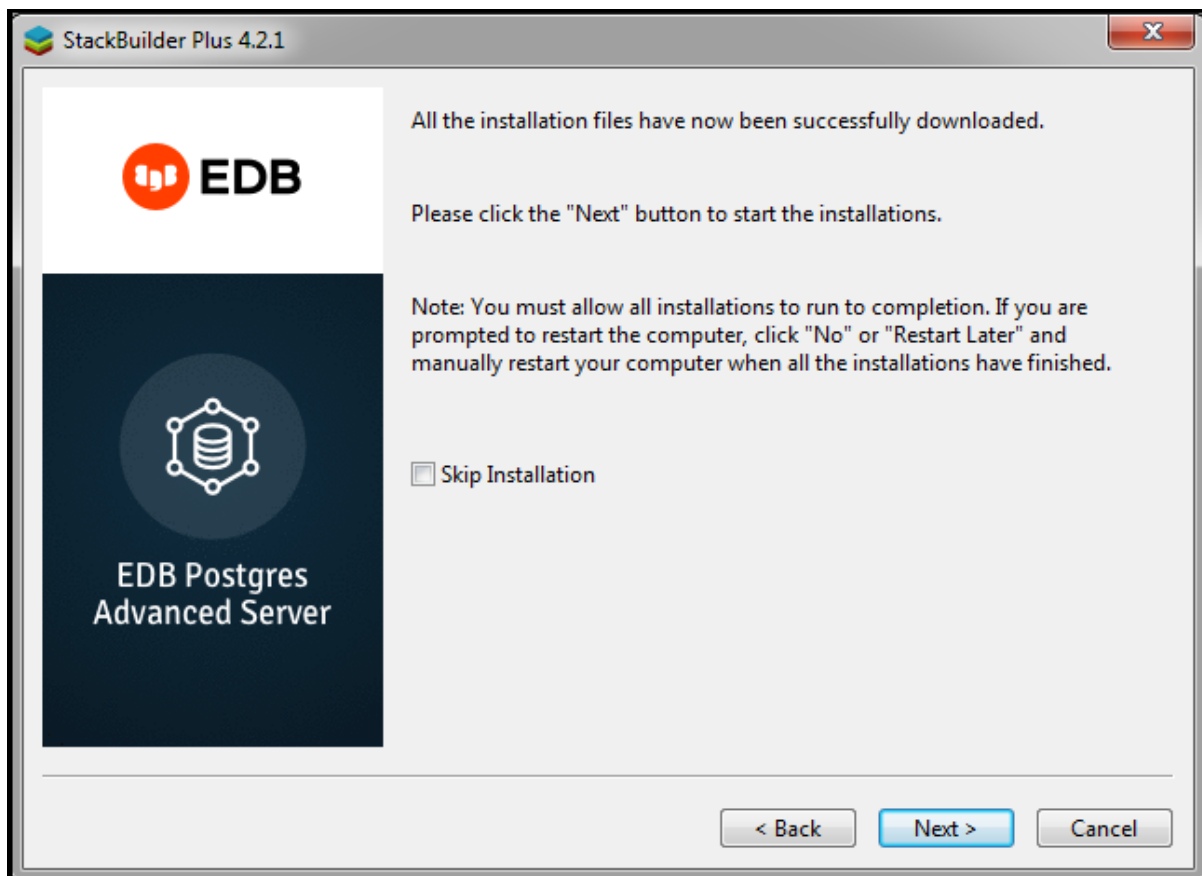


図 4: *StackBuilder* はファイルのダウンロードを確認します。

MigrationToolkit インストールウィザードの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。

MigrationToolkit 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。

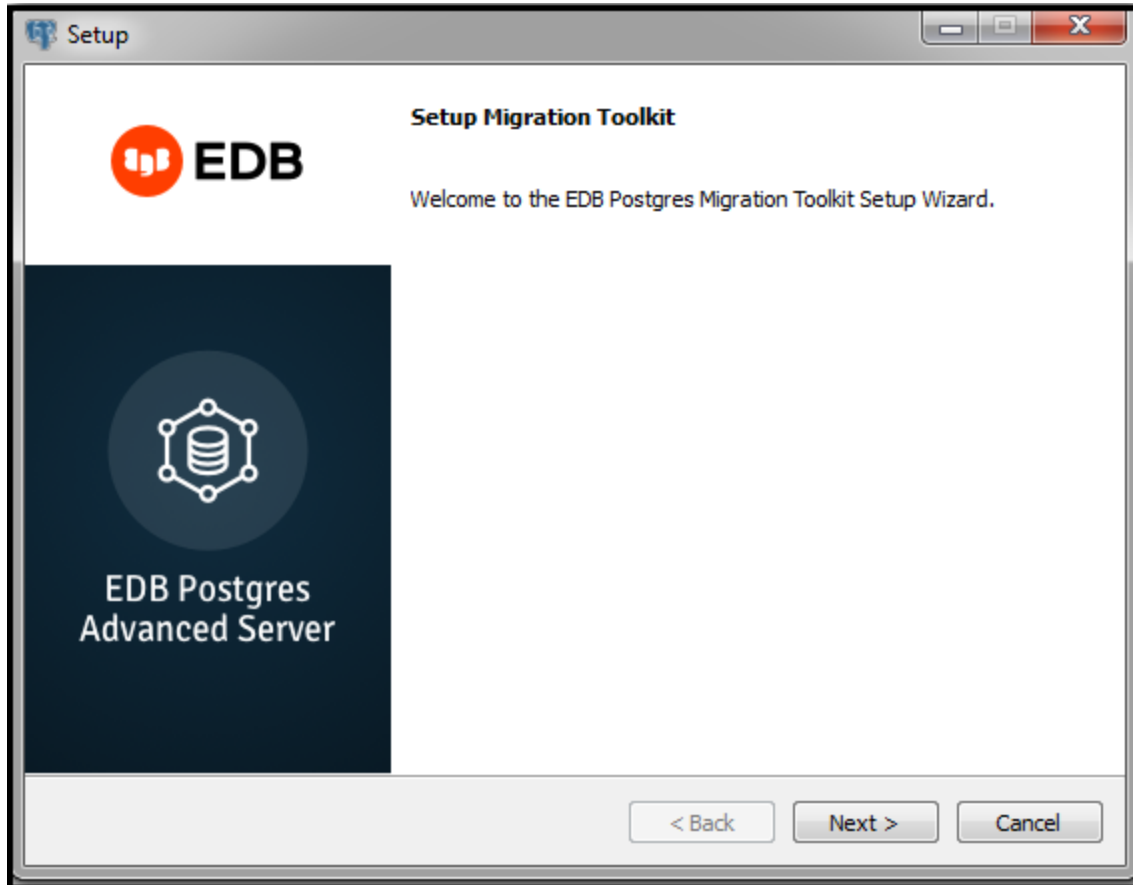


図 5: MigrationToolkit インストールウィザード。

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。

デフォルトでは、MigrationToolkit 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そのまま使用するか、ディレクトリを変更し、Next をクリックして続行します。

インストールウィザードは、Setup プログラムが MigrationToolkit 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。

ダイアログで、MigrationToolkit 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、MigrationToolkit インストーラを終了します。

StackBuilder 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 StackBuilder を閉じます。

スタックビルダで移行ツールキット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をインストールする必要があります。 [ソース固有のドライバのインストール](#) を参照してください。



図 6: EnterpriseDB ライセンス契約。

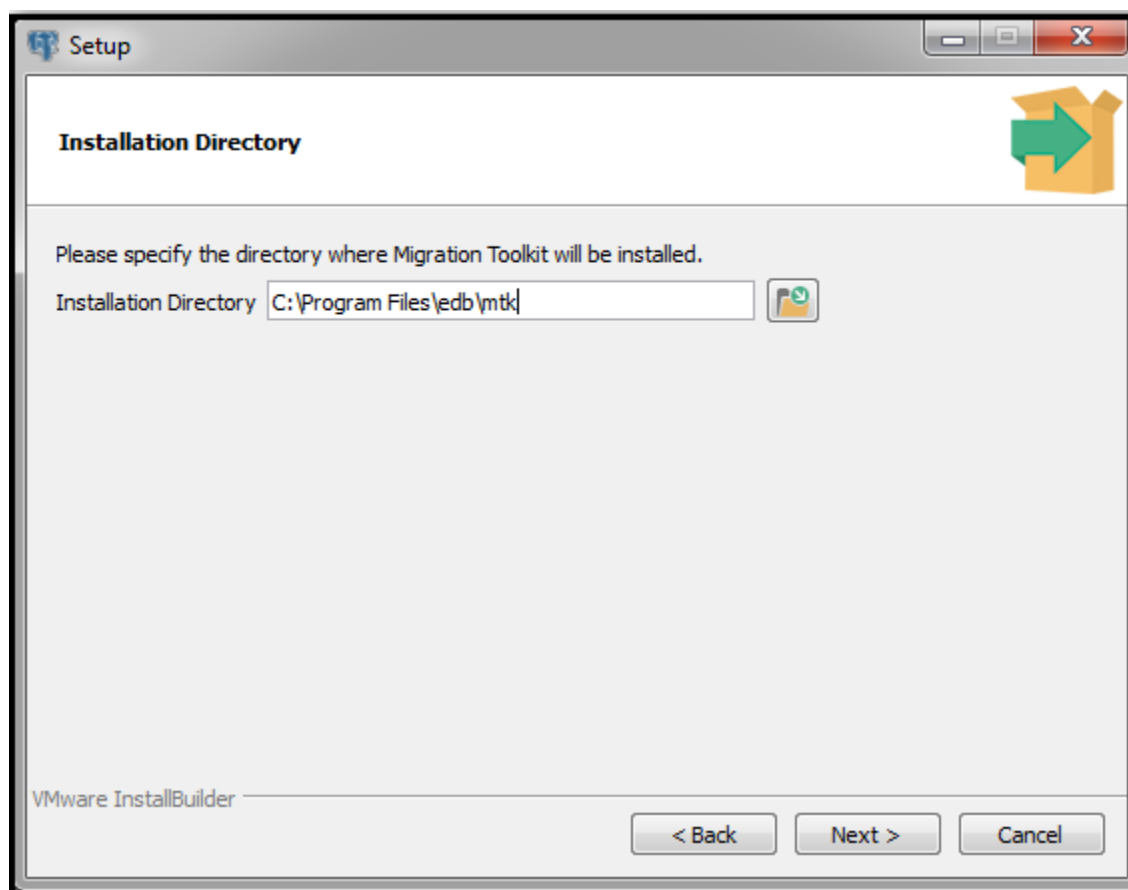


図 7: *MigrationToolkit* のインストールディレクトリを指定します。

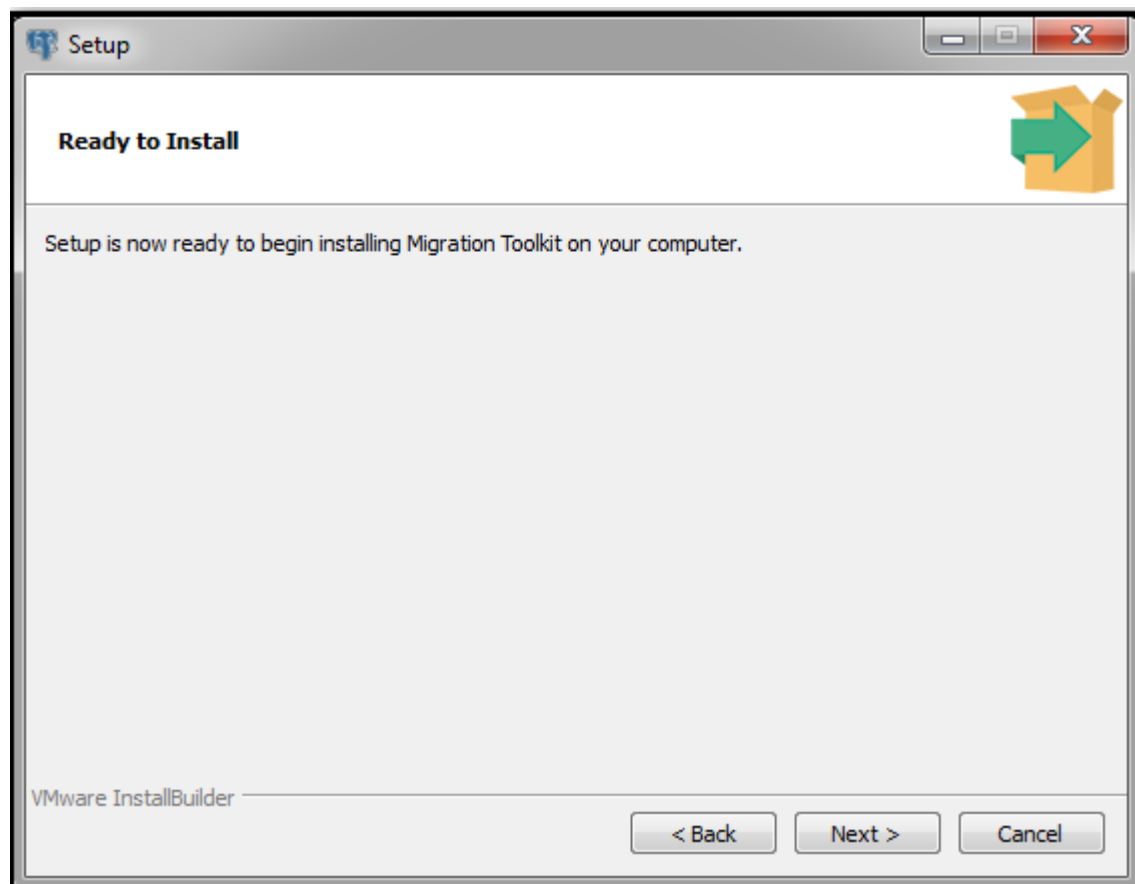


図 8: セットアップで *MigrationToolkit* のインストールをスタートする準備ができました。

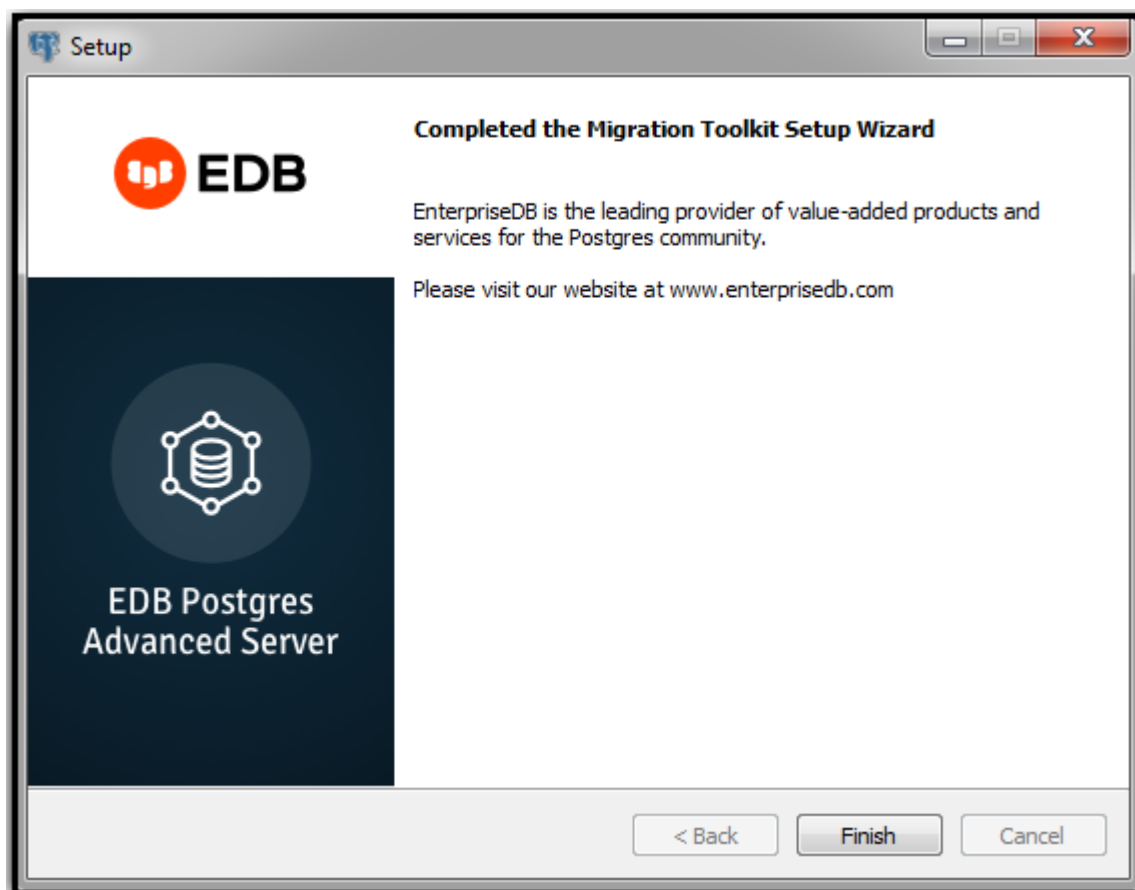


図 9: *MigrationToolkit* ウィザードは終了しました。

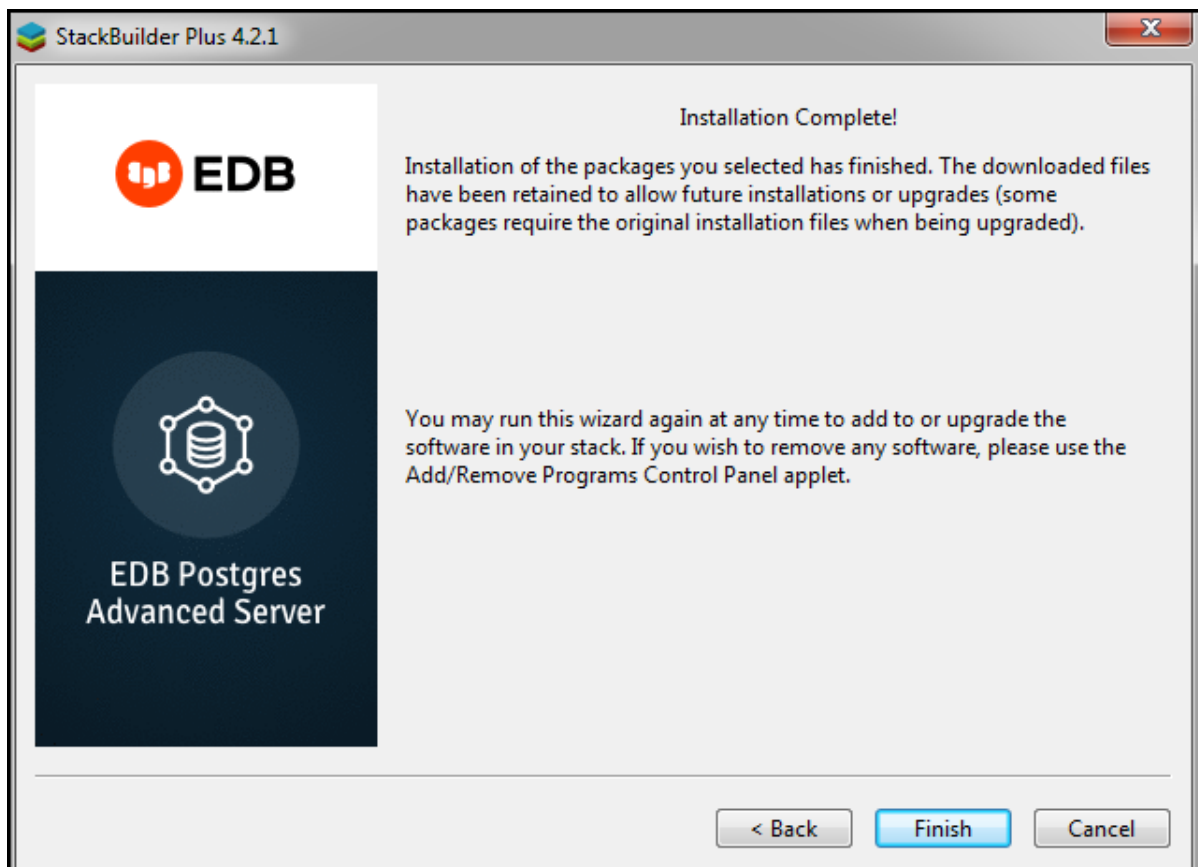


図 10: *StackBuilder* はインストールが完了したことを確認します。

5.6 MacOSX に MigrationToolkit 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、移行ツールキットを MacOSX に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:

- ターミナルのファイルパスに移動します。

```
/<ユーザー/ディレクトリ>/Downloads/edb-migrationtoolkit-54.0.0-1-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
```

- 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。
- MigrationToolkit 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。

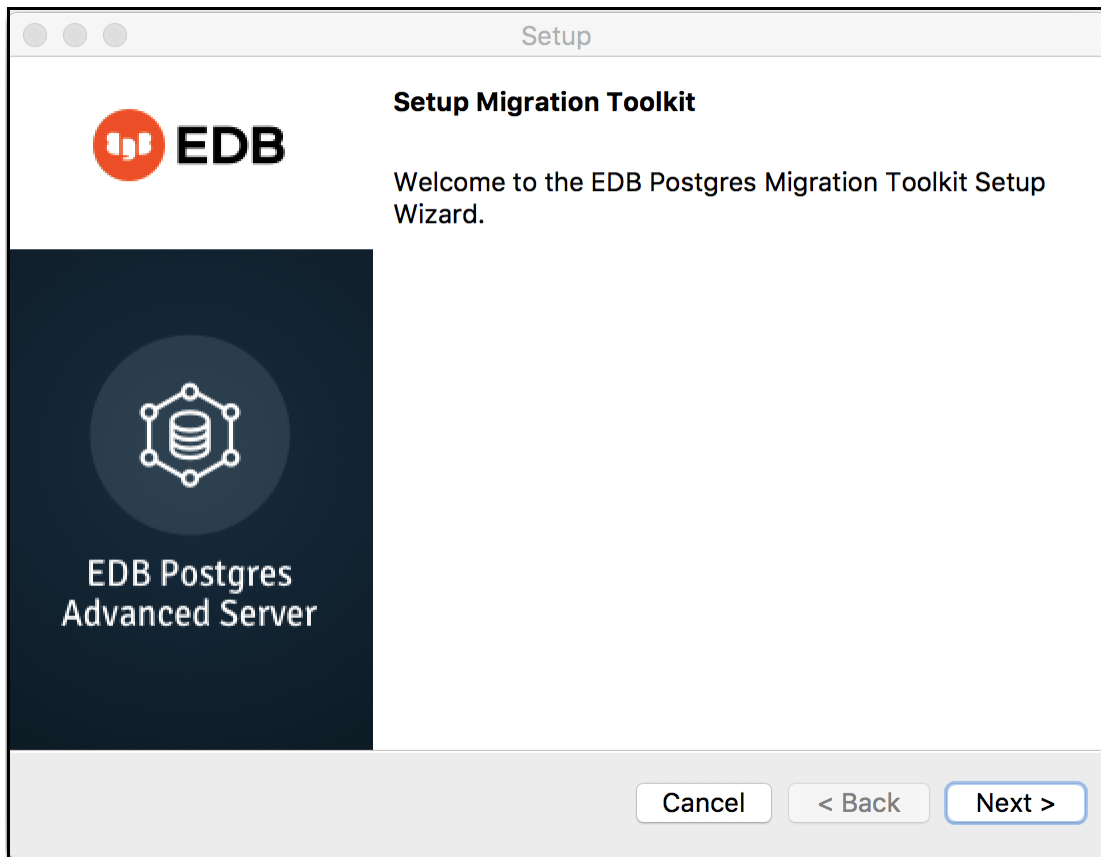
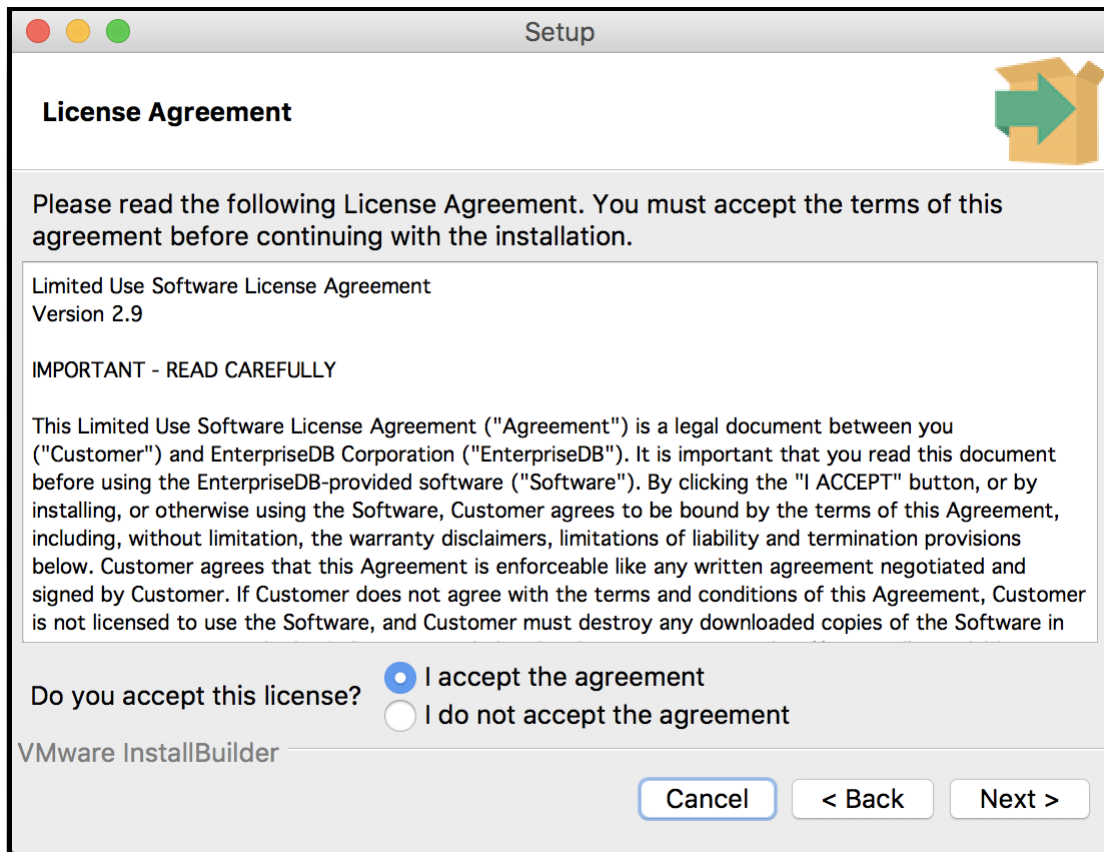


図 11: MigrationToolkit インストールウィザード。

- 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。

デフォルトでは、MigrationToolkit 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そのまま使用するか、ディレクトリを変更し、Next をクリックして続行します。

図 12: *EnterpriseDB* ライセンス契約。

インストールウィザードは、Setup プログラムが MigrationToolkit 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。

ダイアログで、MigrationToolkit 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、MigrationToolkit インストーラを終了します。

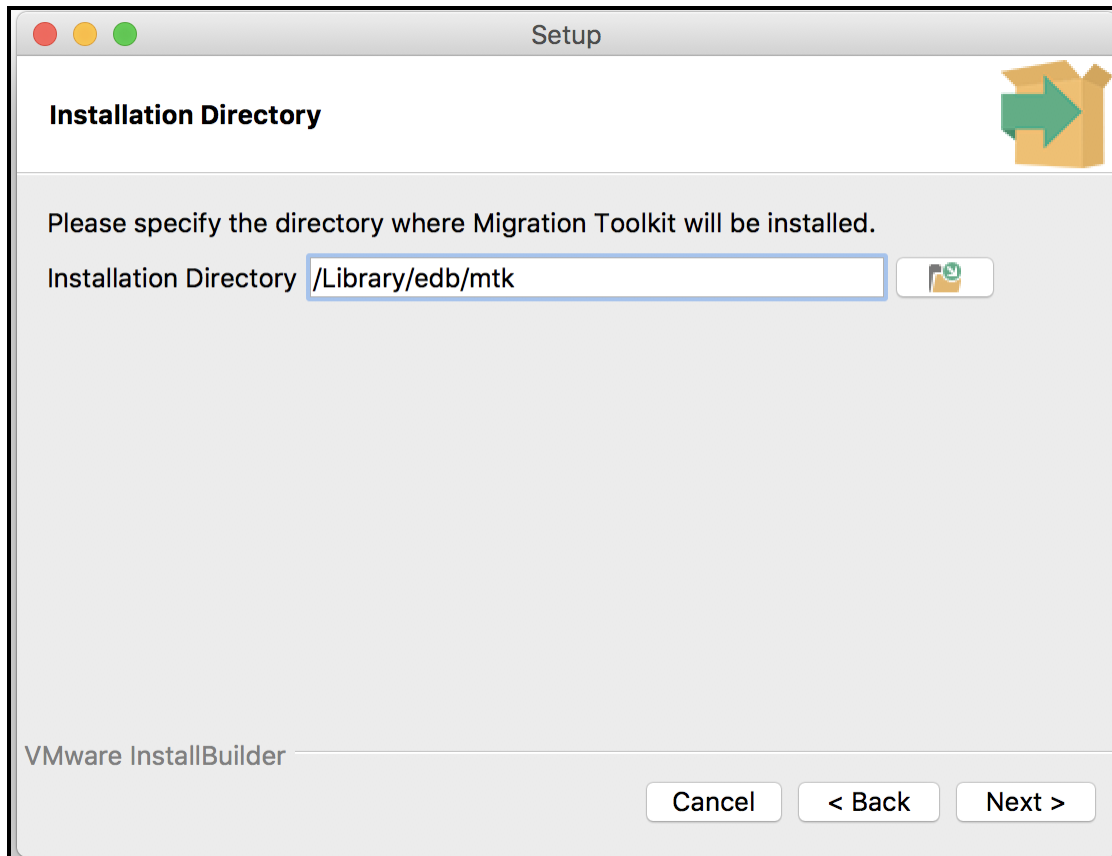


図 13: *MigrationToolkit* のインストールディレクトリを指定します。

5.7 ソース固有のドライバーのインストール

MigrationToolkit 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ページ。

<https://www.enterprisedb.com/advanced-downloads>

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。

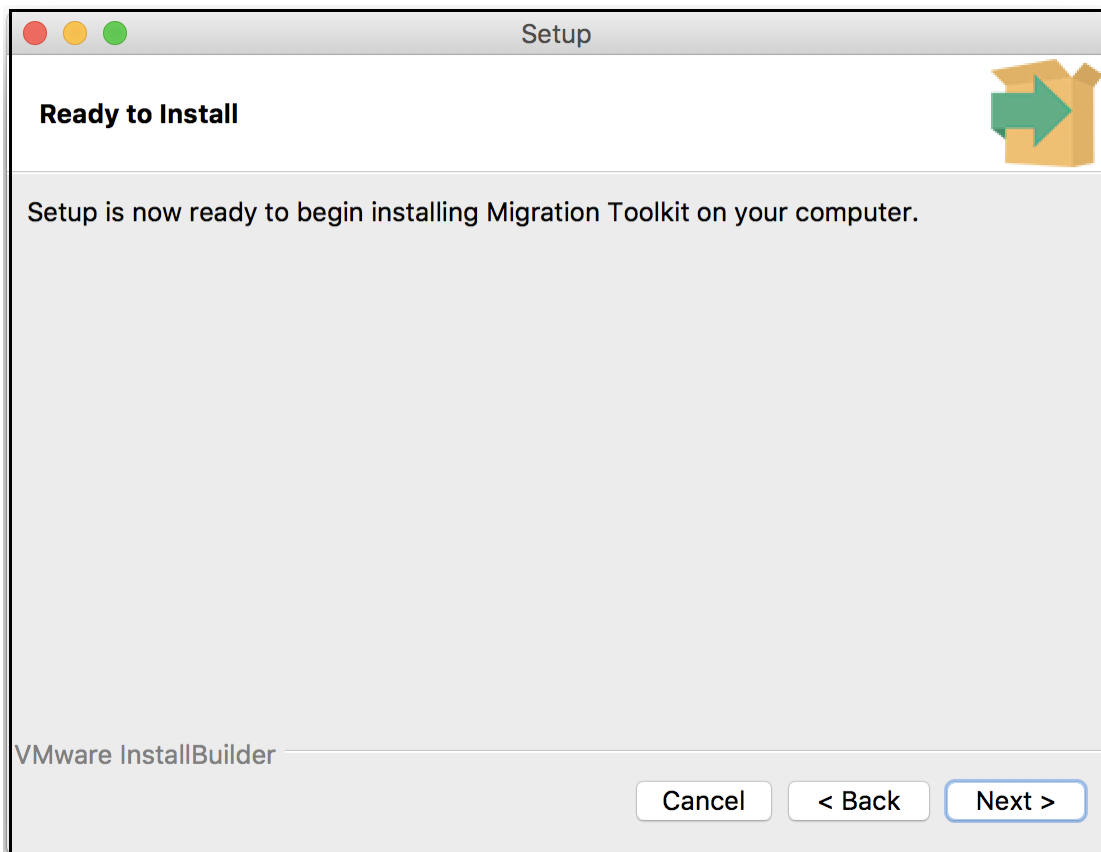


図 14: セットアップで *MigrationToolkit* のインストールをスタートする準備ができました。

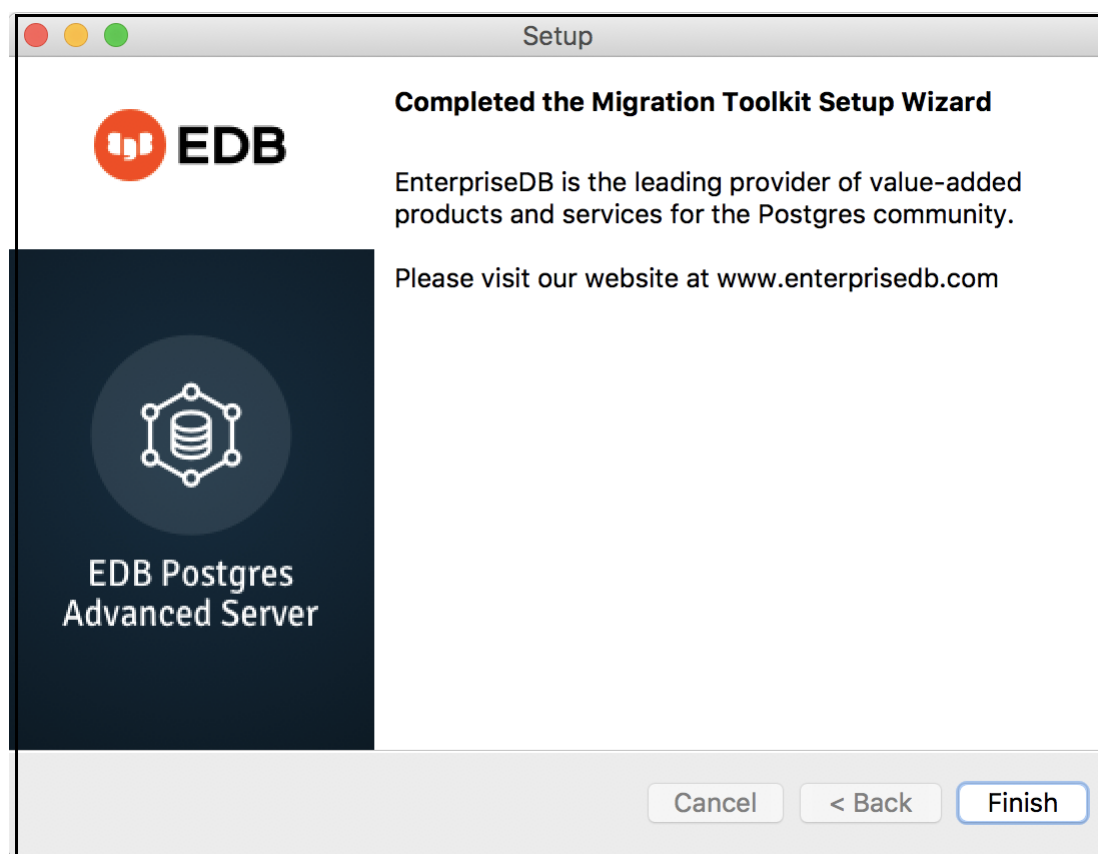


図 15: *MigrationToolkit* ウィザードは終了しました。

第6章 Building the toolkit.properties File

MigrationToolkit 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` ファイルは次の場所にあります。

```
/usr/edb/migrationtoolkit/etc
```

Windows では、ファイルは次の場所にあります。

```
C:/Program Files/edb/mtk/etc
```

サンプルの `toolkit.properties` ファイルを以下に示します。

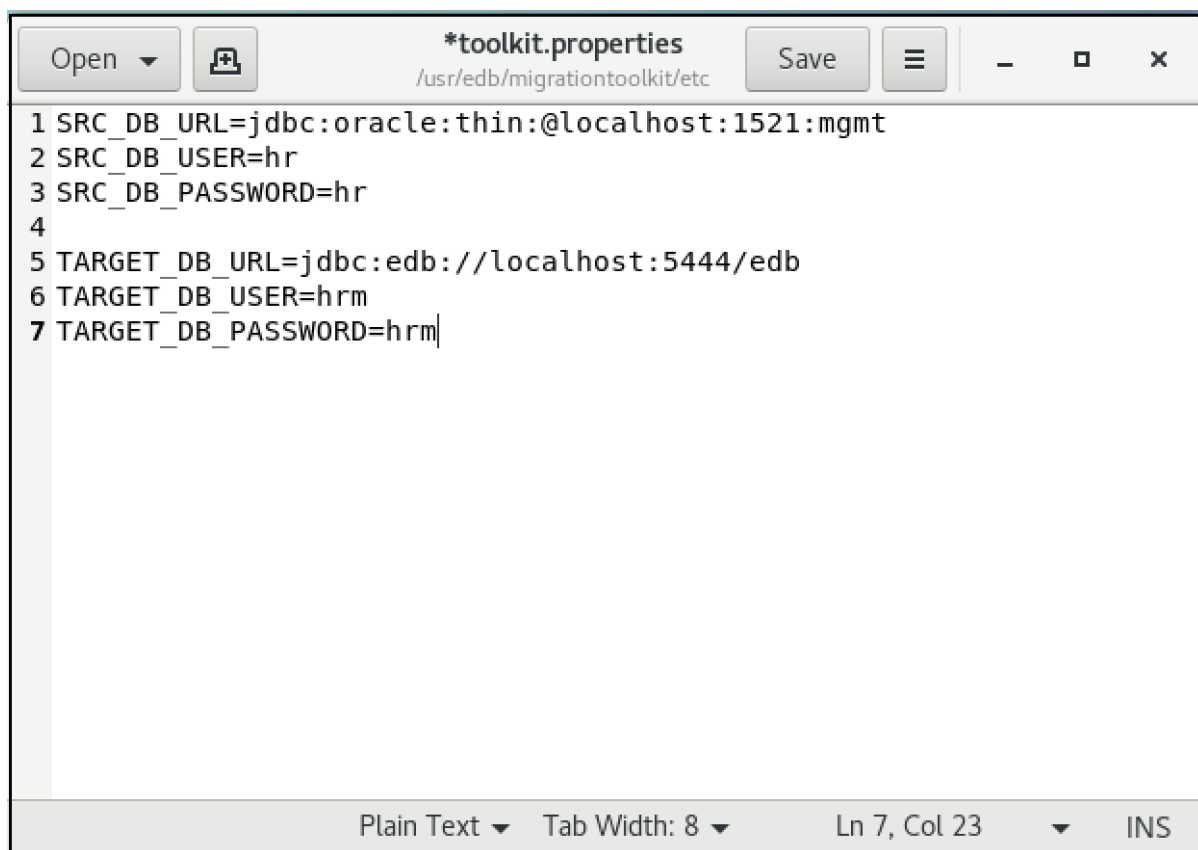


図 1: 典型的な `toolkit.properties` ファイル。

MigrationToolkit 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、MigrationToolkit がソースデータベースに接続する方法を指定します。URL の形成の詳細については、ソースデータベースに対応するセクションを参照してください。

- SRC_DB_USER は、ソースデータベース内の（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 JDBCURL を指定します。
- TARGET_DB_USER は、特権のある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。

6.1 EDBPostgresAdvancedServerURL の定義

MigrationToolkit は、次のプラットフォームから EDBPostgresAdvancedServer への移行を容易にします。

- オラクル
- MySQL
- Sybase
- SQLServer
- PostgreSQL

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

MigrationToolkit 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
```

EDBPostgresAdvancedServerURL には、次の情報が含まれています。

`jdbc`

プロトコルは常に `jdbc` です。

`edb`

EDBPostgresAdvancedServer 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`

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`

EDBPostgresAdvancedServer 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`

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}` は、移行される各タイプのオブジェクトの CREATE に対する特権を持つユーザを指定する必要があります。データを表に移行する場合、指定されたユーザは、各ターゲット表に対して INSERT、TRUNCATE、および REFERENCES 特権も必要とする場合があります。

`{TARGET_DB_PASSWORD|SRC_DB_PASSWORD}` は、特権 EDBPostgresAdvancedServer ユーザのパスワードに設定されます。

注釈: EDBPostgresAdvancedServer への移行中に、TARGET_DB_USER および TARGET_DB_PASSWORD に非スーパーユーザー資格情報を指定できます。ただし、dbLink_ora モジュール (`-copyViaDBLinkOra`) を介してユーザー、プロファイル、許可、およびコピーデータには、スーパーユーザの資格情報を使用する必要があります。

6.2 PostgreSQLURL の定義

MigrationToolkit は、次のプラットフォームから PostgreSQL への移行を容易にします。

- オラクル
- MySQL
- SQLServer
- EDBPostgresAdvancedServer

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

MigrationToolkit 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。

PostgreSQLURL 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`

プロトコルは常に `jdbc` です。

`postgresql`

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`

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`

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`

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}` は、移行される各タイプのオブジェクトの CREATE に対する特権を持つユーザを指定する必要があります。データを表に移行する場合、指定されたユーザは、各ターゲット表に対して INSERT、TRUNCATE、および REFERENCES 特権も必要とする場合があります。

`{SRC_DB_PASSWORD|TARGET_DB_PASSWORD}` は、特権 PostgreSQL ユーザのパスワードに設定されます。

注釈: EDBPostgresAdvancedServer への移行中に、TARGET_DB_USER および TARGET_DB_PASSWORD に非スーパーユーザー資格情報を指定できます。ただし、dbLink_ora モジュール (`-copyViaDBLinkOra`) を介してユーザー、プロファイル、許可、およびコピーデータには、スーパーユーザの資格情報を使用する必要があります。

6.3 オラクル URL の定義

MigrationToolkit は、オラクルデータベースから PostgreSQL または EDBPostgresAdvancedServer データベースへの移行を容易にします。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 JDBCURL 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`

プロトコルは常に `jdbc` です。

`oracle`

サブプロトコルは常に `oracle` です。

`thin`

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`

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`

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`

オラクルデータベースのデータベース SID。

`service_name`

オラクルサービスの名前。

`SRC_DB_USER` は、特権オラクルユーザの名前を指定する必要があります。オラクルユーザには、オブジェクトをオラクルから EDBPostgresAdvancedServer に移行するための DBA 権限が必要です。DBA<権限>が必要なデータベースオブジェクトのすべて移行さを保証するために、オラクル `GRANT DBA TO user` のコマンドを使用して、オラクルユーザに付与することができます。

`SRC_DB_PASSWORD` には、指定されたユーザのパスワードが含まれている必要があります。

6.4 MySQLURL の定義

MigrationToolkit は、MySQL データベースから EDBPostgresAdvancedServer または PostgreSQL データベースへの移行を容易にします。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 `JDBCURL` 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`

プロトコルは常に `jdbc` です。

`mysql`

サブプロトコルは常に `mysql` です。

`//host_name`

移行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]`

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`

ソースデータベースの名前。

`SRC_DB_USER` は、特権のある MySQL ユーザの名前を指定する必要があります。

`SRC_DB_PASSWORD` には、指定されたユーザのパスワードが含まれている必要があります。

6.5 SybaseURL の定義

MigrationToolkit は、Sybase データベースから EDBPostgresAdvancedServer データベースへの移行を容易にします。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 JTDSURL のフォームを取ります。例:

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

SybaseURL には次の情報が含まれます。

`jdbc`

プロトコルは常に `jdbc` です。

`jtds`

ドライバ名前は常に `jtds` です。

`sybase`

サーバータイプは常に `sybase` です。

`host_name`

移行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`

Sybase 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`

ソースデータベースの名前。

`SRC_DB_USER` は、特権 Sybase ユーザの名前を指定する必要があります。

`SRC_DB_PASSWORD` には、指定されたユーザのパスワードが含まれている必要があります。

6.6 SQLServerURL の定義

MigrationToolkit は、SQLServer データベースから PostgreSQL または EDBPostgresAdvancedServer データベースへの移行を容易にします。MigrationToolkit 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
- コンストレイン
- インデックス

MigrationToolkit 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。

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

```
jdbc:jtds:sqlserver://server[:port]/database_id
```

SQLServerURL には次の情報が含まれます。

`jdbc`

プロトコルは常に `jdbc` です。

`jtds`

ドライバ名前は常に `jtds` です。

`sqlserver`

サーバータイプは常に `sqlserver` です。

`server_name`

移行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`

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`

ソースデータベースの名前。

`SRC_DB_USER` は、特権 SQLServer ユーザの名前を指定する必要があります。

`SRC_DB_PASSWORD` には、指定されたユーザのパスワードが含まれている必要があります。

第7章 Invoking Migration Toolkit

MigrationToolkit をインストールし、*toolkit.properties* ファイル 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、MigrationToolkit は移行を実行する準備が整います。

MigrationToolkit 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: MigrationToolkit の呼び出し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許可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.
```

MigrationToolkit 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、*toolkit.properties* ファイルの所有者であり、ファイルの読み取りパーミッションが最低限必要ファイル。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 MigrationToolkit 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
```

Importing Character Data with Embedded Binary Zeros (NULL characters)

MigrationToolkit は、値が NULL の列のインポートを適切にサポートします。

ただし、MigrationToolkit は、JDBC 接続プロトコルした 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。

7.1 オラクルからのスキーマの移行

コマンドラインで指定されていない限り、MigrationToolkit 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 EDBPostgresAdvancedServer 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

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

注釈: 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、そのような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>のプロファイル DEFAULT と MONITORING_PROFILE を除外する。

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>
```

7.2 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 EDBPostgresAdvancedServer であると想定します。

MigrationToolkit 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 <db_type> -targetdbtype <target_type>
[options, ...] schema_name;
```

どこで:

-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 は大文字小文字ません。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。

から移行するには:	指定してください:
オラクル	oracle (デフォルト値)
MySQL	mysql
SQLServer	sqlserver
Sybase	sybase
PostgreSQL	postgres または postgresql
EDBPostgresAdvancedServer	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 は大文字小文字ません。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。

移行するには:	指定してください:
EDBPostgresAdvancedServer	enterprisedb
PostgreSQL	postgres または postgresql

schema_name

schema_name は、移行するソースデータベース (toolkit.properties ファイルで指定) 内のスキーマの名前です。少なくとも 1 つの schema_name 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 EDBPostgresAdvancedServer ホストに移行します。コマンドには - sourcedbtype および -targetdbtype オプションが含まれていることに注意してください。

```
$ ./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
```

MigrationToolkit 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て、MySQL データベースからマルチプルのスキーマを移行します。

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

Windows では、次のコマンドフォームを使用します。

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

第8章 Migration Toolkit Command Options

MigrationToolkit を実行するときに移行オプションして、移行の詳細を簡単に制御します。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
```

以下のセクションには、MigrationToolkit で動作する各コマンドオプションの参照資料が含まれており、ファレンス。オプションは、動作によってグループ化されます。次の表に、MigrationToolkit のオプションとそれらがグループ化されているセクションを示します。

機能:	関連オプション:
オフライン移行オプション	-offlineMigration
インポートオプション	-sourcedbtype、-targetdbtype、-schemaOnly、-dataOnly
スキーマ作成オプション	-dropSchema、-targetSchema
スキーマオブジェクト 選択オプション	-allTables、-tables、 -CONSTRAINTS、-ignoreCheckConstFilter、 -skipCKConst、-skipFKConst、 -skipColDefaultClause、 -indexes、-triggers、 -allViews、-views、 -allSequences、-sequences、 -allProcs、-procs、 -allFuncs、-funcs、 -checkFunctionBodies、 -allPackages、-パッケージソフト、 -allDomains、 -allQueues、-queues、 -allRules、
移行オプション	-truncLoad、-enableConstBeforeDataLoad、 -retryCount、-safeMode、-fastCopy、 -analyze、vacuumAnalyze、-replaceNullChar、 -copyDelimiter、-batchSize、 -cpBatchSize、-lobBatchSize、 -fetchSize、-filterProp -customColTypeMapping、-customColTypeMappingFile
オラクル固有のオプション	-allUsers、-users、 -allProfiles、-プロファイル、 -importPartitionAsTable、 -objectTypes、 -copyViaDBLinkOra、-allDBLinks -allSynonyms、-allPublicSynonyms、 -allPrivateSynonyms、-useOraCase、 -skipUserSchemaCreation
その他のオプション	-ヘルプ、-logdir、-logFileCount、-logFileSize、-logBadSQL-verbose、-バージョン

8.1 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、MigrationToolkit は*オフライン* 移行を実行します。オフライン移行中、MigrationToolkit 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。

注:次の例は、Linux で MigrationToolkit を呼び出す方法を示しています。Windows で MigrationToolkit 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` キーワードを指定し、その後に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、ユーザーの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、MigrationToolkit 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>
```

データのみのオフライン移行（スキーマオブジェクト定義を省略）を実行するには、MigrationToolkit 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>
```

ノート:-singleDataFile オプションは、プレーン SQL フォーマットで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。

8.1.1 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移行。次の例では<EDBPostgresAdvancedServer に保存されている（ACCTG*名前付けの）新しいデータベースに（*時間 名前付けの）スキーマを復元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る場合、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
```

8.2 インポートオプション

デフォルトでは、MigrationToolkit 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-PostgresAdvancedServer と想定しています。 - 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、MigrationToolkit はスキーマを移行するとき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 は、mysql、oracle、sqlserver、sybase、postgresql または enterprisedb のいずれかの値になります。 source_type は大文字小文字ません。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 は、enterprisedb、postgres、または postgresql の値のいずれかです。 target_type は大文字小文字ません。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。このオプションは、 - dataOnly オプションと併用できません。

-dataOnly

このオプションはデータのみをコピーします。 -tables オプションを使用すると、MigrationToolkit は選択したテーブルのデータのみをインポートします（以下の使用方法の詳細を参照）。このオプションは、 -schemaOnly オプションと併用できません。

8.3 スキーマ作成オプション

デフォルトでは、MigrationToolkit はソーススキーマオブジェクトまたはデータ、あるいはその両方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、MigrationToolkit は新しいスキーマを作成します。または、`-targetSchema option` を介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]
```

真に設定すると、MigrationToolkit は既存のスキーマ（およびそのスキーマ内のオブジェクト）を削除し、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。コマンドラインが `-targetSchema` オプションが含まれていない場合は、新しいスキーマの名前は、ソース・スキーマの名前と同じになります。

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、EDBPostgresAdvancedServer のメタデータストレージ用に予約しています。

8.4 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、テーブル名 (例: `-tables emp, dept, acctg`) のコンマ区切りリスト (スペース文字を挿入しない) です。

`-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、MigrationToolkit は、Sybase データベースからのチェックコンストレインとデフォルト句の移行を実装しません。Sybase データベースから移行コンストレインおよびデフォルト条項に `-constraints` パラメータを指定するとき `-ignoreCheckConstFilter` パラメータを含めます。

`-skipCKConst`

チェック制約の移行を省略しコンストレイン。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場合に役立ちます。

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にのみ有効です。

`-skipColDefaultClause`

列の DEFAULT 句の移行を省略し移行。

`-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*および*マテリアライズドビューの両方を移行することにノートしてください。(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、ビュー名 (たとえば、`-views all_emp, mgmt_list, acct_list`) のコンマ区切りリスト (スペース文字なし) です。

`-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、シーケンス名のコンマ区切りリスト (スペース文字を介在させない) 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、プロシージャ名のコンマ区切りリスト (スペース文字を介在させない) 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、ファンクション名のコンマ区切りリスト (間にスペース文字を入れない) 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 (ファンクションに前方参照が含まれる場合のエラーを回避するため)。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、パッケージ名のコンマ区切りリスト (間にスペース文字を入れない) です。

`-allDomains`

ソースデータベースからすべてのドメイン、列挙型、および複合型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。

`-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。EDBPostgresAdvancedServer 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 EDBPostgresAdvancedServer 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、キュー名のコンマ区切りリスト (間にスペース文字を入れない) 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` オプションも指定する必要があります。EDBPostgresAdvancedServer がソースデータベース場合、`-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 EDBPostgresAdvancedServer 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。

8.5 移行オプション

以下にリストされた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。

`-loaderCount [<value>]`

`-loaderCount` オプションを使用して、データのインポート時に **MigrationToolkit** が使用する必要のある並列スレッドの数を指定します。ソースデータベースは、<ラージ>のデータが含まれている場合、このオプションが特に有用であり、(移行ツールキットを実行している) **Postgres** のホストは、ハイエンド CPU と RAM リソースを有しています。value にはゼロ以外の正の数値を指定できますが、value は CPU コアの数を超えないようにすることをお勧めします。デュアルコア CPU では、2 の最適な value が必要です。

value の大きラージ指定すると、移行ツールキットは 'Out of heap space' エラーが発生、終了する可能性がありますのでごノートください。

`-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` オプションと組み合わせてのみ使用できます。

`-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、データ移行の前に、ターゲットテーブルのすべてのトリガー（データを個々のパーティションにリダイレクトするトリガーを含む）を有効にします。`-enableConstBeforeDataLoad` は、`-truncLoad` パラメータも指定されている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のために最初の移行試行中に移行に失敗したオブジェクトは、後続の移行中に正常に移行する場合があります。移行ツールキットは、最初の移行の試行中に失敗したオブジェクトを移行する **make** ます試行回数を指定するには、`-retryCount` オプションを使用します。0 より大きい value 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、**MigrationToolkit** 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、**MigrationToolkit** 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` オプションを使用することを選択した場合、移行が中断された場合、移行れたデータは（ターゲットデータベースで）回復できない可能性があります。

`-replaceNullChar <value>`

MigrationToolkit は、値が NULL の列のインポートを適切にサポートします。

ただし、MigrationToolkit 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 non-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。オブティマイザは、ANALYZE オペレーションによって収集された統計を参照し、情報を利用して効率的なクエリープランを構築します。

`-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。オブティマイザは、ANALYZE オペレーションによって収集された統計を参照し、情報を利用して効率的なクエリープランを構築します。VACUUM オペレーションは、ターゲットデータベース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ときに、コピー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'`（tab）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。Out of Memory 例外が発生した場合、`-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 MB 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、または 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 行ずつ実行されます。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のヒープスペース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の場合、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` が大きい場合は、Out of Memory 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、MySQLJDBC ドライバは、1つのネットワークラウンドトリップで、テーブルにあるすべての行をクライアントアプリケーション (MigrationToolkit) 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。'out of heap space' 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、MigrationToolkit にテーブルデータを 1 行ずつ強制的にロードさせます。

`-filterProp <file_name>`

`file_name` は、キー=`value` ペアの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、制約に対して評価されコンストレイン。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 `not` がスキーマ修飾されている必要があることにノートしてください。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>に次のコンストレインを含みます。

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

AR に等しくない `country_id` 値を持つ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 テーブルと DEPARTMENTS テーブルからのみ関連データを移行します。

```
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、列が想定するデータタイプの名前を指定します。セミコロンで区切られた `column_list` にマルチプルのペアを含めることができます。例、次のカスタムマッピングエントリを使用し、名前が終了 ID で INTEGER を入力するための任意の列をマップします:

```
.*ID=INTEGER
```

カスタムマッピングは、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予約、Linux では './.' を使用して '.' 文字を表します。例、EMP テーブルの EMP_ID 列から行を選択するために、カスタムマッピングを使用するために、次のカスタムマッピングエントリを指定します。

```
EMP//.EMP_ID=INTEGER
```

Windows、'/' 文字を使用して '.' 文字を表します。

```
EMP/.EMP_ID=INTEGER
```

```
-customColTypeMappingFile <property_file>
```

`property_file` にマルチブ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。キー=値 のペアで、ファイル内の各エントリーを個別の行に指定します。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、列が想定するデータタイプの名前を指定します。

8.6 オラクル固有のオプション

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 EDBPostgresAdvancedServer データベースに移行する場合にのみサポートされることにノートしてください。

`-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、ユーザ/ロール名（例:-`users MTK, SAMPLE, acctg`）のコンマ区切りリスト（間にスペース文字を入れない）です。`-users` オプションは、オラクルデータベースから EDBPostgresAdvancedServer データベースに移行する場合にのみサポートされることにノートしてください。

`-allProfiles`

すべてのカスタム（つまり、ユーザー作成）プロファイルをソースデータベースからインポートします。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の場合、プロファイルに関連付けられているだけで次のパスワードのパラメータ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。

- `allProfiles` オプションは、オラクルデータベースから EDBPostgresAdvancedServer データベースに移行する場合にのみサポートされることにノートしてください。

`-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、プロファイル名（例: `-profiles ADMIN_PROFILE, USER_PROFILE`）のコンマ区切りリスト（スペース文字を入れない）です。 `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ー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。

`-profiles` オプションは、オラクルデータベースから `EDBPostgresAdvancedServer` データベースに移行する場合にのみサポートされることにノートしてください。

`-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化されたテーブルの内容を単一のパーティション化されていないテーブルにインポートします。 `table_list` は、テーブル名（たとえば、`-importPartitionAsTable emp,dept,acctg`）のコンマ区切りリスト（間にスペース文字を入れない）です。

`-copyViaDBLinkOra`

`dblink_ora` モジュールは、SQL レベルで `EDBPostgresAdvancedServer` から Oracle への接続を提供します。 `dblink_ora` はバンドルされ、`EDBPostgresAdvancedServer` 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 メソッドよりもかなり高速です。

次の例では、`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 `EDBPostgresAdvancedServer` 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、「オラクル開発者ガイドのデータベース互換性ガイド」<<https://www.enterprisedb.com/docs/en/11.0/EPAS_Oracompat_Guide_v11/Database_Compatibility_for_Oracle_Developers_Guide.1.124.html#>>https://www.enterprisedb.com/docs/en/11.0/EPAS_Oracompat_Guide_v11/Database_Compatibility_for_Oracle_Developers_Guide.1.124.html#>>を参照してください<<https://www.enterprisedb.com/docs/en/11.0/EPAS_Oracompat_Guide_v11/Database_Compatibility_for_Oracle_Developers_Guide.1.124.html#>>。>>

を参照して<<https://www.enterprisedb.com/docs/en/11.0/EPAS_Oracompat_Guide_v11/Database_Compatibility_for_Oracle_Developers_Guide.1.124.html#>>。>>

`-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、指定されない限り、ダミーパスワード（`edb`）に置き換えられます。

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、`name=value` ペアのコンマ区切りリスト（スペース文字なし）を使用して、各データベースリンクのパスワードを指定できます。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。

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

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

MigrationToolkit 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 EDBPostgresAdvancedServer 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 EDBPostgresAdvancedServer 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 EDBPostgresAdvancedServer 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 EDBPostgresAdvancedServer 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。

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の大文字の命名規則は保持されます。これらのデータベースオブジェクトの場合、大文字の命名規則は a) データベースオブジェクトの名前、b) 列名、キー名、インデックス名、制約、等、表およびビューの、C) ビューの SELECT 列リスト、および d) 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。

注釈: 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内で、エラーなしでプログラムを適切に実行ために、識別子参照を手動で編集するオーダーがあります。このような修正は、発生した場合と発生していない場合が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。

注釈: `-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る場合があります。詳細については、このセクションの `-skipUserSchemaCreation` オプションの説明を参照してください。

MigrationToolkit のデフォルトの動作（`-useOraCase` オプションを使用しない場合）では、データベースオブジェクト名は引用符ずにオラクルから抽出されます（データベースオブジェクトがオラクルで引用符れて明示的に作成された場合を除く）。以下は、`-offlineMigration` オプションを使用して MigrationToolkit 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)
);
ALTER TABLE DEPT ADD CONSTRAINT DEPT_PK PRIMARY KEY (DEPTNO);
ALTER TABLE DEPT ADD CONSTRAINT DEPT_DNAME_UQ UNIQUE (DNAME);
```

このテーブルが EDBPostgresAdvancedServer に移行され、作成されると、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 EDBPostgresAdvancedServer 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)
```

EDBPostgresAdvancedServer 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が小文字になっているため、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");
```

このテーブルを EDBPostgresAdvancedServer に移行して作成すると、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 EDBPostgresAdvancedServer 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")
```

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 オプションが指定されている場合に、1つのスキーマ名前が小文字で、もう1つが大文字である場合を除き、同じ名前の2つのスキーマが作成されないようにオーダー場合に特に役立ちます。-useOraCase オプションを指定すると、MigrationToolkit が呼び出されたとき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、-useOraCase オプションが -skipUserSchemaCreation オプションなしで指定された場合、ターゲットデータベースは、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を持つことになります。-useOraCase オプションが -skipUserSchemaCreation オプションと共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。

8.7 その他のオプション

移行ツールキットのヘルプとバージョンをビューするには、以下に挙げる移行>を使用します。このセクションのオプションを使用して、MigrationToolkit 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>`

ログファイルが書き込まれる場所を指定するには、このオプションを含めます。 `log_path` は、アプリケーションログファイルが保存されるパスを表します。デフォルトでは、Linux ではログファイルは次の場所に書き込まれます。

`$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され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。値 0 を指定すると、ログファイルのローテーションが無効になり、単一のログファイルが作成されます (`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、このオプションを含めて、最大ファイルサイズ制限 (メガバイト単位) を指定します。 `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 MB 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るのと同じパスの下に保存され、 `schema_name` がスキーマの名前で、 `timestamp` が MigrationToolkit 実行のタイムスタンプである `mtk_bad_sql_<schema_name_timestamp>.sql` のフォーマットで名前付けます。

`-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、詳細は on です)。

`-version`

MigrationToolkit バージョンを表示します。

8.8 例

次の例は、オラクルから EDBPostgresAdvancedServer への移行の実行を示しています。

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、**MigrationToolkit** 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
```

(次のページに続く)

(前のページからの続き)

```

Loading Table: "MixedCase" ...
["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
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

```

(次のページに続く)

(前のページからの続き)

```
-----
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)
*****
```

スキップされた、サポートされていないデータベースオブジェクトの省略に注意してください。移行情報は、実行の最後に移行概要に要約されます。

第9章 Migration Errors

移行中、移行の概要は、移行の進捗状況が表示されます。エラーが発生した場合、概要には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。ログファイルのデフォルトの場所は次のとおりです。

Linux の場合:

```
$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
```

MigrationToolkit の `-logdir <log_path>` オプションを使用して、代替ログファイルディレクトリを指定できます。

9.1 接続エラー

MigrationToolkit は、`toolkit.properties` ファイル<`building_toolkit.properties_file`>の情報をを使用して、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。MigrationToolkit 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。次のセクションを使用して、一般的な接続エラーを特定し、それらを解決する方法を学習します。

9.1.1 無効なユーザー名/パスワード

MigrationToolkit 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` プロパティと `SRC_DB_PASSWORD` プロパティで移行を実行するのに十分な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` file を編集します。

9.1.2 接続が拒否されました:致命的:パスワード

私は移行ツールキットでの移行を実行しようとする*、私は次のエラーを取得します:*

```
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のに無効です。

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のに十分な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。

9.1.3 例外:ORA-28000:アカウントはロックされています

MigrationToolkit 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。

```
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 パラメーターと SRC_DB_PASSWORD パラメーターで移行を実行するための十分な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。

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

私は移行ツールキットでの移行を実行しようとする*、移行が失敗し、私はエラーメッセージます:

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

MigrationToolkit 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、[ソース固有のドライバーのインストール](#) を参照してください。

9.1.5 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

私は移行ツールキットでの移行を実行しようとする*、私は次のエラーを取得します:*

```
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 `*JDBCURL*` 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、 toolkit.properties ファイル<building_toolkit.properties_file>を編集して、```SRC_DB_URL``` 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。 SRC_DB_URL の値はデータベース固有であることにノートしてください。ソースデータベースタイプの詳細については、目次を参照してください。

9.1.6 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

私は移行ツールキットでの移行を実行しようとするとき、私は次のエラーを取得します:*

```
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たターゲットデータベース (EDBPostgresAdvancedServer) の JDBCURL 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、 toolkit.properties ファイルを編集して、 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ます。EDBPostgresAdvancedServer の JDBCURL の形成については、 [EDBPostgresAdvancedServerURL](#) の定義を参照してください。

9.2 移行エラー

MigrationToolkit 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場合があります。

9.2.1 エラー:予想される最後の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:

```
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。

注釈: 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 (|) が発生し、`-copyDelimiter '|'` オプションを使用して MigrationToolkit が実行され、エラーが発生します。

9.2.2 テーブルへのデータ読み込みエラー: **TABLE_NAME**

データのため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、データ移行を再試行します。

9.2.3 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:

```
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
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制約外部キー制約が含まれていません。MigrationToolkit は、外部キー制約を省略してテーブルを作成します。

MigrationToolkit 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。

9.2.4 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・データベース内の列には、移行されたデータを受信するために十分なラージではありません。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。

`Where: 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、移行を再試行し移行。

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、MySQLJDBC ドライバは、1 つのネットワークラウンドトリップで、テーブルにあるすべての行をクライアントアプリケーション (MigrationToolkit) 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10章 Error Codes

MigrationToolkit 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。

この章では、エラーコード、メッセージ、およびそれらの解決策について説明します。

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、エラークラス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。

エラークラス	説明
00	正常終了
01	情報
02	警告
03	一般的なエラー
0x	予約済み
10	無効なユーザー入力
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	手続き言語の移行
17	データの読み込み

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。テーブルの作成が原因 Postgres>データベースサーバ内の既存のテーブルに失敗した場合、エラーコード 42P07 は、データベースサーバによって返されます。MigrationToolkit ログの特定のエラーは DB-42P07 として表示されます。

10.1 エラーコードの概要

次のセクションでは、MigrationToolkit のエラーコードを要約します。次の表の「エラーコード」列には、MigrationToolkit エラーコードがリストされています。[メッセージと解像度] 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。このメッセージは、エラーの原因とその解決方法を説明しています。

[メッセージと解像度] 列の \$NAME は、実行時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。

10.1.1 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。

エラーコード	メッセージと解決策
MTK-02000	このクラスに関連し、特定のエラーコードバインディングを持たないすべての警告は、このエラーコードを使用します。
MTK-02001	「runMTK-ヘルプ」を実行して、使用方法の詳細を確認します。
MTK-02002	警告！オフライン移行パス*\$OFFLINE_PATH* は存在しません。スクリプトはユーザのホームフォルダーの下に作成されます。

10.1.2 クラス 10-無効なユーザー入力

このクラスは、無効なユーザ入力値を表します。

エラーコード	メッセージと解決策
MTK-10000	このクラスに関連し、特定のエラーコードバインディングを持たないすべてのエラーは、このエラーコードを使用します。
MTK-10001	information_schema、dbo、sys、または pg_catalog をターゲットスキーマとして選択することはできません。これらは、メタデータ情報を*\$DATABASE* に保存するために使用されます。
MTK-10002	「-schemaOnly」オプションと「-dataOnly」オプションを同時に指定することはできません。
MTK-10003	escapeTabDelimiter オプションを使用する場合、copyDelimiter として"/t" が必要です。
MTK-10004	-truncateLoad オプションは、-dataOnly オプションでのみ指定できます。
MTK-10005	'-dataOnly' オプションは、-allTables/-tables オプションにのみ適用可能です。このオプションが設定されている場合、スキーマ DDL はコピーできません。
MTK-10006	-constraints、-indexes と-triggers オプションは、-allTables/-tables オプションのコンテキストで適用されます。
MTK-10007	「-vacuumAnalyze」オプションと「-analyze」オプションを同時に指定することはできません。
MTK-10008	コンストレイン--skipFKConst オプションは与えられません。
MTK-10009	コンストレイン--skipCKConst オプションは与えられません。
MTK-10010	-schemaOnly オプションと一緒に-fastCopy オプションを指定することはできません。
MTK-10011	-skipColDefaultClause を-dataOnly オプションと一緒に指定することはできません。
MTK-10012	「-customColTypeMapping」オプションと「-customColTypeMappingFile」オプションを同時に指定することはできません。
MTK-10013	デフォルトの日付時刻は、「yyyy-MM-dd_HH:mm:ss」のフォーマットで指定する必要があります。時間部分はオプション。時間を指定するには、アンダースコア記号「_」が必要です。

次のページに続く

表 1 – 前のページからの続き

エラーコード	メッセージと解決策
MTK-10014	MySQL および Sybase を移行のターゲットデータベースとして選択することはできません。
MTK-10015	コピーデリミタは単一の文字でなければなりません。
MTK-10017	バッチサイズの数値が無効です。1～50000 の数値を使用してください。
MTK-10018	コピーバッチサイズは 0 より大きい必要があります。
MTK-10019	コピーバッチサイズの数値が無効です。0 より大きい数値を使用してください。
MTK-10020	フェッチサイズは 0 より大きい必要があります。
MTK-10021	フェッチサイズの数値が無効です。0 より大きい数値を使用してください。
MTK-10022	1 つ以上のコマンドライン引数が無効です。
MTK-10023	-targetDBVersion 値は、メジャー。/マイナー パターンを使用して指定する必要があります。
MTK-10024	-targetDBVersion は-offlineMigration モードでのみ使用できます。
MTK-10025	オプション (-constraints/-indexes/-triggers/-tables/-views/-sequences/-procs/-funcs/-packageソフト/-synonyms) マルチプルのスキーマオプションでは使用できません。
MTK-10026	-allSchemas オプションは、コマンドラインオプションリストの最後のオプションとして指定する必要があります。
MTK-10027	無効なコマンドライン引数を指定しました。「runMTK-ヘルプ」を実行して、使用方法の詳細を確認します。
MTK-10028	-replaceNullChar は、-schemaOnly オプションと一緒に指定できません。
MTK-10029	-nullReplacementChar は、-replaceNullChar オプションでのみ指定できます。
MTK-10030	constraints--ignoreCheckConstFilter だけで与えることができます。
MTK-10031	-enableConstBeforeDataLoad は、-truncLoad オプションでのみ指定できます。
MTK-10032	retryCount 値は 0 より大きい必要があります。
MTK-10033	retryCount の数値が無効です。0 より大きい数値を使用してください。
MTK-10034	loaderCount 値は 0 より大きい必要があります。
MTK-10035	loaderCount の数値が無効です。0 より大きい数値を使用してください。
MTK-10036	COPY フォーマットのオフラインデータ移行に singleDataFile オプションを使用できません。
MTK-10037	COPY フォーマットのオフラインデータ移行は、PostgreSQL または EnterpriseDB (EDBPostgresAdvancedServer) がターゲットデータベース場合にのみサポートされます。
MTK-10038	\$SCHEMA*は、*\$DATABASE のスキーマ名前として使用できません。-targetSchema オプションで別のスキーマ名前を選択します。
MTK-10039	ログファイルサイズは 0 より大きい必要があります。
MTK-10040	ログファイルの数は 0 以上必要があります。
MTK-10041	オフライン移行は、スキーマのみのオプションでのみ使用できます。
MTK-10042	copyViaDBLinkOra オプションは、オラクルから EnterpriseDB にデータをコピーする場合にのみ使用できます。
MTK-10043	-recreateConst オプションは、-dataOnly オプションでのみ指定できます。
MTK-10044	\$DATABASE データベースタイプは、MigrationToolkit でサポートされていません。有効なデータベースタイプ文字列（つまり、EnterpriseDB、Postgres、オラクル、SQLServer、Sybase、または MySQL）を指定します。

次のページに続く

表 1 – 前のページからの続き

エラーコード	メッセージと解決策
MTK-10045	<code>\$DATABASE</code> データベースに指定された URL は無効です。接続資格情報を確認してください。
MTK-10046	<code>-escapeKeywords</code> 値は、真または false です。
MTK-10047	<code>-useOraCase</code> オプションは、オラクルからの移行にのみ使用できます。
MTK-10048	Postgres のエクスポートされた <code>snapshotID</code> は無効です。
MTK-10049	オラクルデータベースに指定された URL は、 <code>dblink_ora</code> ではサポートされていません。接続資格情報を確認し、有効な URL を指定してください。
MTK-10050	スキーマ <code>*\$SCHEMA*</code> がソースデータベースに見つかりません。
MTK-10051	<code>skipUserSchemaCreation</code> オプションは、オラクルから EnterpriseDB へのユーザ移行にのみ使用できます。
MTK-10052	<code>-truncLoad</code> オプションは、オフラインデータ移行では使用できません。

10.1.3 クラス 11-構成の問題

このクラスは、`toolkit.properties` ファイルまたは `MigrationToolkit` で使用される他の構成ファイルで提供される無効な構成設定を表します。

エラーコード	メッセージと解決策
MTK-11000	このクラスに関連し、特定のエラーコードバインディングを持たないすべてのエラーは、このエラーコードを使用します。
MTK-11001	テーブルフィルター句を含むプロパティファイルを読み込めません。
MTK-11002	カスタムタイプマッピングファイルをロードできませんでした。理由:\$REASON。
MTK-11003	カスタムタイプマッピングファイルに無効なマッピングエントリーが含まれています。
MTK-11004	カスタムタイプマッピングファイルには、マッピングエントリーが含まれていません。
MTK-11005	ソースデータベースの接続情報は利用できません。
MTK-11006	ターゲットデータベースの接続情報は利用できません。
MTK-11007	ログディレクトリを作成できません。\$PATH。
MTK-11008	テーブル列フィルターリストを含むプロパティファイルを読み込めません。
MTK-11009	データベースの接続エラー*\$DATABASE*。
MTK-11010	\$DBLINKORA_FILE ファイルのローディングエラー。
MTK-11011	DBLinkOra モジュールのローディング中にエラーが発生しました。\$DBLINKORA_MODULE。ターゲットの EnterpriseDB サーバーに dblink_ora がインストール/構成されていることを確認します。dblink_ora モジュールのインストールと構成の詳細については、『オラクル開発者ガイドのデータベース互換性ガイド』を参照してください。
MTK-11012	指定された DBLink_Ora モジュールのローディング中にエラーが発生しました。\$DBLINKORA_MODULE。ターゲットの EnterpriseDB サーバーに dblink_ora がインストール/構成されていることを確認します。dblink_ora モジュールのインストールと構成の詳細については、『オラクル開発者ガイドのデータベース互換性ガイド』を参照してください。
MTK-11013	DBLinkOra モジュールをロードできませんでした。
MTK-11014	DBLinkOra への接続エラー。
MTK-11015	接続資格情報ファイル*\$TOOLKIT_PROP_FILE* は安全ではなく、グループ/他のユーザーはアクセスできません。このファイルにはプレーンパスワードが含まれており、MigrationToolkit の所有者ユーザのみに制限する必要があります。

10.1.4 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。

エラーコード	メッセージと解決策
MTK-12000	このクラスに関連し、特定のエラーコードバインディングを持たないすべてのエラーは、このエラーコードを使用します。
MTK-12001	権限が不十分なため、ユーザ/ロールの移行に失敗しました。オラクルカタログ DBA_ROLES、DBA_USERS、DBA_TAB_PRIVS、DBA_PROFILES、DBA_ROLE_PRIVS、ROLE_ROLE_PRIVS、DBA_SYS_PRIVS に対する SELECT 権限をユーザに付与します。
MTK-12002	権限が不十分なため、権限の移行に失敗しました。ユーザにオラクルカタログ dba_tab_privs に対する SELECT 権限します。

10.1.5 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の実装を提供していないか、MigrationToolkit がその移行をハンドリングしていません。

エラーコード	メッセージと解決策
MTK-13000	このクラスに関連し、特定のエラーコードバインディングを持たないすべてのエラーは、このエラーコードを使用します。
MTK-13001	オラクルがインデックスをスキップすると、同じ列リストに対するマルチプルインデックスは許可されません。
MTK-13002	<code>\$DATABASE</code> ビューの移行はサポートされていません。
MTK-13003	<code>\$DATABASE</code> ロールの場合、移行はサポートされていません。
MTK-13004	MySQL、SQLServer、Sybase データベースのトリガー、シーケンス、プロシージャ、関数、パッケージソフト、シノニムは移行できません。
MTK-13005	SQLServer データベースのシーケンスとパッケージソフトを移行することはできません。
MTK-13006	トリガーが移行されない場合、トリガーは別のユーザスキーマによって所有され、スキーマ間トリガーは EnterpriseDB でサポートされません。
MTK-13007	指定されたトリガーは移行されません。トリガーには、EnterpriseDB でサポートされていない WHEN 句があります。
MTK-13008	データベースリンク <code>*\$DATABASE_LINK*</code> をスキップしています。EnterpriseDB は現在、このタイプのデータベースリンクをサポートしていません。
MTK-13009	警告！トリガー <code>*\$TRIGGER*</code> の移行をスキップします。現在、非テーブルトリガーはターゲットデータベースではサポートされていません。
MTK-13010	プロシージャ、パッケージソフト、シノニム、およびデータベースリンクを PostgreSQL データベースに移行することはできません。
MTK-13011	ドメインオブジェクトは、ターゲットデータベースではサポートされていません。
MTK-13012	ルールは <code>*\$DATABASE*</code> データベースではサポートされていません。
MTK-13013	データベースの種類はサポートされていません。
MTK-13015	<code>\$TYPE</code> は COPY ではサポートされていません。
MTK-13016	PostgreSQL への移行は、オラクルまたは PostgreSQL がソースデータベース場合にのみサポートされます。
MTK-13017	グループは <code>*\$DATABASE*</code> データベースではサポートされていません。
MTK-13018	プロファイルの移行は、 <code>\$SRC_DB*</code> から <code>*\$TARGET_DB</code> への順列ではサポートされていません。
MTK-13019	不明なプロファイル <code>*\$PROFILE*</code> を移行できません。
MTK-13020	プロファイルをデータベースバージョン <code>*\$VERSION*</code> に移行できません。
MTK-13021	パスワードプロファイル確認ファンクション <code>*\$MY_VERIFICATION_FUNCTION*</code> は明示的に移行する必要があります。
MTK-13022	アドバンスト・Queues の移行は、 <code>\$の SRC_DB*</code> データベース <code>*\$TARGET_DB</code> へ データベースからサポートされていません。
MTK-13023	アドバンスト Queues は、データベースバージョン <code>*\$VERSION*</code> に移行できません。
MTK-13024	INTERVAL パーティションは <code>*\$DATABASE*</code> ではサポートされていません。テーブルは INTERVAL 定義なしで移行され定義。
MTK-13025	警告！ユーザープロファイルの移行は、ターゲットデータベースバージョン <code>*\$VERSION*</code> ではサポートされていません。ユーザ <code>/'\$USER/'</code> のプロファイル <code>/'\$PROFILE'</code> はスキップされます。
MTK-13026	<code>\$SRC_DB*</code> から <code>*\$TARGET_DB</code> へのオブジェクトタイプの移行はサポートされていません。

10.1.6 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。

エラーコード	メッセージと解決策
MTK-14000	このクラスに関連し、特定のエラーコードバインディングを持たないすべてのエラーは、このエラーコードを使用します。
MTK-14001	ソース*\$DATABASE* データベースに 1 つ以上のテーブルがありません。
MTK-14002	ソース*\$DATABASE* データベースに 1 つ以上のテーブルが見つかりませんでした。-tables モードでは、大文字と小文字を区別しない限り、テーブル名前は大文字にする必要があります。
MTK-14003	ソース*\$DATABASE* データベースに 1 人以上のユーザーが見つかりませんでした。-users モードでは、大文字と小文字を区別しない限り、ユーザ名前は大文字にする必要があります。

10.1.7 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。

エラーコード	メッセージと解決策
MTK-15000	このクラスに関連し、特定のエラーコードバインディングを持たないすべてのエラーは、このエラーコードを使用します。
MTK-15001	制約*\$CONSTRAINT* の作成エラー。
MTK-15002	ロール*\$ROLE* の作成中にエラーが発生しました。
MTK-15003	\$OBJECT*の特定の権限*\$PRIVILEGE*を*\$USER に付与する際にエラーが発生しました。
MTK-15004	\$USER*への*\$ROLE の付与エラー。
MTK-15005	権限*\$REASON* の付与エラー。
MTK-15006	ユーザ*\$USER* の作成中にエラーが発生しました。
MTK-15007	インデックス*\$INDEX* の作成中にエラーが発生しました。
MTK-15008	ビュー*\$VIEW* の作成中にエラーが発生しました。
MTK-15009	マテリアライズドビュー*\$MVIEW* の作成中にエラーが発生しました。
MTK-15010	パブリックシノニム*\$SYNONYM* の作成中にエラーが発生しました。
MTK-15011	プライベートシノニム*\$SYNONYM* の作成中にエラーが発生しました。
MTK-15012	シーケンス*\$SEQUENCE* の作成中にエラーが発生しました。
MTK-15013	テーブル*\$TABLE* の作成中にエラーが発生しました。
MTK-15014	データベースリンク*\$DATABASE_LINK* の作成中にエラーが発生しました。
MTK-15015	指定されたオブジェクトタイプ*\$OBJECT_TYPE* の作成中にエラーが発生しました。
MTK-15016	テーブル*\$TABLE*を*\$DATABASE* データベースに作成できませんでした。
MTK-15017	リンクされたスキーマ*\$LINKED_SCHEMA* はターゲットデータベースに存在しません。スキーマを作成してからリトライします。
MTK-15018	ドメイン*\$DOMAIN* の作成中にエラーが発生しました。
MTK-15019	カスタムデータタイプ*\$DATA_TYPE* の作成中にエラーが発生しました。
MTK-15020	グループ*\$GROUP* のメンバーシップの作成中にエラーが発生しました。
MTK-15021	テーブル名前*\$TABLE* にはスキーマ修飾子がありません。マルチプルのスキーマ移行コンテキストでは、各テーブルをスキーマ修飾する必要があります。
MTK-15022	スキーマ修飾子*\$SCHEMA* は、スキーマリストとマッチしません。
MTK-15023	テーブルのメタデータ情報は利用できません。
MTK-15024	テーブルリストはまだ初期化されていません。
MTK-15025	データベースメタデータ情報の取得中にエラーが発生しました。
MTK-15026	グループ*\$GROUP* の作成中にエラーが発生しました。
MTK-15027	プロファイル*\$PROFILE* の作成中にエラーが発生しました。

10.1.8 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。

エラーコード	メッセージと解決策
MTK-16000	このクラスに関連し、特定のエラーコードバインディングを持たないすべてのエラーは、このエラーコードを使用します。
MTK-16001	トリガー*\$TRIGGER* の作成中にエラーが発生しました。
MTK-16002	ルール*\$RULE* の作成エラー。
MTK-16003	パッケージ仕様*\$PACKAGE* の作成中にエラーが発生しました。
MTK-16004	パッケージ本体*\$PACKAGE* の作成中にエラーが発生しました。
MTK-16005	パッケージボディが無効です。スキップしています... Note: このエラーメッセージは、パッケージ仕様が正常に移行されたが、ソースデータベースに対応するパッケージボディがない場合にも表示されます。この場合、エラーメッセージされても、ターゲットデータベースでパッケージ仕様が適切にファンクションする必要があります。
MTK-16006	プロシージャ*\$PROCEDURE* の作成中にエラーが発生しました。
MTK-16007	ファンクション*\$FUNCTION* の作成中にエラーが発生しました。
MTK-16008	匿名ブロック*\$BLOCK* の作成エラー。
MTK-16009	キュー*\$QUEUE* の作成中にエラーが発生しました。

10.1.9 クラス 17-データの読み込み

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。

エラーコード	メッセージと解決策
MTK-17000	このクラスに関連し、特定のエラーコードバインディングを持たないすべてのエラーは、このエラーコードを使用します。
MTK-17001	テーブルへのデータ読み込みエラー:\$TABLE
MTK-17002	エンコード変換では、代わりに一括挿入を使用して読み込まれるいくつかの文字が検出されました。
MTK-17003	コピーテーブル*\$REASON* のエラー。
MTK-17004	無効なデータタイプ。
MTK-17005	このテーブルには、一括挿入ロード用にマークされた CLOB データが含まれます。

第11章 Unsupported Features

EDBPostgresAdvancedServer 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。MigrationToolkit 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、MigrationToolkit 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、EDBPostgresAdvancedServer 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。EDBPostgresAdvancedServer データベースには、かなり以前からテキストサーチのサポートが含まれていましたが、実装は Oracle のものとはかなり異なります。MigrationToolkit は、この機能を利用するオブジェクトを移行できません。

EDBPostgresAdvancedServer がまだサポートしていない機能もあります。このカテゴリの機能には、自動ストレージ管理、テーブル圧縮、外部テーブルが含まれます。多くの場合、成功する回避策を調整できます。

- 自動ストレージ管理は、(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。
- テーブルの圧縮は、圧縮された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で実装できます。
- 外部テーブルは EDBPostgresAdvancedServer には存在しませんが、データベース内のステー징テーブルにフラットテキストファイルをロードできます。EDBPostgresAdvancedServer データベース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。
- オラクルデータベースから EDBPostgresAdvancedServer にマルチプルのプロファイルを移行する場合、移行の完了時にユーザ（または複数のユーザー）にプロファイルを手動で割り当てる必要があります。

11.1 サポートされていない **Postgres** 機能

MigrationToolkit 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、次の URL で入手可能な PostgreSQL コア文書を参照してください。

<https://www.postgresql.org/docs/current/indexes-opclass.html>

11.2 よくある質問

Does Migration Toolkit support the migration of packages?

MigrationToolkit は、オラクルデータベースから EDBPostgresAdvancedServer へのパッケージソフトの移行をサポートしています。EDBPostgresAdvancedServer が提供する移行サポートについては、[機能概要セクション](#)をご覧ください。

Is there a way to transfer only the data from the source database?

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。

Does Migration Toolkit support migration of tables that contain data of the CLOB data type?

MigrationToolkit 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移行。

Does EDB Postgres Advanced Server support the enum data type?

EDBPostgresAdvancedServer は現在 enum データタイプをサポートしていませんが、将来のリリースでそれらをサポートします。それまでは、検査制約を使用して、EDBPostgresAdvancedServer 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、列が取り得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（サポート）に含まれていない値を挿入しようとする、EDBPostgresAdvancedServer はエラーをスローします。

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

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

Does EDB Postgres Advanced Server support materialized views?

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。現在、自動クエリー書き換えは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。

When I try to migrate from a MySQL database that includes a TIME data type, I get the following error:Error Loading Data into Table:Bad format for Time. Does Postgres support MySQL ``TIME`` data

types?

時間コンポーネントの値が 24 より大きいである限り、Postgres は TIME データ型の保存に問題はありません。

Postgres とは異なり、MySQL の TIME データタイプでは、TIME または INTERVAL の値を表す値を保存できます。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、その値がレンジ外であると認識した場合、エラーが結果れます。

第12章 Conclusion

EDBMigrationToolkit ガイド

Copyright © 2007-2021EnterpriseDBCorporation。

禁無断転載

EnterpriseDB34CrosbyDrive, Suite201, Bedford, MA01730, USA

T+17813573390F+19784671307Einfo@enterprisedb.comwww.enterprisedb.com

- EnterpriseDB および PostgresEnterpriseManager は、EnterpriseDBCorporation の登録商標です。EDB および EDBPostgres は、EnterpriseDBCorporation の商標です。オラクルは、オラクル Inc の登録商標です。その他の商標は、それぞれの所有者の商標です。
- EDB は、存在する場合、EDBPostgres 製品のログオン UI の設計、コーディングのベストプラクティスの確立、レビュー、および入力バリデーションを行います。EDB は追加の入力コンポーネントに対して同じアプローチに従いますが、製品の性質上、限定されたバリデーションが可能な信頼できるユーザーによって入力および送信されるフリーフォーム SQL、WMI またはその他の文字列を受け入れる必要がある場合があります。このような場合、ユーザーが誤った入力や危険な入力を禁止することはできません。
- EDB は、フリーフォーム SQL または将来的に認証され、信頼されたユーザーからの他の潜在的に危険な入力を受け入れるの製品に機能を追加する権利を留保しますが、彼らは最低限のリスクの可能性を提供確保する保証に設計され、テストされているようなすべての機能を保証し、どこ可能であれば、スーパーユーザまたは同等の特権が必要です。
- EDB は、すべての潜在的な脅威を予測できることを保証しないため、すべての潜在的な脆弱性が対処または検討されたことを完全に保証することはできません。

索引

- Building the toolkit.properties File, 37
- CentOS Installation, 13
- CentOS ppcle Installation, 16
- Conclusion, 93
- Configuring IDENT Authentication, 15
- Connection Errors, 73
- Debian Installation, 18
- Defining a MySQL URL, 44
- Defining a PostgreSQL URL, 41
- Defining a SQL Server URL, 46
- Defining a Sybase URL, 45
- Defining an Advanced Server URL, 39
- Defining an Oracle URL, 43
- Error Codes, 79
- Executing Offline Migration Scripts, 54
- FAQ, 91
- Functionality Overview, 11
- Import Options, 55
- Installing Migration Toolkit, 13
- Installing Source-Specific Drivers, 34
- Invoking Migration Toolkit, 47
- Linux Installation, 13
- MacOSX Installation, 26
- Migration Errors, 73, 76
- Migration Methodology, 7
- Migration Options, 60
- Migration Toolkit Command Options, 51
- Offline Migration Options, 53
- Oracle Specific Options, 64
- RHEL Installation, 13
- RHEL ppcle Installation, 16
- Schema Creation Options, 56
- Schema Object Selection Options, 57
- SLES Installation, 19
- Supported Operating Systems and Database Versions, 5
- Ubuntu Installation, 18
- Unsupported Features, 89
- Unsupported PostgreSQL Features, 90
- Upgrading an RPM Installation, 16
- What's New, 3
- Windows Installation, 20