



EDB

Migration Toolkit

Release 55.2.0

EnterpriseDB

Jan 04, 2022

CONTENTS

1	Supported operating systems and database versions	3
1.1	Database versions	3
1.2	Supported operating systems versions	4
2	Migration methodology	5
2.1	The migration process	5
2.2	Connecting an application to Postgres	6
3	Functionality overview	9
3.1	Object migration support	10
3.2	Online migration versus offline migration	11
4	Installing Migration Toolkit	13
4.1	Installing Migration Toolkit on a CentOS or RHEL 7 and 8 Host	13
4.1.1	Upgrading an RPM Installation	16
4.2	Installing Migration Toolkit on a CentOS, or RHEL 7 ppc64le Host	16
4.3	Installing Migration Toolkit on a Debian or Ubuntu Host	17
4.4	Installing Migration Toolkit on a SLES Host	18
4.5	Installing Migration Toolkit on Windows	19
4.6	Installing Migration Toolkit on Mac OS X	30
4.7	Installing Source-Specific Drivers	35
5	Building the toolkit.properties file	37
5.1	Defining an Advanced Server URL	38
5.2	Defining a PostgreSQL URL	39
5.3	Defining a BigAnimal URL	40
5.4	Defining an Oracle URL	41
5.5	Defining a MySQL URL	42
5.6	Defining a Sybase URL	43
5.7	Defining a SQL Server URL	44
6	Invoking Migration Toolkit	47
6.1	Importing character data with embedded binary zeros (NULL characters)	47
6.2	Migrating a schema from Oracle	48
6.3	Migrating from a non-Oracle source database	48
7	Migration Toolkit command options	51
7.1	Offline migration options	52
7.1.1	Executing offline migration scripts	53
7.2	Import options	53
7.3	Schema creation options	54

7.4	Schema object selection options	54
7.5	Migration options	56
7.6	Migration options for parallel data loading	58
7.6.1	High-level overview of the parallel data-loading algorithm	59
7.6.2	System resource recommendations and constraints**	60
7.7	Oracle-specific options	60
7.8	Miscellaneous options	64
7.9	Example	65
8	Migration errors	69
8.1	Connection errors	69
8.1.1	Invalid username/password	69
8.1.2	Connection rejected: FATAL: password	70
8.1.3	Exception: ORA-28000: the account is locked	70
8.1.4	Exception: oracle.jdbc.driver.OracleDriver	70
8.1.5	I/O exception: The Network Adapter could not establish the connection	70
8.1.6	Exception: The URL specified for the “target” database is invalid	71
8.2	Migration errors	71
8.2.1	ERROR: Extra data after last expected column	71
8.2.2	Error loading data into table: <i>TABLE_NAME</i>	71
8.2.3	Error creating constraint <i>CONS_NAME_FK</i>	72
8.2.4	Error Loading Data into Table	72
9	Error codes	75
9.1	Error code summary	75
9.1.1	Class 02 - Warning	76
9.1.2	Class 10 - Invalid user input	76
9.1.3	Class 11 - Configuration issues	76
9.1.4	Class 12 - Insufficient privileges	77
9.1.5	Class 13 - Unsupported features	77
9.1.6	Class 14 - Missing objects	78
9.1.7	Class 15 - Schema migration	79
9.1.8	Class 16 - Procedural language migration	80
9.1.9	Class 17 - Data loading	80
10	Unsupported features	81
10.1	Unsupported Postgres features	81
10.2	Frequently asked questions	82
11	Release Notes	83
11.1	Version 55.2.0	83
11.2	Version 55.1.0	84
11.3	Version 55.0.0	85

Migration Toolkit is a powerful command-line tool that offers granular control of the migration process. Migration Toolkit helps with migrating database objects and data to an EDB Postgres Advanced Server or PostgreSQL database from:

- Oracle
- MySQL
- Microsoft SQL Server

You can also use Migration Toolkit to migrate database objects and data from Sybase Adaptive Server Enterprise to EDB Postgres Advanced Server or between EDB Postgres Advanced Server and PostgreSQL.

You can install Migration Toolkit with the EDB Postgres Advanced Server installer or by using Stack Builder. Stack Builder is distributed with both the EDB Postgres Advanced Server and the PostgreSQL one-click installer available from the [EnterpriseDB website](#).

!!! Note * The term Postgres in this documentation refers to either an installation of EDB Postgres Advanced Server or PostgreSQL. The term Stack Builder refers to either StackBuilder Plus (distributed with EDB Postgres Advanced Server) or Stack Builder (distributed with the PostgreSQL one-click installer from EnterpriseDB).

SUPPORTED OPERATING SYSTEMS AND DATABASE VERSIONS

1.1 Database versions

You can use the following database product versions with Migration Toolkit:

- PostgreSQL versions 10, 11, 12, 13, and 14
- EDB Postgres Advanced Server versions 10, 11, 12, 13, and 14
- Oracle 10g Release 2
- Oracle 11g Release 2
- Oracle 12c Release 1
- Oracle 18c Release 1
- Oracle 19c Release 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

Contact your EnterpriseDB Account Manager or sales@enterprisedb.com if you require support for other database products.

!!! Note * The Migration Toolkit isn't tested with and doesn't officially support use with Oracle Real Application Clusters (RAC) and Exadata. However, Migration Toolkit might work when it is connected to a single persistent node. For more information, contact your EDB representative.

1.2 Supported operating systems versions

- Centos **or** RHEL **or** OL **7 and 8**
- CentOS **or** RHEL PPCLE **7**
- SLES **12** SP5
- Debian **9 and 10**
- Ubuntu **18.04 and 20**
- Windows (64-bit), **2016, and 2019**
- Mac OS X **10.12+**

!!! Note * The ojdbc7.jar can cause large data migrations to be incomplete due to a limit on records over Integer.MAX_VALUE (2147483647) while fetching from the ResultSet. To avoid this issue, we recommend upgrading to ojdbc8.jar.

MIGRATION METHODOLOGY

You might consider migrating from one database to another for many reasons. Migration can allow you to take advantage of new or better technology. If your current database doesn't offer the right set of capabilities to allow you to scale the system, moving to a database that offers the functionality you need is the best move for your company.

Migration can also be cost effective. Migrating systems with significant maintenance costs can save money spent on system upkeep. By consolidating the number of databases in use, you can also reduce in-house administrative costs. By using fewer database platforms (or possibly taking advantage of database compatibility), you can do more with your IT budget.

Using more than one database platform can offer you a graceful migration path if a vendor raises their pricing or changes their company directive. EnterpriseDB has, for years, helped companies migrate their existing database systems to Postgres.

2.1 The migration process

The migration path to Postgres that we recommend includes the following main steps:

1. Start the migration process by determining the database objects and data to include in the migration. Form a migration team that includes someone with solid knowledge of the architecture and implementation of the source system.
2. Identify potential migration problems. If it is an Oracle-to-EDB Postgres Advanced Server migration, consult the [EnterpriseDB Postgres Advanced Server documentation](#) for complete details about the compatibility features supported in EDB Postgres Advanced Server. Consider using the EnterpriseDB migration assessment service to help with this review.
3. Prepare the migration environment. Obtain and install the needed software, and establish connectivity between the servers.
4. If the migration involves a large body of data, consider migrating the schema definition before moving the data. Verify the results of the DDL migration and resolve any problems reported in the migration summary. [Migration errors](#) includes information about resolving migration problems.
5. Migrate the data. For small data sets, use Migration Toolkit. If it's an Oracle migration into EDB Postgres Advanced Server, and the data set is large or if you notice slow data transfer, take advantage of one of the other data movement methods available:
 - – Use the EDB Postgres Advanced Server database link feature compatible with Oracle databases. - If your data has BLOB or CLOB data, use the `dblink_ora` style database links instead of the Oracle style database links.
 - Both of these methods use the Oracle Call Interface (OCI) to connect to Oracle. After connecting, use an SQL statement to select the data from the 'linked' Oracle database and insert the data into the EDB Postgres Advanced Server database.

6. Confirm the results of the data migration, and resolve any problems reported in the migration summary.
7. Convert applications to work with the newly migrated Postgres database. Applications that use open standard connectivity such as JDBC or ODBC normally require changes only to the database connection strings and selection of the EnterpriseDB driver. See *Connecting an application to Postgres* for more information.
8. Test the system performance, and tune the new server. If you're migrating into an EDB Postgres Advanced Server database, take advantage of EDB Postgres Advanced Server's performance tuning utilities:
 - – Use Dynatune to dynamically adjust database configuration resources. – Use Optimizer Hints to direct the query path. – Use the ANALYZE command to retrieve database statistics.
 - See [EDB Postgres Advanced Server](#) and [Database Compatibility for Oracle Developers](#) for information about the performance tuning tools available with EDB Postgres Advanced Server.

2.2 Connecting an application to Postgres

To convert a client application to use a Postgres database, you must modify the connection properties to specify the new target database. In the case of a Java application, change the JDBC driver name (`Class.forName`) and JDBC URL.

A Java application running on Oracle might have the following connection properties:

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

Modify the connection string to connect to a Postgres server:

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

Converting an ODBC application to connect to an instance of Postgres is a two-step process.

1. To connect an ODBC application, use an ODBC data source administrator to create a data source that defines the connection properties for the new target database.
 - Most Linux and Windows systems include graphical tools that allow you to create and edit ODBC data sources. After installing ODBC, check the **Administrative Tools** menu for a link to the ODBC Data Source Administrator. Select **Add** to start the Create New Data Source wizard. Then, complete the dialogs to define the new target data source.
2. Change the application to use the new data source.
 - The application contains a call to `SQLConnect` or `SQLDriverConnect`. Edit the invocation to change the data source name. In the following example, the data source is named `OracleDSN`:

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

To connect to an instance of Postgres defined in a data source named PostgresDSN, change the data source name:

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

After establishing a connection between the application and the server, test the application to find any compatibility problems between the application and the migrated schema. In most cases, a simple change resolves any incompatibility that the application encounters. When a feature is not supported, use a workaround or third-party tool to provide the functionality required by the application. See [Migration errors](#), for information about some common problems and their workarounds.

FUNCTIONALITY OVERVIEW

Migration Toolkit is a powerful command-line tool that offers granular control of the migration process. Using Migration Toolkit is a two-step process:

1. Edit the `toolkit.properties` file to specify the source and target database.
2. Invoke Migration Toolkit at the command line, specifying migration options.

Migration Toolkit helps with migration of database objects and data to an EDB Postgres Advanced Server or PostgreSQL database from:

- Oracle
- MySQL
- SQL Server

Migration Toolkit also allows you to migrate database objects and data to an EDB Postgres Advanced Server database from Sybase. You can also use Migration Toolkit to migrate between EDB Postgres Advanced Server and PostgreSQL. Migration Toolkit includes a number of options, allowing you granular control of the migration process:

- Use the ``-safeMode`` option to commit each row as it is migrated.
- Use the ``-fastCopy`` option to bypass WAL logging to optimize migration.
- Use the ``-batchSize`` option to control the batch size of bulk inserts.
- Use the ``-cpBatchSize`` option to specify the batch size used with the COPY command.
- Use the ``-lobBatchSize`` option to specify the batch size used for large object data, [↪ types](#).
- Use the ``-filterProp`` option to migrate only those rows that meet a condition you, [↪ specify](#).
- Use the ``-customColTypeMapping`` option to change the data type of selected columns.
- Use the ``-dropSchema`` option to drop the existing schema and create a new schema, [↪ prior to migration](#).

- On EDB Postgres Advanced Server, use the ``-allDBLinks`` option to migrate all Oracle database links.
- On EDB Postgres Advanced Server, use the ``-copyViaDBLinkOra`` option to enable the ``dblink_ora`` module.

3.1 Object migration support

Migration Toolkit migrates object definitions (DDL), table data, or both. The following table contains a platform-specific list of the types of database objects that Migration Toolkit can migrate:

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			

continues on next page

Table 1 – continued from previous page

Object	Oracle	Sybase	SQL Server	MySQL

For detailed information about the commands that offer granular control of the objects imported, see [Schema object selection options](#).

3.2 Online migration versus offline migration

Migration Toolkit can migrate immediately and directly into a Postgres database (*online migration*). You can also choose to generate scripts to use later to re-create object definitions in a Postgres database (*offline migration*).

By default, Migration Toolkit creates objects directly into a Postgres database. Alternatively, include the `-offlineMigration` option to generate SQL scripts you can use later to reproduce the migrated objects or data in a new database. You can alter migrated objects by customizing the migration scripts generated by Migration Toolkit before you execute them. With the `-offlineMigration` option, you can schedule the actual migration at a time that best suits your system load.

For more information about the `-offlineMigration` option, see [Offline migration options](#).

INSTALLING MIGRATION TOOLKIT

Before installing Migration Toolkit, you must install Java (version 1.8.0 or later).

To install Migration Toolkit, you must have credentials that allow access to the EnterpriseDB repository. To request credentials for the repository, visit the EnterpriseDB website at:

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

The following are the various installation methods:

- [Installing Migration Toolkit on a CentOS **or** RHEL Host](#rpm_installation)
- [Installing Migration Toolkit on a CentOS, **or** RHEL 7 ppc64le Host](#rhel_centos_ppc64le_installation)
- [Installing Migration Toolkit on a Debian **or** Ubuntu Host](#debian_or_ubuntu_installation)
- [Installing Migration Toolkit on a SLES Host](#sles_installation)
- [Installing Migration Toolkit on Windows](#windows_installation)
- [Installing Migration Toolkit on Mac OS X](#mac_installation)

4.1 Installing Migration Toolkit on a CentOS or RHEL 7 and 8 Host

You can use an RPM package to install Migration Toolkit on a CentOS or RHEL host. The following steps provide detailed information about accessing the EnterpriseDB repository and installing Migration Toolkit.

Creating a Repository Configuration File

1. To create the repository configuration file, assume superuser privileges and invoke the following command:

- On RHEL **or** CentOS 7:

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

- On RHEL **or** CentOS 8:

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

- The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.
2. After creating the `edb.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is 1, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EnterpriseDB user.

```
* [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. After saving your changes to the configuration file, install the EPEL repository:

- On RHEL **or** CentOS **7**:

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

- On RHEL **or** Centos 8:

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

4. If you are using RHEL or CentOS 8.x, you must enable an additional repository to resolve dependencies:

- On CentOS 8 (Powertools repo):

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

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

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

Installing Migration Toolkit

- On RHEL **or** CentOS **7**:

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

- On RHEL **or** CentOs **8**:

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

When you install an RPM package that is signed by a source that is not recognized by your system, yum may ask for your permission to import the key to your local server. If prompted, and you are satisfied that the packages come from a trustworthy source, enter a y, and press Return to continue.

During the installation, yum may encounter a dependency that it cannot resolve. If it does, it will provide a list of the required dependencies that you must manually resolve.

After installing Migration Toolkit, you must configure the installation. Perform the following steps before invoking Migration Toolkit.

Using Migration Toolkit with IDENT Authentication

By default, the `pg_hba.conf` file for the RPM installer enforces IDENT authentication for remote clients. Before invoking Migration Toolkit, you must either modify the `pg_hba.conf` file, changing the authentication method to a form other than IDENT (and restarting the server), or perform the following steps to ensure that an IDENT server is accessible:

1. Confirm that an `identd` server is installed and running. You can use the yum package manager to install an `identd` server by invoking the command:

- ``yum install xinetd authd``

- The command should create a file named ``/etc/xinetd.d/authd`` that contains:

```
* 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

- If the file includes a ``-E`` argument at the end of ``server_args``, please erase the `↵`-E``.

- Then, to start the `identd` server, invoke the commands:

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

2. Open the `pg_ident.conf` file and create a user mapping:

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

- Where:

- – The name specified in the `map_name` column is a user-defined name that will identify the mapping in the `pg_hba.conf` file. – The name specified in the `system_username` column is `enterprisedb`. – The name specified in the `postgres_username` column is `enterprisedb`.

3. Open the `pg_hba.conf` file and modify the IDENT entries.

- On an IPv4 host:
 - `host all all 127.0.0.0/0 ident map=edbas`
 - On an IPv6 host:
 - `host all all ::1/128 ident map=edbas`
4. Restart the EDB Postgres Advanced Server service before invoking Migration Toolkit.

4.1.1 Upgrading an RPM Installation

If you have an existing RPM installation, you can use `yum` to upgrade your repository configuration file and update to a more recent product version. To update the `edb.repo` file, assume superuser privileges and enter:

```
yum upgrade edb-repo
```

`yum` will update the `edb.repo` file to enable access to the current EnterpriseDB repository, configured to connect with the credentials specified in your `edb.repo` file. Then, you can use `yum` to upgrade Migration toolkit:

```
yum upgrade edb-migrationtoolkit
```

4.2 Installing Migration Toolkit on a CentOS, or RHEL 7 ppc64le Host

You can use an RPM package to install Migration Toolkit on a CentOS or RHEL ppc64le host.

Before installing Migration toolkit , you must:

1. Create the Advance Toolchain repository configuration file; assume superuser privileges and invoke the following command:

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

2. After creating the `advance-toolchain.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is 1, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EnterpriseDB user.

```
* [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
```

The following steps provide detailed information about accessing the EnterpriseDB repository and installing Migration Toolkit.

Creating a Repository Configuration File

1. To create the EDB repository configuration file, assume superuser privileges and invoke the following command:

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

- The repository configuration file is named `edb.repo`. The file resides in `/etc/yum.repos.d`.
2. After creating the `edb.repo` file, use your choice of editor to ensure that the value of the `enabled` parameter is 1, and replace the `username` and `password` placeholders in the `baseurl` specification with the name and password of a registered EnterpriseDB user.

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

3. After saving your changes to the configuration file, install the EPEL package:

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

4. If you are using RHEL 7 ppc64le, you must enable an additional repository to resolve dependencies:

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

5. To install Migration Toolkit, run the following command.

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

4.3 Installing Migration Toolkit on a Debian or Ubuntu Host

To install a package with `apt`, complete the following steps. When invoking the commands, replace the `username` and `password` with the user credentials provided by EnterpriseDB.

1. Assume superuser privileges:

```
* sudo su -
```

2. Configure the EnterpriseDB repository:

- – On Debian 9, perform the following step to configure the EnterpriseDB repository:

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

- – On Debian 10, perform the following steps to configure the EnterpriseDB repository:

- a. Set up the EnterpriseDB repository:

```
* 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. Substitute your EnterpriseDB credentials for the ``username`` and ``password`` placeholders in the following command:

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

- c. Add support to your system **for** secure APT repositories:

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

- d. Add the EBD signing key:

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

- e. Update the repository metadata:

```
* apt-get update
```

- f. Install the ``edb-migrationtoolkit`` package:

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

4.4 Installing Migration Toolkit on a SLES Host

You can use the zypper package manager to install a Migration toolkit on an SLES 12 host. zypper will attempt to satisfy package dependencies as it installs a package, but requires access to specific repositories that are not hosted at EnterpriseDB.

1. You must assume superuser privileges and stop any firewalls before installing Migration Toolkit. Then, use the following commands to add EnterpriseDB repositories to your system:

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

2. The command creates the repository configuration files in the `/etc/zypp/repos.d` directory. Use the following command to refresh the metadata on your SLES host to include the EnterpriseDB repository:

```
* * zypper refresh
```

- When prompted, provide credentials for the repository; specify **a** to always trust the provided key and update the metadata to include the EnterpriseDB repository.

3. You must also add SUSEConnect and the SUSE Package Hub extension to the SLES host, and register the host with SUSE, allowing access to SUSE repositories. Use the commands:

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

4. Install SUSEConnect to register the host with SUSE, allowing access to SUSE repositories:

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

5. Use the following command to refresh the metadata:

```
* * zypper refresh
```

6. Install OpenJDK (version 1.8) for Java based components:

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

7. Now you can use the zypper utility to install a Migration Toolkit:

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

- For detailed information about registering a SUSE host, visit:
- <https://www.suse.com/support/kb/doc/?id=7016626>

4.5 Installing Migration Toolkit on Windows

You can use the graphical installer to install the Migration Toolkit on Windows. You can download the installer from www.enterprisedb.com.

Once you download the installer, right-click the installer icon, and select **Run As Administrator** from the context menu. When the **Language Selection** popup opens, select an installation language and click **OK** to continue to the **Setup** window.

!!! Note * This guide uses the term **Stack Builder** to refer to either **StackBuilder Plus** (distributed with EDB Postgres Advanced Server) or **Stack Builder** (distributed with the PostgreSQL one-click installer from EnterpriseDB).

You must have a Java JVM (version 1.8.0 or later) in place before Stack Builder can perform a Migration Toolkit installation.

The Java executable must be in your search path (%PATH% on Windows) Use the following commands to set the search path (substituting the name of the directory that holds the Java executable for `javadir`):

On Windows, use the command:

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

On Linux, use the command:

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

After setting the search path, you can use the Stack Builder installation wizard to install Migration Toolkit into either EDB Postgres Advanced Server or PostgreSQL.

Launching StackBuilder Plus from EDB Postgres Advanced Server

To launch StackBuilder Plus from an existing EDB Postgres Advanced Server installation, navigate through the Start (or Applications) menu to the EDB Postgres menu; open the EDB Add-ons menu, and select the StackBuilder Plus menu option.

Launching Stack Builder from PostgreSQL

To launch Stack Builder from a PostgreSQL installation, navigate through the Start (or Applications) menu to the PostgreSQL menu, and select the Application StackBuilder Plus menu option.

Stack Builder opens as shown below.

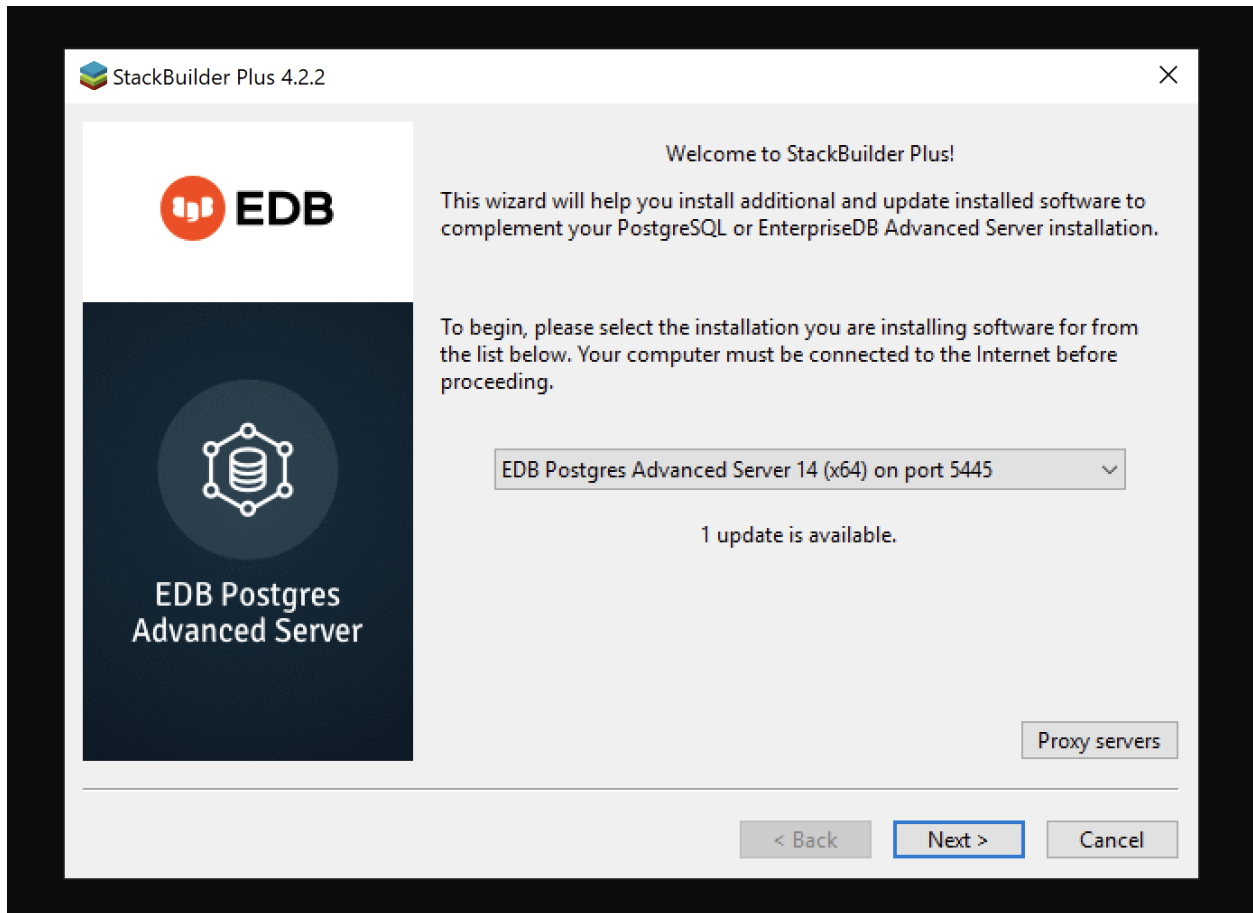


Fig. 1: The Stack Builder welcome window.

- The Stack Builder welcome window*

Use the drop-down listbox to select the target server installation from the list of available servers. If your network requires you to use a proxy server to access the Internet, use the Proxy servers button to open the Proxy servers dialog and specify a server; if you do not need to use a proxy server, click Next to open the application selection window.

If you are using StackBuilder Plus to add Migration Toolkit to your EDB Postgres Advanced Server installation, expand the Add-ons, tools and utilities node of the tree control, and check the box next to EnterpriseDB Migration Toolkit. Click Next to continue.

!!! Note * If you are using Stack Builder to add Migration Toolkit to your PostgreSQL installation, expand the EnterpriseDB Tools node of the tree control (located under the Registration-required and trial productsnode), and select Migration Toolkit checkbox. Click Next to continue.

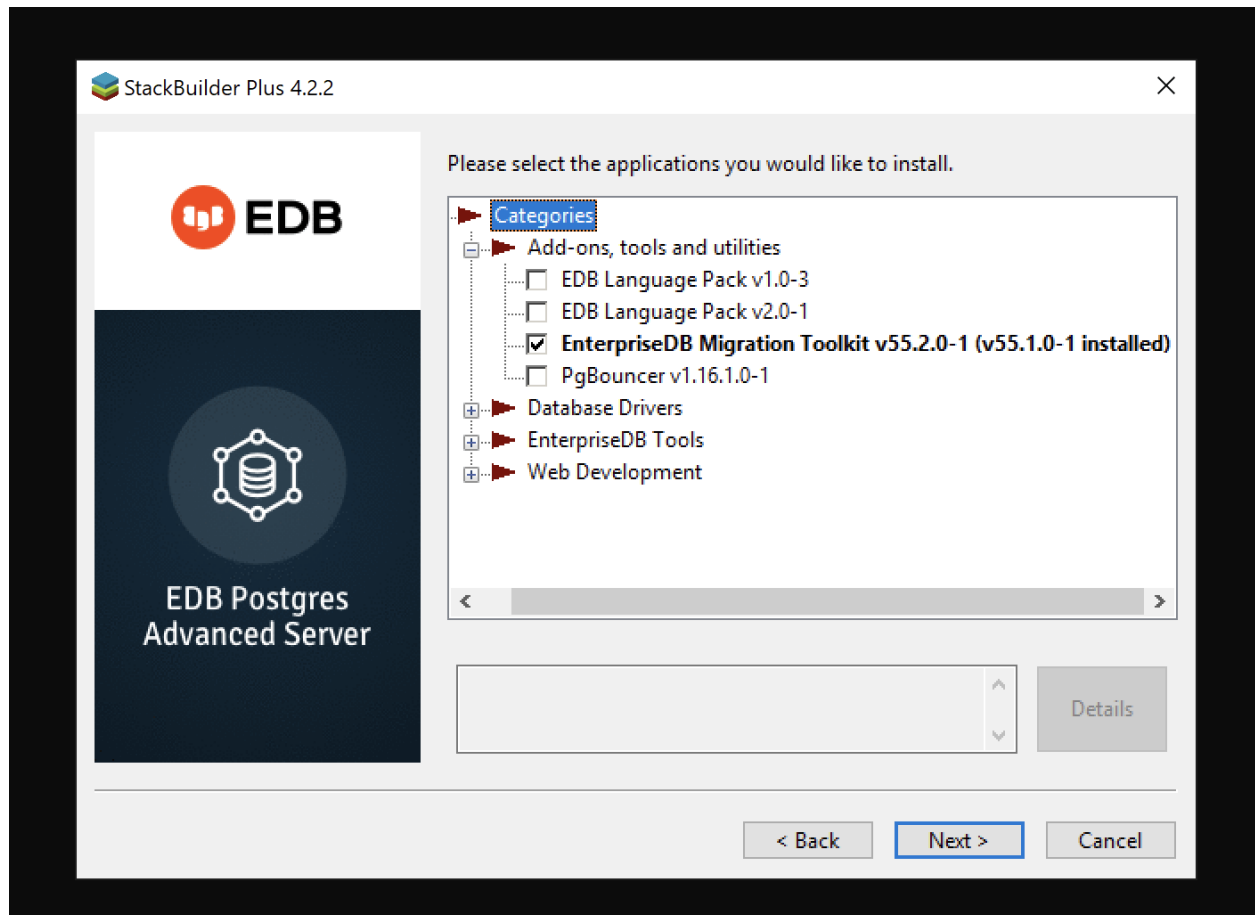


Fig. 2: The Stack Builder application selection window.

- The Stack Builder application selection window*

Confirm that Migration Toolkit is included in the **Selected Packages** list and that the **Download** directory field contains an acceptable download location.

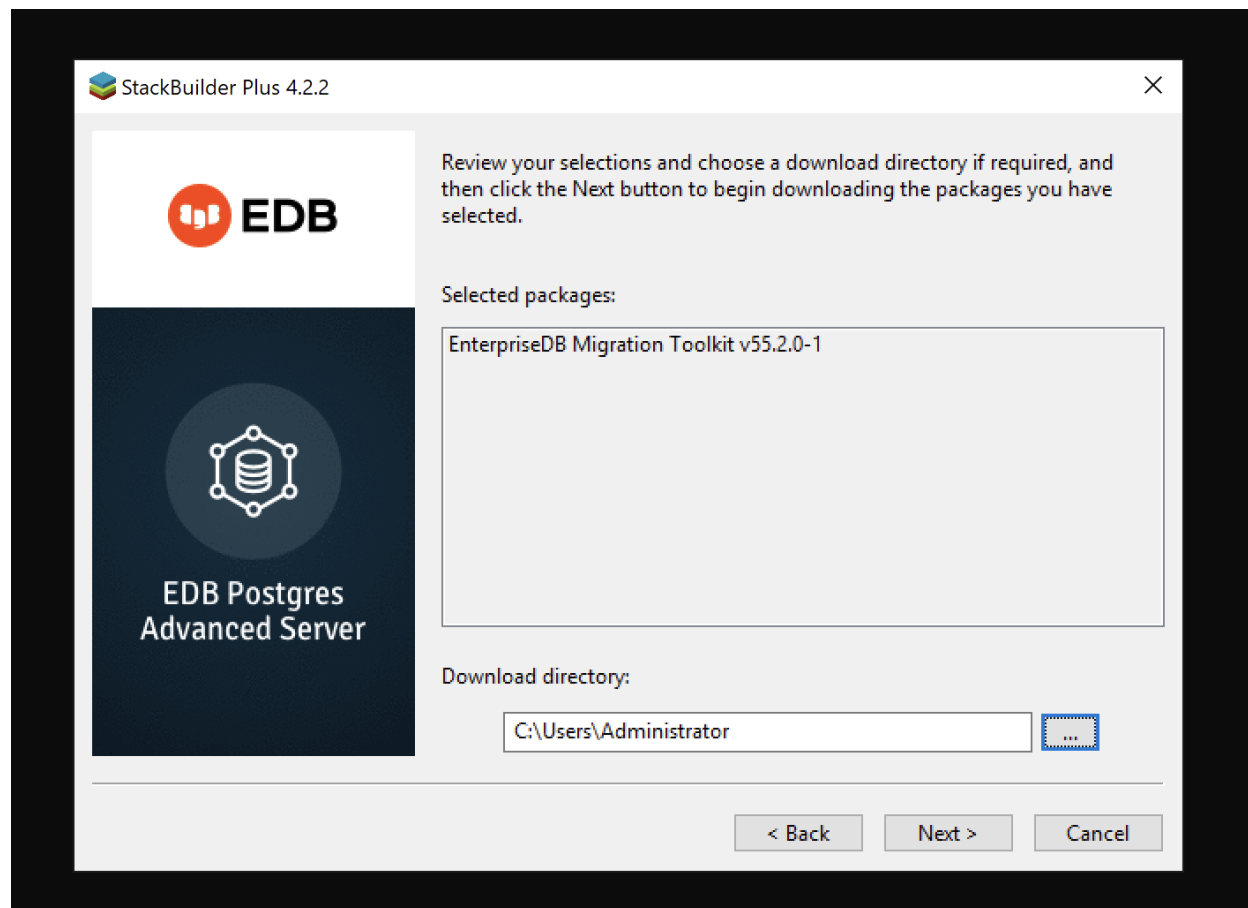


Fig. 3: The Stack Builder selection confirmation window.

- The Stack Builder selection confirmation window*

Click **Next** to start the Migration Toolkit download.

- Stack Builder confirms the file download*

When the download completes, Stack Builder confirms that the installation files have been successfully downloaded. Choose **Next** to open the Migration Toolkit installation wizard.

When prompted by the Migration Toolkit installation wizard, specify a language for the installation and click **OK** to continue.

The Migration Toolkit installation wizard opens. Click **Next** to continue.

- The Migration Toolkit installation wizard*

Carefully review the license agreement before highlighting the appropriate radio button; click **Next** to continue.

- The EnterpriseDB license agreement*

By default, Migration Toolkit will be installed in the `mtk` directory; accept the default installation directory as displayed, or modify the directory, and click **Next** to continue.

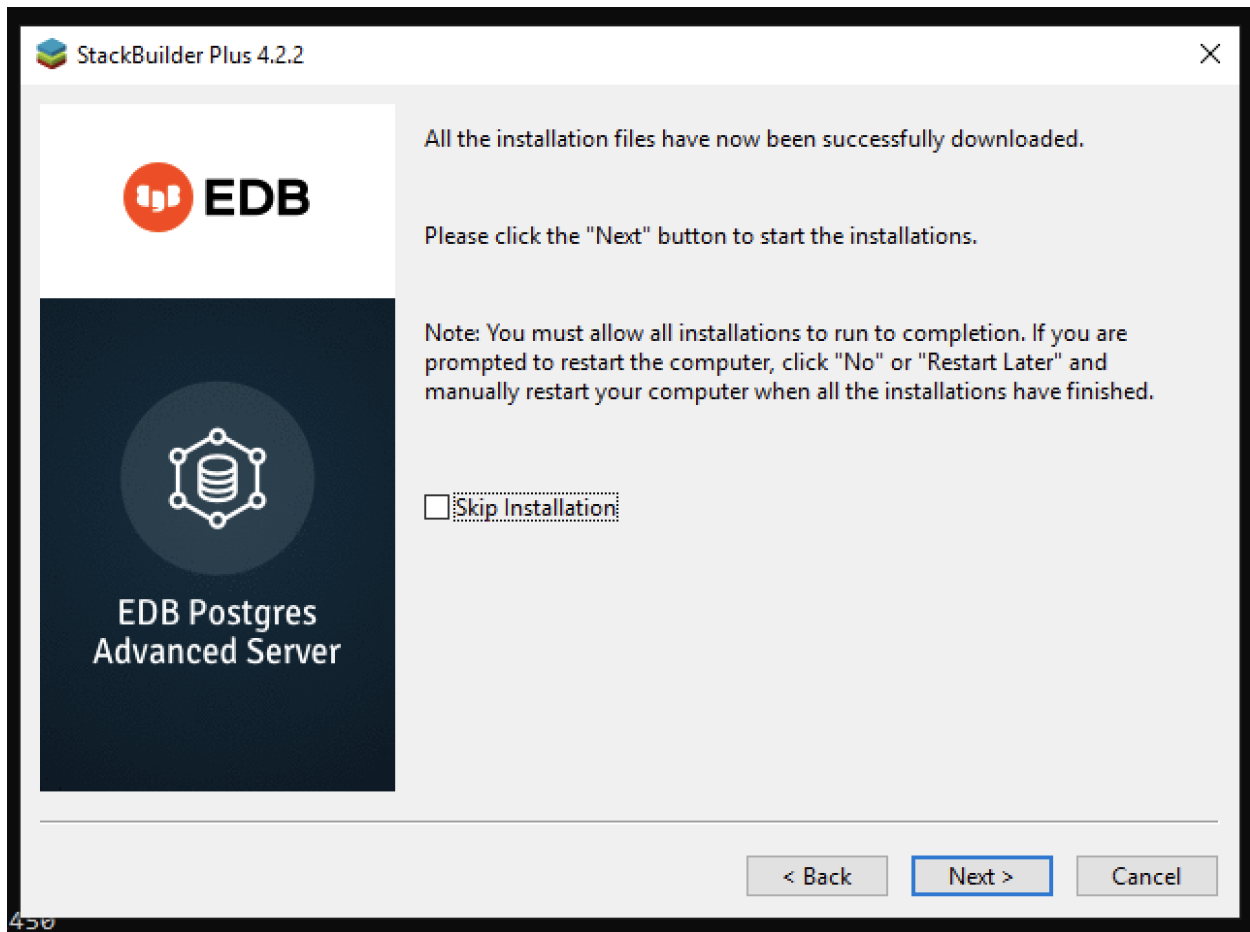


Fig. 4: Stack Builder confirms the file download.

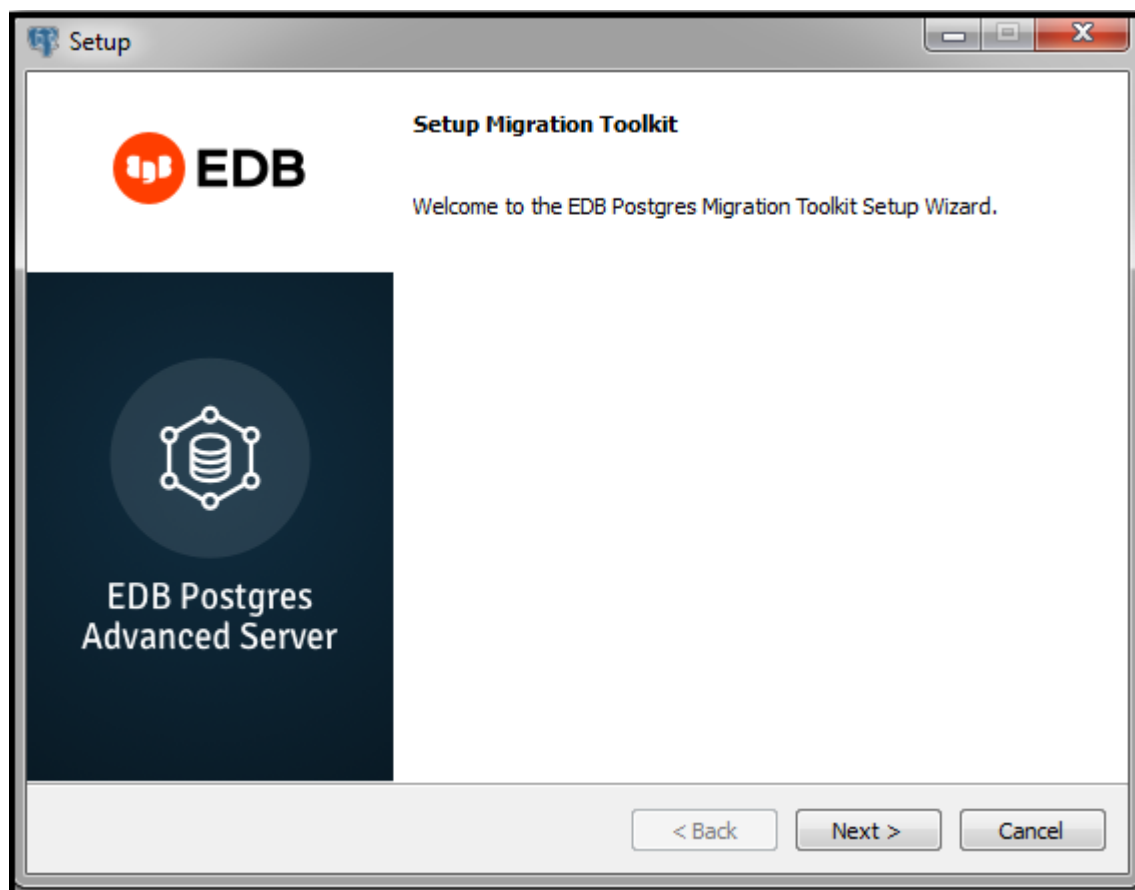


Fig. 5: The Migration Toolkit installation wizard.

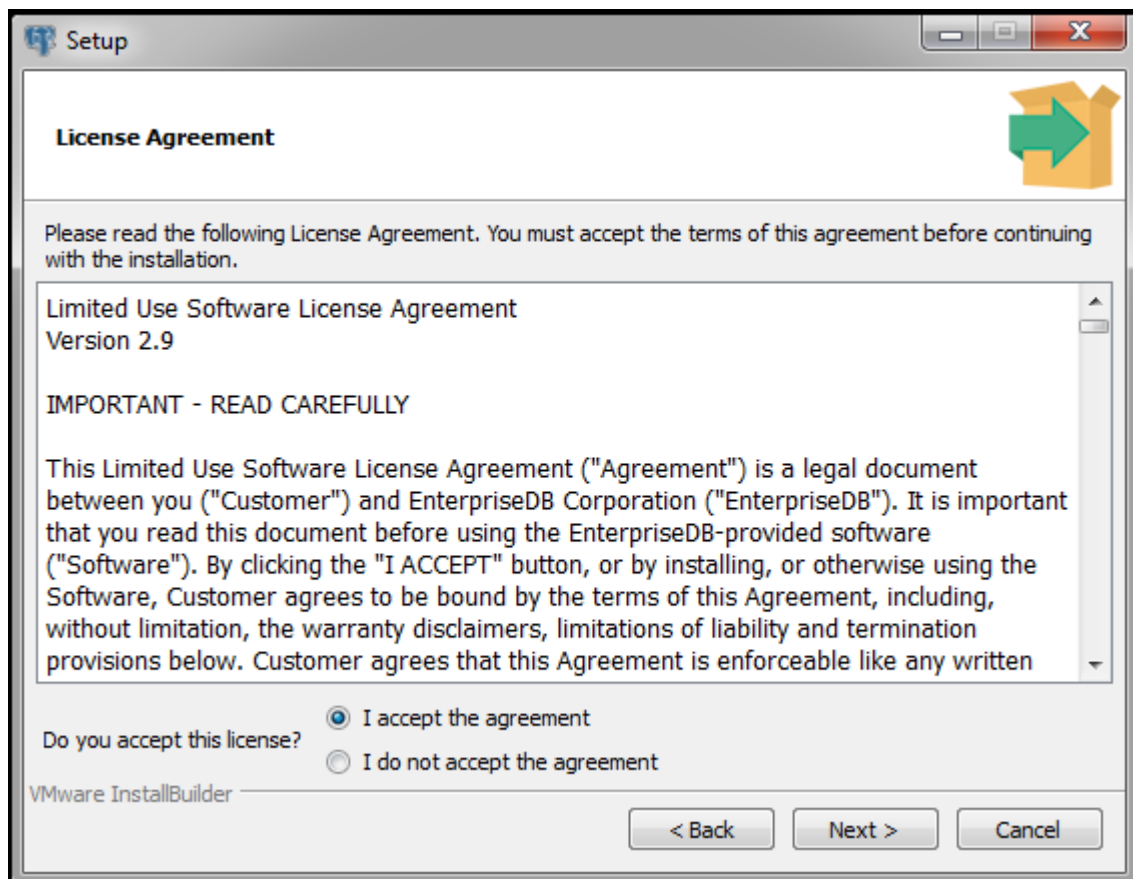


Fig. 6: The EnterpriseDB license agreement.

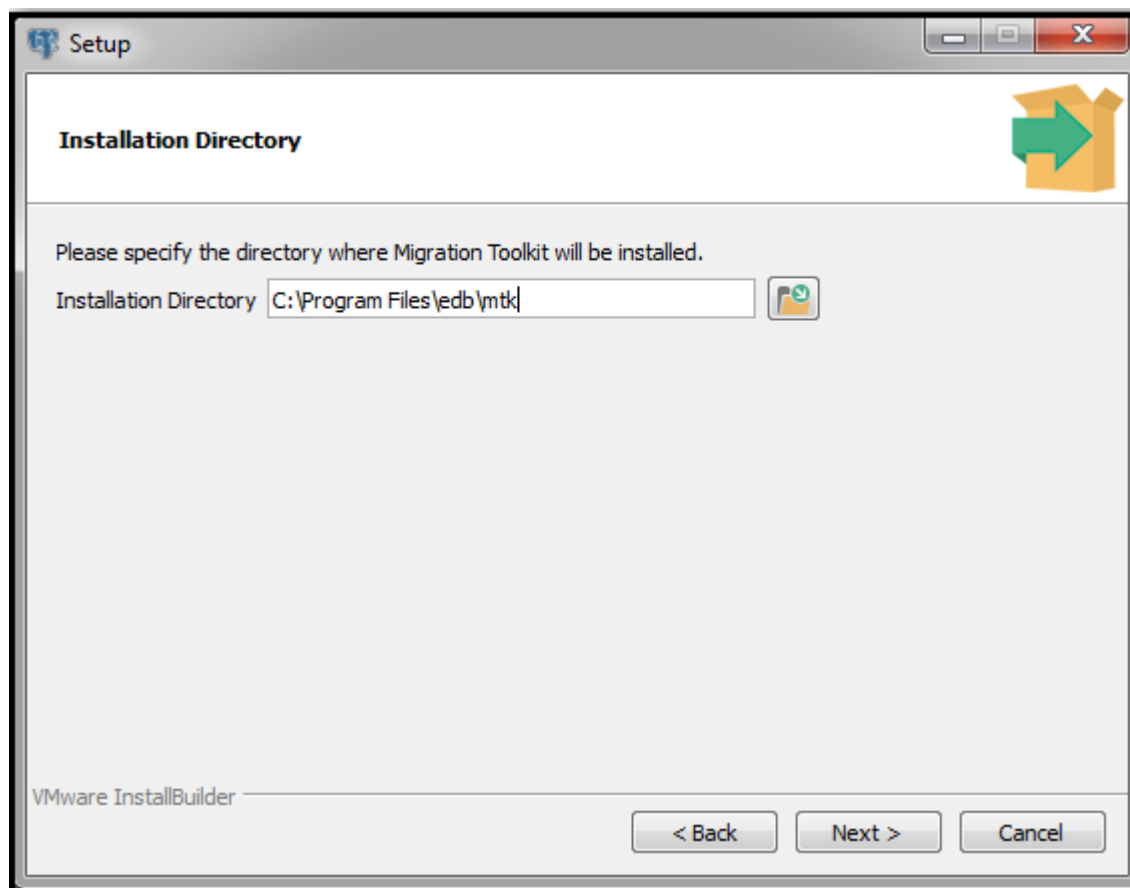


Fig. 7: Specify a Migration Toolkit installation directory.

- Specify a Migration Toolkit installation directory*

The installation wizard confirms that the Setup program is ready to install Migration Toolkit; click **Next** to start the installation.

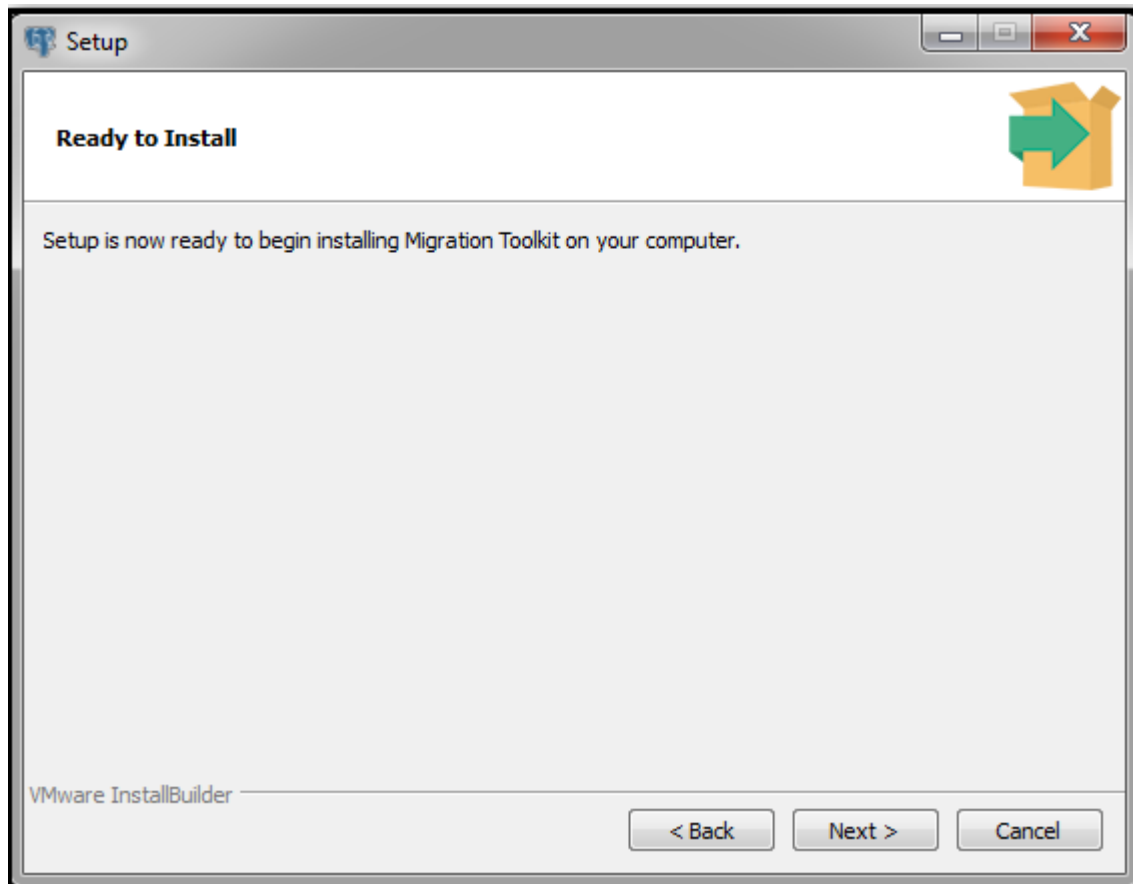


Fig. 8: Setup is ready to start the Migration Toolkit installation.

- Setup is ready to start the Migration Toolkit installation*

A dialog confirms that the Migration Toolkit installation is complete; click **Finish** to exit the Migration Toolkit installer.

- The Migration Toolkit wizard has finished*

When Stack Builder finalizes installation of the last selected component, it displays the **Installation Completed** window. Click **Finish** to close Stack Builder.

- Stack Builder confirms the installation is complete*

After installing Migration Toolkit with Stack Builder, you must install the appropriate source-specific drivers before performing a migration; please see *Installing Source-Specific Drivers* for more information.

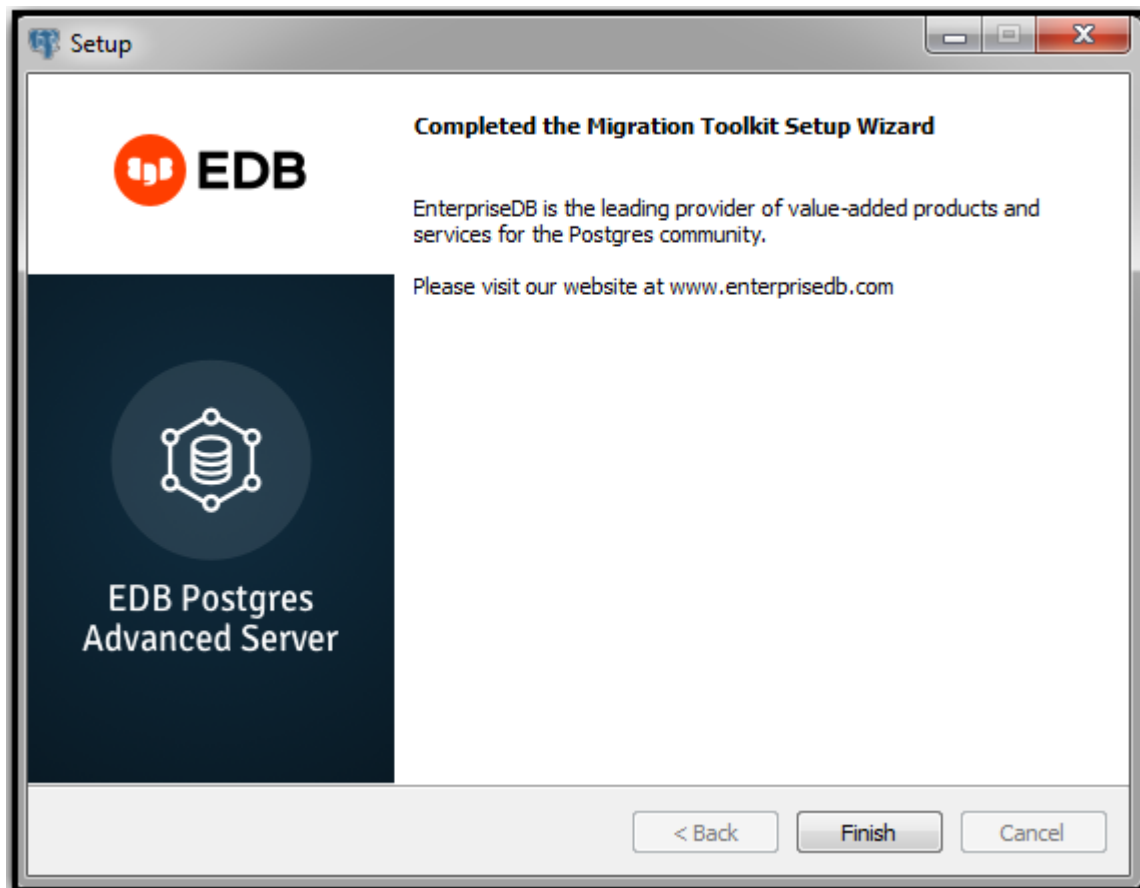


Fig. 9: The Migration Toolkit wizard has finished.

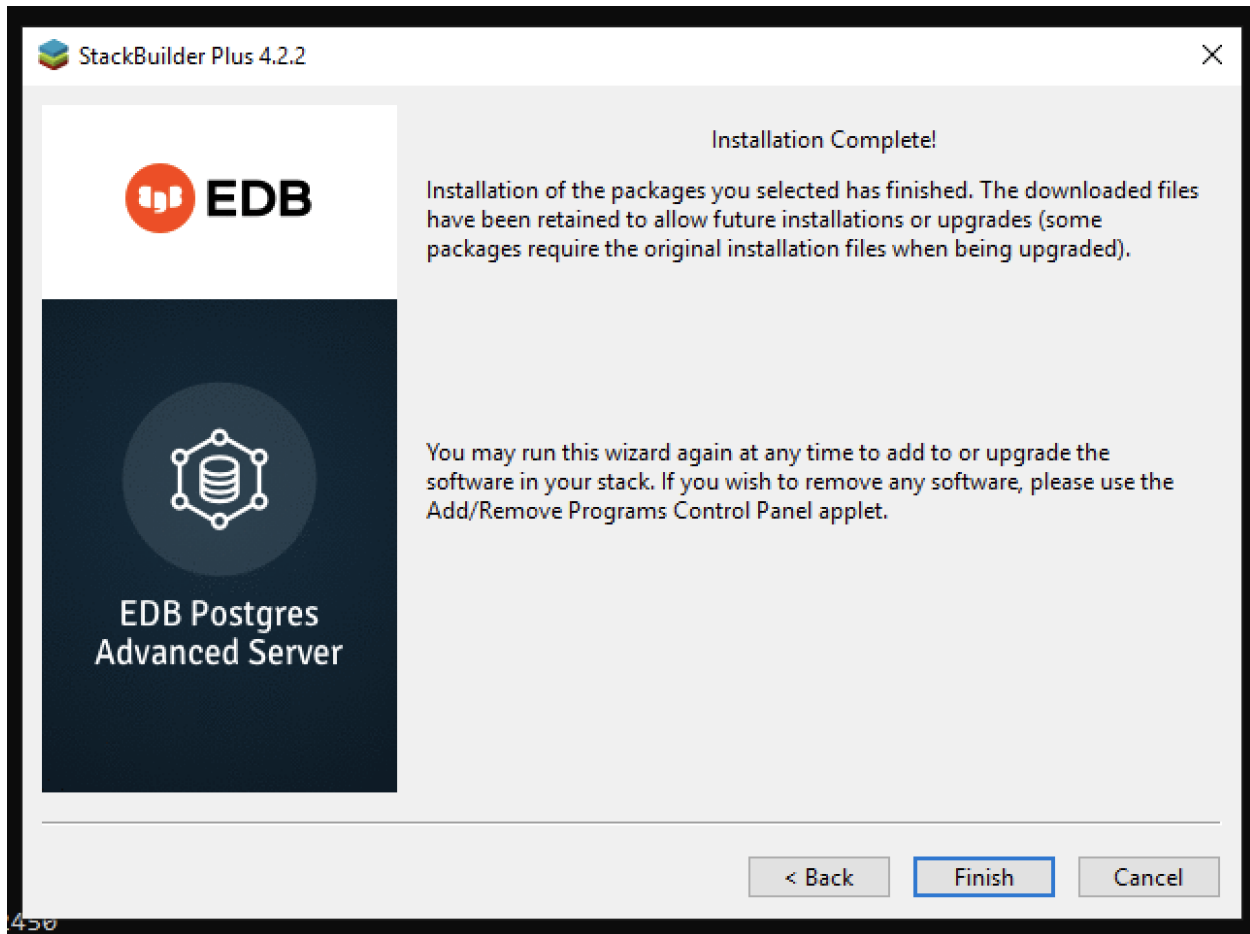


Fig. 10: Stack Builder confirms the installation is complete.

4.6 Installing Migration Toolkit on Mac OS X

You can use the graphical installer to install the Migration Toolkit on Mac OS X. To download the installer, visit www.enterprisedb.com.

To launch the installer:

- Navigate to the file path **in** the terminal:

```
- /User/Directory>/Downloads/edb-migrationtoolkit-55.2.0-1-osx.app/Contents/MacOS
```

- Assume superuser privileges **and** invoke the following command:

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

1. On the Language Selection popup opens, select an installation language and click OK to continue to the Setup window.
2. The Migration Toolkit installation wizard opens. Click Next to continue.

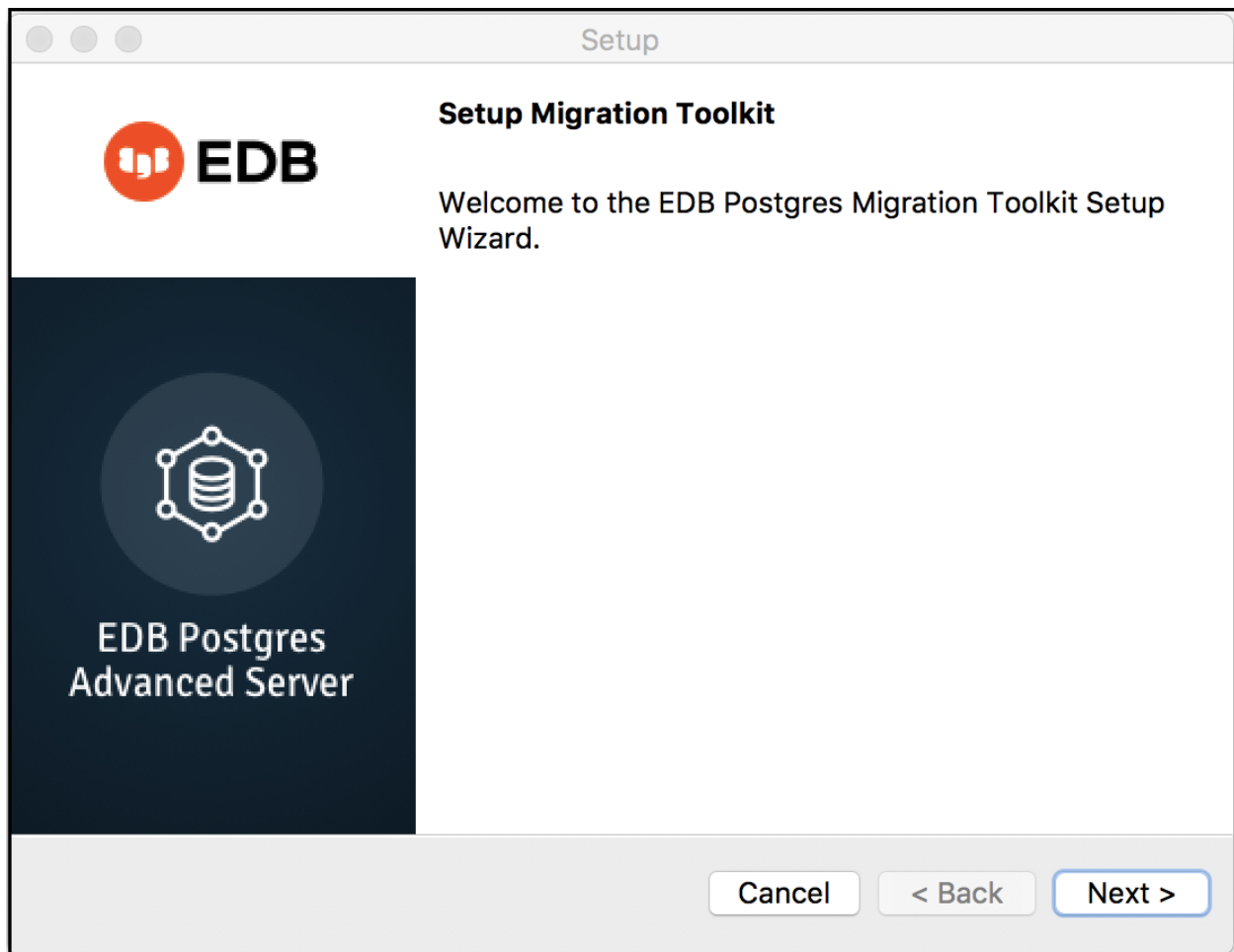


Fig. 11: The Migration Toolkit installation wizard.

- The Migration Toolkit installation wizard*

3. Carefully review the license agreement before highlighting the appropriate radio button; click **Next** to continue.

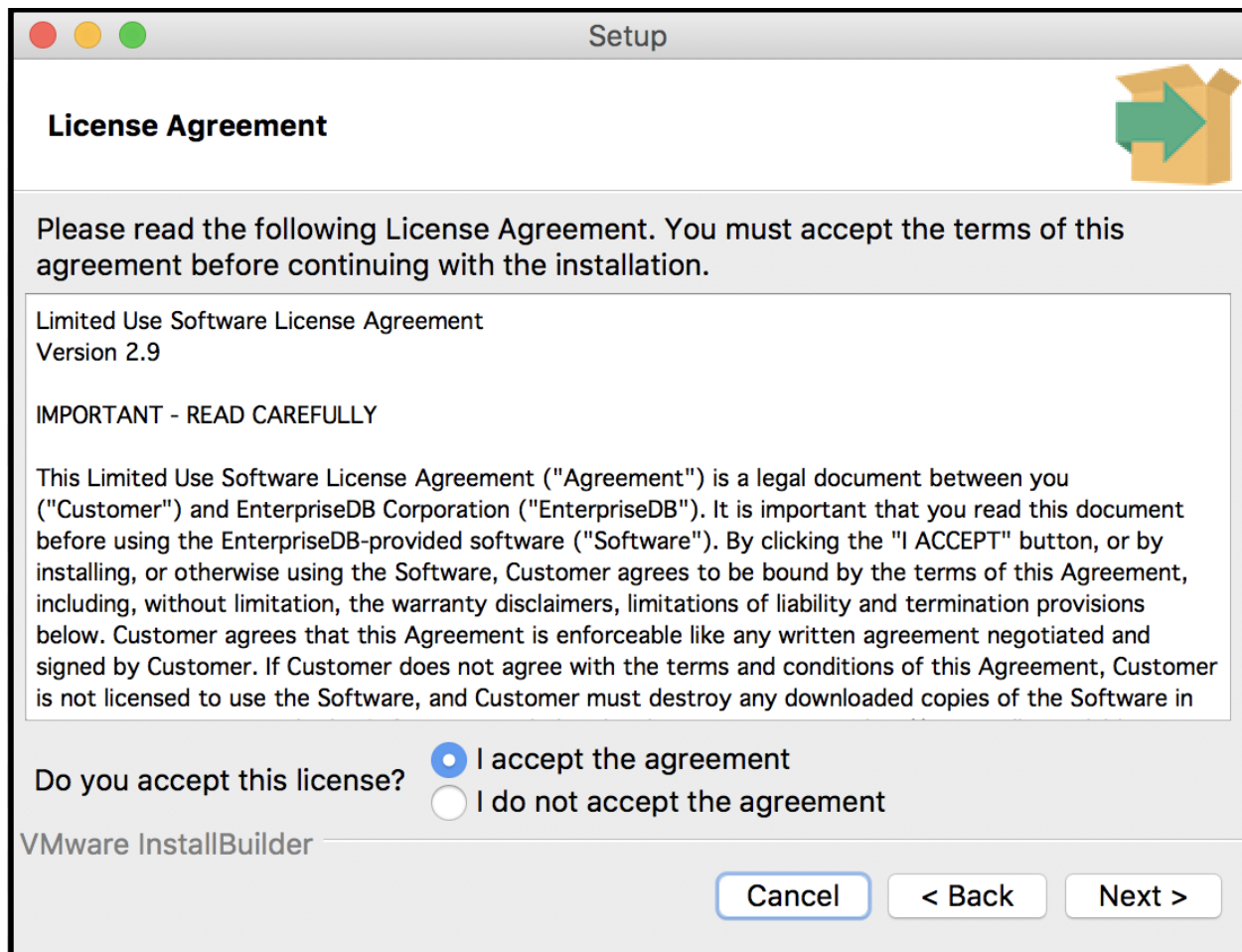


Fig. 12: The EnterpriseDB license agreement.

- The EnterpriseDB license agreement*

By default, Migration Toolkit will be installed in the `mtk` directory; accept the default installation directory as displayed, or modify the directory, and click **Next** to continue.

- Specify a Migration Toolkit installation directory*

The installation wizard confirms that the **Setup** program is ready to install Migration Toolkit; click **Next** to start the installation.

- Setup is ready to start the Migration Toolkit installation*

A dialog confirms that the Migration Toolkit installation is complete; click **Finish** to exit the Migration Toolkit installer.

- The Migration Toolkit wizard has finished.*

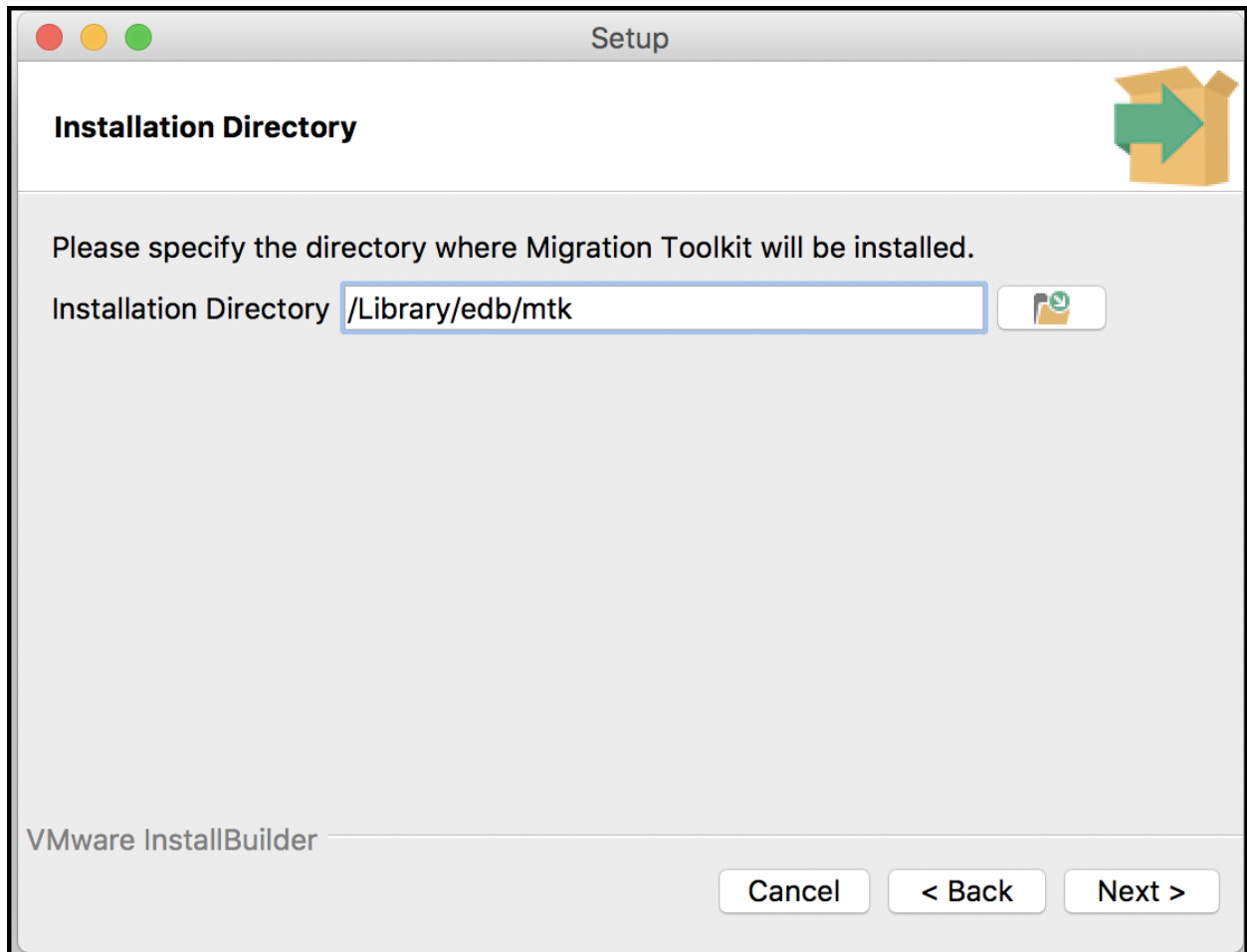


Fig. 13: Specify a Migration Toolkit installation directory.

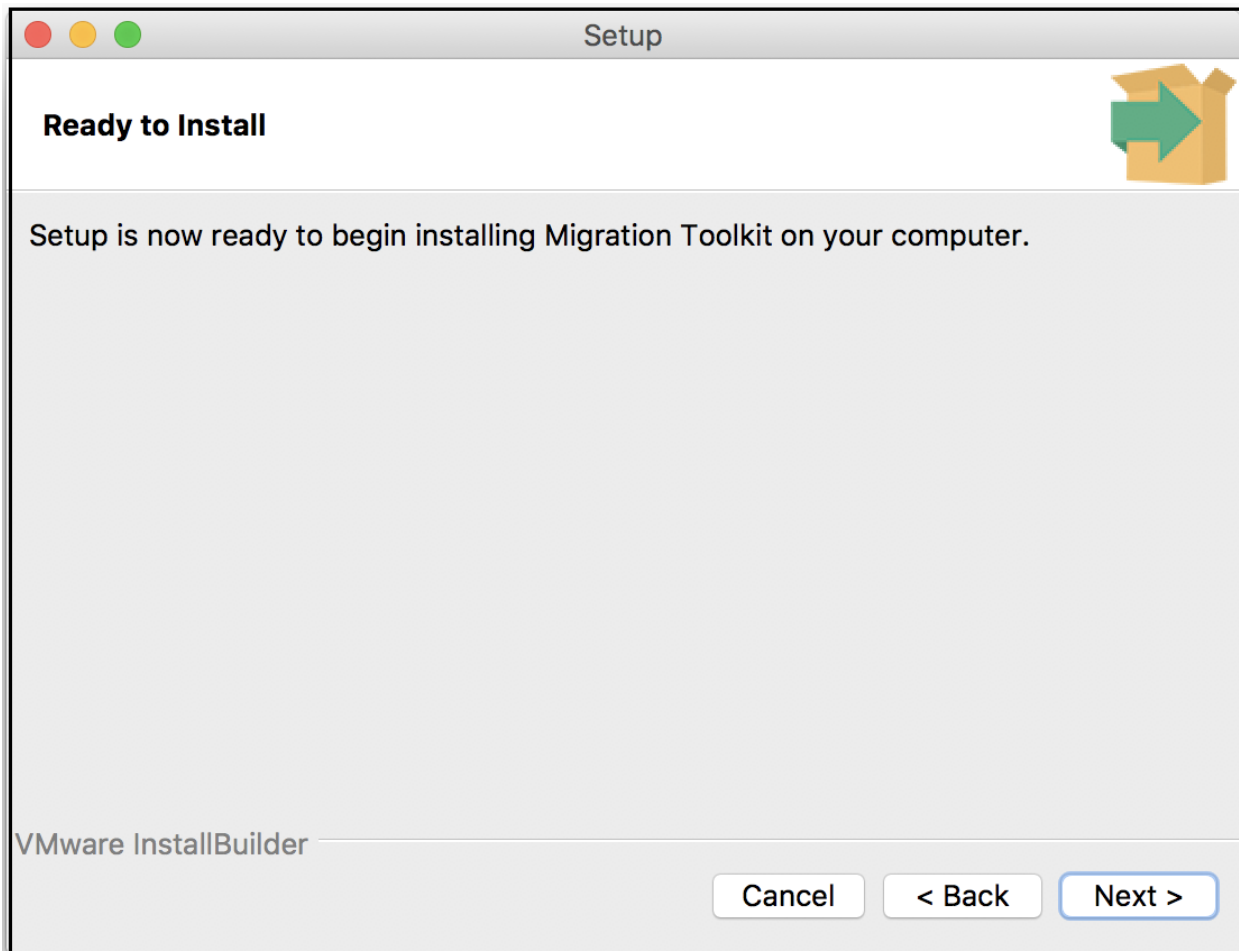


Fig. 14: Setup is ready to start the Migration Toolkit installation.

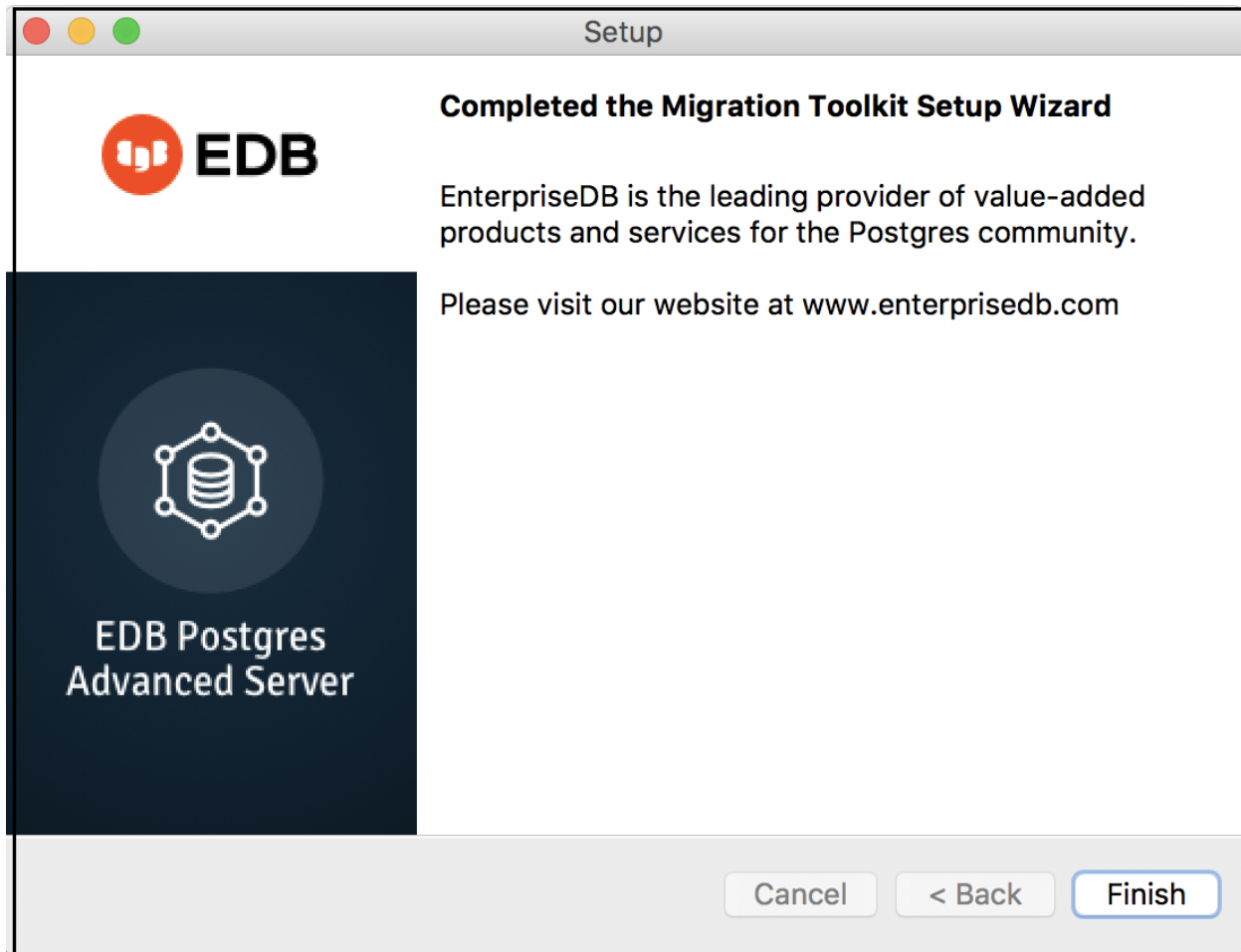


Fig. 15: The Migration Toolkit wizard has finished.

4.7 Installing Source-Specific Drivers

Before invoking Migration Toolkit, you must download and install a freely available source-specific driver. To download a driver, or for a link to a vendor download site, visit the **Third Party JDBC Drivers** section of the **Advanced Downloads** page at the EnterpriseDB website:

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

After downloading the source-specific driver, move the driver file into the `<mtk_install_dir>/lib` directory.

BUILDING THE TOOLKIT.PROPERTIES FILE

Migration Toolkit uses the configuration and connection information stored in the `toolkit.properties` file during the migration process to identify and connect to the source and target databases.

On Linux, the `toolkit.properties` file is located in `/usr/edb/migrationtoolkit/etc`.

On Windows, the file is located in `C:\Program Files\edb\mtk\etc`.

The figure shows a sample `toolkit.properties` file:

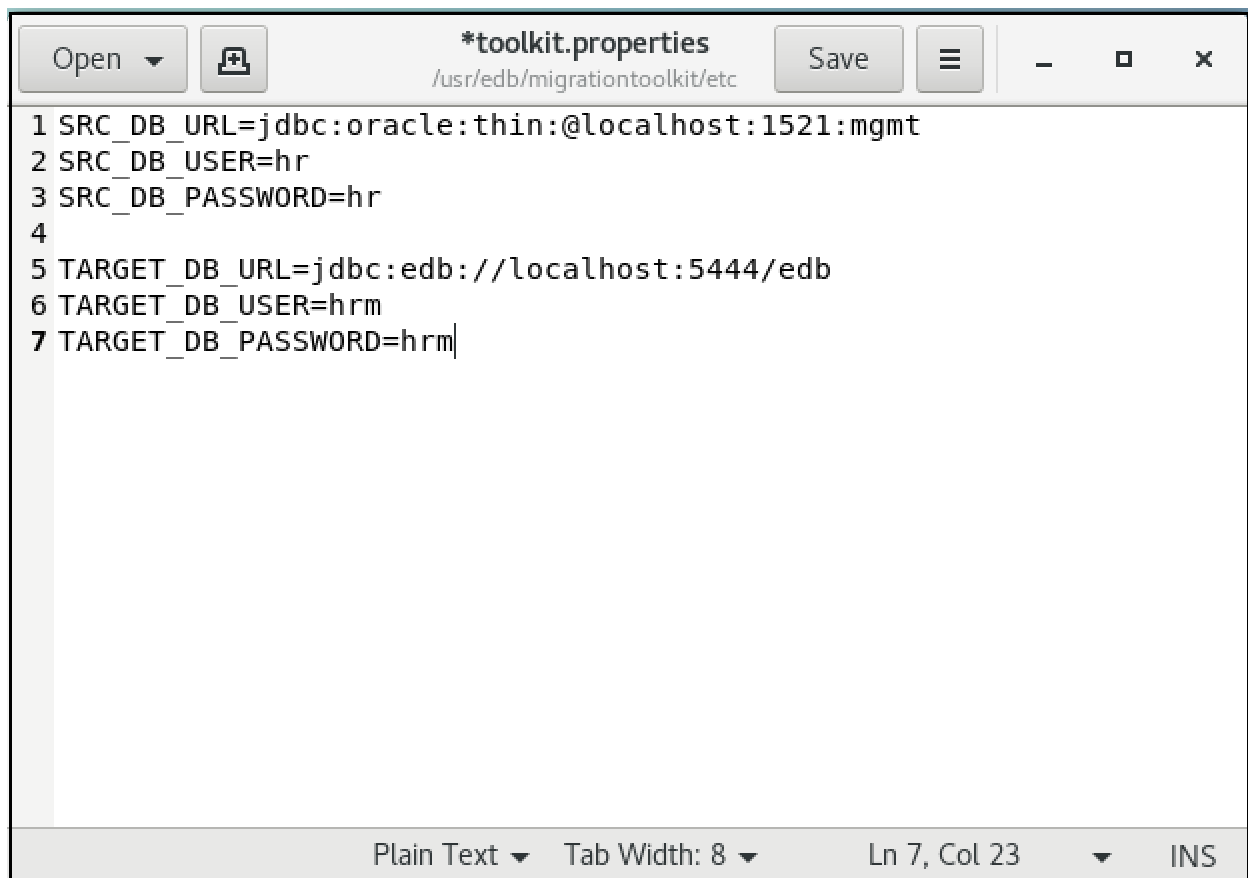


Fig. 1: A typical `toolkit.properties` file.

Before executing Migration Toolkit commands, modify the `toolkit.properties` file with the editor of your choice. Update the file to include the following information:

- ``SRC_DB_URL`` specifies how Migration Toolkit connects to the source database.↵
↪Details on forming the URL for each source database follows.
- ``SRC_DB_USER`` specifies a user name (with sufficient privileges) in the source↵
↪database.
- ``SRC_DB_PASSWORD`` specifies the password of the source database user.
- ``TARGET_DB_URL`` specifies the JDBC URL of the target database.
- ``TARGET_DB_USER`` specifies the name of a privileged target database user.
- ``TARGET_DB_PASSWORD`` specifies the password of the target database user.

For specifying a target database on BigAnimal, see *Defining target information for BigAnimal*.

5.1 Defining an Advanced Server URL

Migration Toolkit helps with migration from the following platforms to Advanced Server:

- Oracle
- PostgreSQL
- MySQL
- Sybase
- SQL Server

For a definitive list of the objects migrated from each database type, refer to [Functionality overview](#).

Migration Toolkit reads connection specifications for the source and the target database from the `toolkit.properties` file. Connection information for each must include:

- The URL of the database
- The name of a privileged user
- The password associated **with** the specified user

The URL conforms to JDBC standards and takes the form:

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

An Advanced Server URL contains the following information:

- ``jdbc`` — The protocol is always ``jdbc``.

- ``edb`` — If you're using Advanced Server, specify ``edb`` for the subprotocol value.
- ``<host>`` — The name or IP address of the host where the Postgres instance is running.
- ``<port>`` — The port number that the Advanced Server database listener is monitoring. The default port number is 5444.
- ``<database_id>`` — The name of the source or target database.
- ``{TARGET_DB_USER|SRC_DB_USER}`` — Specifies a user with privileges to create each type of object migrated. If migrating data into a table, the specified user might also require insert, truncate, and references privileges for each target table.
- ``{TARGET_DB_PASSWORD|SRC_DB_PASSWORD}`` — Set to the password of the privileged Advanced Server user.

5.2 Defining a PostgreSQL URL

Migration Toolkit helps with migration from the following platforms to PostgreSQL:

- Oracle
- Advanced Server
- MySQL
- SQL Server

For a definitive list of the objects migrated from each database type, refer to [Functionality overview](#).

Migration Toolkit reads connection specifications for the source and the target database from the `toolkit.properties` file. Connection information for each must include:

- The URL of the database
- The name of a privileged user
- The password associated with the specified user

A PostgreSQL URL conforms to JDBC standards and takes the form:

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

The URL contains the following information:

- ``jdbc`` — The protocol is always ``jdbc``.
- ``postgresql`` — If you're using PostgreSQL, specify ``postgresql`` for the `subprotocol` value.
- ``<host>`` — The name or IP address of the host where the Postgres instance is `running`.
- ``<port>`` — The port number that the Postgres database listener is monitoring. `The default port number is 5432.`
- ``<<database_id>`` — The name of the source or target database.
- ``{SRC_DB_USER|TARGET_DB_USER}`` — Specify a user with privileges to create `each type of object migrated. If migrating data into a table, the specified user might also need insert, truncate, and references privileges for each target table.`
- ``{SRC_DB_PASSWORD|TARGET_DB_PASSWORD}`` — Set to the password of the `privileged PostgreSQL user.`

5.3 Defining a BigAnimal URL

Migration Toolkit helps with migration from the following platforms to BigAnimal:

- Oracle
- Advanced Server
- PostgreSQL
- MySQL
- SQL Server

Migration Toolkit reads connection specifications for the source and the target database from the `toolkit.properties` file. Connection information for each must include:

- The URL of the database
- The name of a privileged user
- The password associated **with** the specified user

When migrating to BigAnimal, `TARGET_DB_URL` takes the form of a JDBC URL. For example:

The URL contains the following information:

- ``jdbc`` — The protocol is always ``jdbc``.
- ``postgres_type`` — The subprotocol is the Postgres type. Specify ``edb`` if you're using Advanced Server or ``postgresql`` if you're using PostgreSQL.
- ``<host_name>`` — The host name of your cluster. You can copy it from the `**Host**` field on the `**Connect**` tab in the BigAnimal portal.
- ``<port>`` — The port number that the database listener is monitoring. You can copy it from the `**Port**` field on the `**Connect**` tab in the BigAnimal portal.
- ``<database_id>`` — The name of the target database. Set this to the name of the database in your cluster that you want to use as your migration target database. The name of the default database for your cluster is shown in the `**Dbname**` field on the `**Connect**` tab in the BigAnimal portal. Often a separate database is created for use as the migration target.
- ``TARGET_DB_USER`` — Specifies the name of a privileged database user. You can copy it from the `**User**` field on the `**Connect**` tab in the BigAnimal portal.
- ``TARGET_DB_PASSWORD`` — Contains the password of the specified user.
- ``sslmode`` — Either "require" or "verify-full". See [Recommended settings for SSL mode](../../biganimal/latest/using_cluster/02_connect_to_cluster/#recommended-settings-for-ssl-mode). Listed at the end of the `**Service URI**` value on the `**Connect**` tab in the BigAnimal portal.

5.4 Defining an Oracle URL

Migration Toolkit helps with migration from an Oracle database to a PostgreSQL or Advanced Server database. When migrating from Oracle, you must specify connection specifications for the Oracle source database in the toolkit.properties file. The connection information must include:

- The URL of the Oracle database
- The name of a privileged user
- The password associated **with** the specified user

When migrating from an Oracle database, SRC_DB_URL must contain a JDBC URL, specified in one of two forms. The first form is:

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

The second form is:

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

An Oracle URL contains the following information:

- ``jdbc`` — The protocol is always ``jdbc``.
- ``oracle`` — The subprotocol is always ``oracle``.
- ``thin`` — The driver type. Specify a driver type of ``thin``.
- ``<host_name>`` — The name or IP address of the host where the Oracle server is running.
- ``<port>`` — The port number that the Oracle database listener is monitoring.
- ``<database_id>`` — The database SID of the Oracle database.
- ``<service_name>`` — The name of the Oracle service.
- ``SRC_DB_USER`` — Specifies the name of a privileged Oracle user. The Oracle user needs DBA privilege to migrate objects from Oracle to Advanced Server. The DBA privilege can be granted to the Oracle user with the Oracle ``GRANT DBA TO user`` command to ensure all of the desired database objects are migrated.
- ``SRC_DB_PASSWORD`` — Contains the password of the specified user.

5.5 Defining a MySQL URL

Migration Toolkit helps with migration from a MySQL database to an Advanced Server or PostgreSQL database. When migrating from MySQL, you must specify connection specifications for the MySQL source database in the `toolkit.properties` file. The connection information must include:

- The URL of the source database
- The name of a privileged user
- The password associated **with** the specified user

When migrating from MySQL, `SRC_DB_URL` takes the form of a JDBC URL. For example:

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

The URL contains the following information:

- ``jdbc`` — The protocol is always ``jdbc``.
- ``mysql`` — The subprotocol is always ``mysql``.
- ``<host_name>`` — The name or IP address of the host where the source server is running.

- ``[<port>]`` — The port number that the MySQL database listener is monitoring.
- ``<database_id>`` — The name of the source database.
- ``SRC_DB_USER`` — Specifies the name of a privileged MySQL user.
- ``SRC_DB_PASSWORD`` — Contains the password of the specified user.

!!! Note * - If datatype `tinyInt(1)` is used to store byte values other than 0 and 1 in the MySQL source database, make sure to append the optional parameter `?tinyInt1isBit=false` in the MySQL Connector/J JDBC Driver URL.

- – Due to a bug in the MySQL Connector/J JDBC Driver 8.0.26, the migration of foreign key constraints fails in certain cases. EDB recommends that you don't use this driver for migrating data using Migration Toolkit. You can use MySQL Connector/J JDBC Driver 8.0.25 driver instead.
- For detailed information about this bug, see the [MySQL bug report](<https://bugs.mysql.com/bug.php?id=95280>).

TINYINT(1) is mapped to BIT(1) in PostgreSQL/EDB Postgres Advanced Server (which might not be expected in some cases). But as the MySQL JDBC driver reports it as BIT(1), Migration Toolkit maps it to BIT(1) in PostgreSQL/EDB Postgres Advanced Server.

Unlike TINYINT(1) that can hold the values -128 to 127, BIT(1) can hold only 0 and 1. So if you're storing any value other than 0 and 1 in this field, you might need to change the default behavior of the MySQL JDBC driver so that the driver reports it as TINYINT.

To change this behavior, set the following URL option in the connection string:

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

In this case, the JDBC driver reports TINYINT(1) as TINYINT and is mapped to SMALLINT in the target PostgreSQL/EDB Postgres Advanced Server.

5.6 Defining a Sybase URL

Migration Toolkit helps with migration from a Sybase database to an Advanced Server database. When migrating from Sybase, you must specify connection specifications for the Sybase source database in the `toolkit.properties` file. The connection information must include:

- The URL of the source database
- The name of a privileged user
- The password associated **with** the specified user

When migrating from Sybase, SRC_DB_URL takes the form of a JTDS URL. For example:

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

A Sybase URL contains the following information:

- ``jdbc`` — The protocol is always ``jdbc``.

- ``jtds`` — The driver name is always ``jtds``.
- ``sybase`` — The server type is always ``sybase``.
- ``<host_name>`` — The name or IP address of the host where the source server is ↪running.
- ``<port>`` — The port number that the Sybase database listener is monitoring.
- ``<database_id>`` — The name of the source database.
- ``SRC_DB_USER`` — Specifies the name of a privileged Sybase user.
- ``SRC_DB_PASSWORD`` — Contains the password of the specified user.

5.7 Defining a SQL Server URL

Migration Toolkit helps with migration from an SQL Server database to a PostgreSQL or Advanced Server database. Migration Toolkit supports migration of the following object definitions:

- Schemas
- Tables
- Table data
- Constraints
- Indexes

Migration Toolkit reads connection specifications for the source database from the `toolkit.properties` file. The connection information must include:

- The URL of the source database
- The name of a privileged user
- The password associated with the specified user

If you're connecting to a SQL Server database, `SRC_DB_URL` takes the form of a JTDS URL. For example:

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

A SQL server URL contains the following information:

- ``jdbc`` — The protocol is always ``jdbc``.

- ``jtcs`` — The driver name is always ``jtcs``.
- ``sqlserver`` — The server type is always ``sqlserver``.
- ``<server_name>`` — The name or IP address of the host where the source server is running.
- ``<port>`` — The port number that the source database listener is monitoring.
- ``<database_id>`` — The name of the source database.
- ``SRC_DB_USER`` — Specifies the name of a privileged SQL Server user.
- ``SRC_DB_PASSWORD`` — Contains the password of the specified user.

INVOKING MIGRATION TOOLKIT

After installing Migration Toolkit and specifying connection properties for the source and target databases in the `toolkit.properties` file, Migration Toolkit is ready to perform migrations.

The Migration Toolkit executable is named `runMTK.sh` on Linux systems and `runMTK.bat` on Windows systems. On a Linux system, the executable is located in:

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

On Windows, the executable is located in:

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

!!! Note * If the following error appears upon invoking the Migration Toolkit, check the file permissions of the `toolkit.properties` file.

```
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.
```

The operating system user account running the Migration Toolkit must be the owner of the `toolkit.properties` file with a minimum of read permission on the file. In addition, there must be no permissions of any kind for group and other users. The following is an example of the recommended file permissions, where user `enterprisedb` is running the Migration Toolkit.

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

6.1 Importing character data with embedded binary zeros (NULL characters)

Migration Toolkit supports importing a column with a value of NULL. However, Migration Toolkit doesn't support importing NULL character values (embedded binary zeros 0x00) with the JDBC connection protocol. If you're importing data that includes the NULL character, use the `-replaceNullChar` option to replace the NULL character with a single, non-NULL, replacement character.

!!! Note * - MTK implicitly replaces NULL characters with an empty string. - The `-replaceNullChar` option doesn't work with the `-copyViaDBLinkOra` option.

Once the data is migrated, use a SQL statement to replace the character specified by `-replaceNullChar` with binary zeros.

6.2 Migrating a schema from Oracle

Unless specified in the command line, Migration Toolkit expects the source database to be Oracle and the target database to be EDB Postgres Advanced Server. To migrate a complete schema on Linux, navigate to the executable and invoke the following command:

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

To migrate a complete schema on Windows, navigate to the executable and invoke the following command:

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

Where:

<schema_name> is the name of the schema in the source database specified in the `toolkit.properties` file that you want to migrate. You must include at least one `schema_name` value.

!!! Note * - When the default database user of a migrated schema is automatically migrated, the custom profile of the default database user is also migrated if a custom profile exists. A custom profile is a user-created profile. For example, custom profiles exclude Oracle profiles `DEFAULT` and `MONITORING_PROFILE`. - PostgreSQL default rows are limited to 8 KB in size. This means that each table row must fit into a single 8 KB block, Otherwise, an error occurs indicating, for example, that we can create a table with 1600 columns of INT and insert data for all the columns. However, we can't do the same with BIGINT columns because INT is stored as 4 bytes, but each BIGINT requires more space (8 bytes). For more information, see [PostgreSQL Limits](#) in the PostgreSQL documentation.

You can migrate multiple schemas by following the command name with a comma-delimited list of schema names.

On Linux, execute the following command:

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

On Windows, execute the following command:

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

6.3 Migrating from a non-Oracle source database

If you don't specify a source database type and a target database type, Postgres assumes the source database is Oracle and the target database is EDB Postgres Advanced Server.

To invoke Migration Toolkit, open a command window, navigate to the executable, and invoke the following command:

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

```
-sourcedbtype <source_type>
```

<source_type> specifies the server type of the source database. <source_type> isn't case sensitive. By default, <source_type> is `oracle`. `source_type` can be one of the following values:

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

continues on next page

Table 1 – continued from previous page

To migrate from:	Specify:

`-targetdbtype <target_type>`

`<target_type>` specifies the server type of the target database. `<target_type>` isn't case sensitive. By default, `<target_type>` is `enterprisedb`. `<target_type>` can be one of the following values:

To migrate to:	Specify:
EDB Postgres Advanced Server	<code>enterprisedb</code>
PostgreSQL	<code>postgres</code> or <code>postgresql</code>

`<schema_name>`

`<schema_name>` is the name of the schema in the source database specified in the `toolkit.properties` file that you want to migrate. You must include at least one `<schema_name>` value.

The following example migrates a schema (table definitions and table content) named `HR` from a MySQL database on a Linux system to an EDB Postgres Advanced Server host. The command includes the `-sourcedbtype` and `-targetdbtype` options.

On Linux, use the following command:

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

On Windows, use the following command:

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

You can migrate multiple schemas from a source database by including a comma-delimited list of schemas at the end of the Migration Toolkit command. The following example migrates multiple schemas (named `HR` and `ACCTG`) from a MySQL database to a PostgreSQL database.

On Linux, use the following command:

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

On Windows, use the following command:

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


MIGRATION TOOLKIT COMMAND OPTIONS

Append migration options when you run Migration Toolkit to control details of the migration. For example, to migrate all schemas in a database, append the `-allSchemas` option to the command:

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

!!! Note * - The `-allSchemas` parameter is supported only for the Oracle, EDB Postgres Advanced Server, and PostgreSQL source database. It isn't supported for Sybase, MS SQL Server, and MySQL source databases. - Migration Toolkit disables the user-created triggers while migrating the data. However, as PostgreSQL/EDB Postgres Advanced Server doesn't allow disabling inherited child partition triggers using the `DISABLE TRIGGER USER` clause in `ALTER TABLE`, the triggers activate during data migration on partition tables, which can cause unexpected results.

The command options that work with Migration Toolkit are grouped by their behavior, as shown in the table.

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,
[Migration options](#migration-options)	-truncLoad, -enableConstBeforeDataLoad, -retryCount, -safeMode, -fastCopy, -analyze, vacuumAnalyze, -replaceNullChar, -copyDelimiter, -batchSize, -cpBatchSize, -lobBatchSize, -fetchSize, -filterProp, -customColTypeMapping, -customColTypeMappingFile

continues on next page

Table 1 – continued from previous page

Feature	Relevant options
[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

7.1 Offline migration options

If you specify the `-offlineMigration` option in the command line, Migration Toolkit performs an offline migration. During an offline migration, Migration Toolkit reads the definition of each selected object and creates an SQL script that, when executed at a later time, replicates each object in Postgres.

!!! Note * The following examples invoke the Migration Toolkit in Linux. To invoke Migration Toolkit in Windows, use the `runMTK.bat` command instead of the `runMTK.sh` command.

To perform an offline migration of both schema and data, specify the `-offlineMigration` keyword, followed by the schema name:

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

Each database object definition is saved in a separate file with a name derived from the schema name and object type in your home folder. To specify an alternative file destination, include a directory name after the `-offlineMigration` option:

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

To perform an offline migration of schema objects only (creating empty tables), specify the `-schemaOnly` keyword in addition to the `-offlineMigration` keyword when invoking Migration Toolkit:

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

To perform an offline migration of data only, omitting any schema object definitions, specify the `-dataOnly` keyword and the `-offlineMigration` keyword when invoking Migration Toolkit:

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

By default, data is written in COPY format. To write the data in a plain SQL format, include the `-safeMode` keyword:

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

By default, when you perform an offline migration that contains table data, a separate file is created for each table. To create a single file that contains the data from multiple tables, specify the `-singleDataFile` keyword:

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

!!! Note * The `-singleDataFile` option is available only when migrating data in a plain SQL format. You must include the `-safeMode` keyword if you include the `-singleDataFile` option.

7.1.1 Executing offline migration scripts

You can use the `edb-psql` or `psql` command line to execute the scripts generated during an offline migration. The following example describes restoring a schema (named `hr`) into a new database (named `acctg`) stored in EDB Postgres Advanced Server.

1. Use the `createdb` command to create the `acctg` database, into which you'll restore the migrated database objects:
 - `createdb -U enterprisedb acctg`
2. Connect to the new database with `edb-psql`:
 - `edb-psql -U enterprisedb acctg`
3. Use the `\i` meta-command to invoke the migration script that creates the object definitions:
 - `acctg=# \i ./mtk_hr_ddl.sql`
4. If the `-offlineMigration` command included the `-singleDataFile` keyword, the `mtk_hr_data.sql` script will contain the commands required to re-create all of the objects in the new target database. Populate the database with the command:
 - `acctg=# \i ./mtk_hr_data.sql`

7.2 Import options

By default, Migration Toolkit assumes the source database is Oracle and the target database is EDB Postgres Advanced Server. Include the `-sourcedbtype` and `-targetdbtype` keywords to specify a nondefault source or target database.

By default, Migration Toolkit imports both the data and the object definition when migrating a schema. Alternatively, you can choose to import either the data or the object definitions.

`-sourcedbtype <source_type>`

The `-sourcedbtype` option specifies the source database type. For `source_type`, use one of the following values: `mysql`, `oracle`, `sqlserver`, `sybase`, `postgresql` or `enterprisedb`. `source_type` is not case sensitive. By default, `source_type` is `oracle`.

`-targetdbtype <target_type>`

The `-targetdbtype` option specifies the target database type. For `target_type`, use one of the following values: `enterprisedb`, `postgres`, or `postgresql`. `target_type` is not case sensitive. By default, `target_type` is `enterprisedb`.

`-schemaOnly`

This option imports the schema definition and creates all selected schema objects in the target database. You can't use this option with the `-dataOnly` option.

`-dataOnly`

This option copies the data only. When used with the `-tables` option, Migration Toolkit imports data only for the selected tables. You can't use this option with the `-schemaOnly` option.

7.3 Schema creation options

By default, Migration Toolkit imports the source schema objects or data into a schema of the same name. If the target schema doesn't exist, Migration Toolkit creates a new schema. Alternatively, you can specify a custom schema name by using the `-targetSchema` option. You can choose to drop the existing schema and create a new schema using the following option:

`-dropSchema [true|false]`

With this option set to `true`, Migration Toolkit drops the existing schema and any objects in that schema and creates a new schema. By default, `-dropSchema` is `false`.

`-targetSchema <schema_name>`

Use the `-targetSchema` option to specify the name of the migrated schema. If you're migrating multiple schemas, specify a name for each schema in a comma-separated list with no intervening space characters. Without the `-targetSchema` option, the name of the new schema is the same as the name of the source schema.

You can't specify `information-schema`, `dbo`, `sys`, or `pg_catalog` as target schema names. These schema names are reserved for meta-data storage in EDB Postgres Advanced Server.

7.4 Schema object selection options

Use the following options to select specific schema objects to migrate:

`-allTables`

Import all tables from the source schema.

`-tables <table_list>`

Import the selected tables from the source schema. `table_list` is a comma-separated list of table names with no intervening space characters (e.g., `-tables emp, dept, acctg`).

`-constraints`

Import the table constraints. This option is valid only when importing an entire schema or when you specify the `-allTables` or `-tables <table_list>` options.

`-ignoreCheckConstFilter`

By default, Migration Toolkit doesn't implement migration of check constraints and default clauses from a Sybase database. Include the `-ignoreCheckConstFilter` parameter when specifying the `-constraints` parameter to migrate constraints and default clauses from a Sybase database.

`-skipCKConst`

Omit the migration of check constraints. This option is useful when migrating check constraints that are based on built-in functions in the source database that aren't supported in the target database.

This option is valid only when importing an entire schema or when the `-allTables` or `-tables <table_list>` options are specified.

`-skipFKConst`

Omit the migration of foreign key constraints. This option is valid only when importing an entire schema or when the `-allTables` or `-tables <table_list>` options are specified.

`-skipColDefaultClause`

Omit the migration of the column `DEFAULT` clause.

-indexes

Import the table indexes. This option is valid when importing an entire schema or when the `-allTables` or `-tables <table_list>` option is specified.

-triggers

Import the table triggers. This option is valid when importing an entire schema or when the `allTables` or `-tables <table_list>` option is specified.

-allViews

Import the views from the source schema. This option migrates dynamic and materialized views from the source. Oracle and Postgres materialized views are supported.

-views <view_list>

Import the specified materialized or dynamic views from the source schema. Oracle and Postgres materialized views are supported. `view_list` is a comma-separated list of view names with no intervening space characters (e.g., `-views all_emp,mgmt_list,acct_list`).

-allSequences

Import all sequences from the source schema.

-sequences <sequence_list>

Import the selected sequences from the source schema. `<sequence_list>` is a comma-separated list of sequence names with no intervening space characters.

-allProcs

Import all stored procedures from the source schema.

-procs <procedures_list>

Import the selected stored procedures from the source schema. `procedures_list` is a comma-separated list of procedure names with no intervening space characters.

-allFuncs

Import all functions from the source schema.

-funcs <function_list>

Import the selected functions from the source schema. `function_list` is a comma-separated list of function names with no intervening space characters.

-checkFunctionBodies [true/false]

When `false`, disables validation of the function body during function creation. Disabling this validation avoids errors if the function contains forward references. The default value is `true`.

-allPackages

Import all packages from the source schema.

-packages <package_list>

Import the selected packages from the source schema. `package_list` is a comma-separated list of package names with no intervening space characters.

-allDomains

Import all domain, enumeration, and composite types from the source database. This option is valid only when both the source and target are stored on a Postgres host.

-allQueues

Import all queues from the source schema. These are queues created and managed by the DBMS_AQ and DBMS_AQADM built-in packages. When Oracle is the source database, you must also specify the `-objectTypes` option. When EDB Postgres Advanced Server is the source database, you must also specify the `-allDomains` and `-allTables` options. Oracle and EDB Postgres Advanced Server queues are supported.

`-queues <queue_list>`

Import the selected queues from the source schema. `queue_list` is a comma-separated list of queue names with no intervening space characters. These are queues created and managed by the DBMS_AQ and DBMS_AQADM built-in packages. When Oracle is the source database, you must also specify the `-objectTypes`. When EDB Postgres Advanced Server is the source database, you must also specify `-allDomains` and `-allTables`. Oracle and EDB Postgres Advanced Server queues are supported.

`-allRules`

Import all rules from the source database. This option is valid only when both the source and target are stored on a Postgres host.

7.5 Migration options

Use the migration options to control the details of the migration process.

`-truncLoad`

Truncate the data from the table before importing new data. Use this option with the `-dataOnly` option.

`-enableConstBeforeDataLoad`

Include the `-enableConstBeforeDataLoad` option if a nonpartitioned source table is mapped to a partitioned table. This option enables all triggers on the target table, including any triggers that redirect data to individual partitions, before the data migration. `-enableConstBeforeDataLoad` is valid only if you also specify the `-truncLoad` parameter.

`-retryCount [<value>]`

If you're performing a multiple-schema migration, objects that fail to migrate during the first migration attempt due to cross-schema dependencies might successfully migrate during a later migration. Use the `-retryCount` option to specify the number of attempts for Migration Toolkit to make to migrate an object that failed during an initial migration attempt. Specify a value greater than 0. The default value is 2.

`-safeMode`

If you include the `-safeMode` option, Migration Toolkit commits each row as migrated. If the migration fails to transfer all records, rows inserted prior to the point of failure remain in the target database.

`-fastCopy`

Including the `-fastCopy` option specifies for Migration Toolkit to bypass WAL logging to perform the COPY operation in an optimized way. It is disabled by default. If you choose to use the `-fastCopy` option, you might not be able to recover the migrated data in the target database if the migration is interrupted.

`-replaceNullChar <value>`

The Migration Toolkit supports importing a column with a value of NULL. However, the Migration Toolkit doesn't support importing NULL character values (embedded binary zeros 0x00) with the JDBC connection protocol. If you're importing data that includes the NULL character, use the `-replaceNullChar` option to replace the NULL character with a single, non-NULL replacement character. Don't enclose the replacement character in quotes or apostrophes.

Once the data is migrated, use a SQL statement to replace the character specified by `-replaceNullChar` with binary zeros.

`-analyze`

Include the `-analyze` option to invoke the Postgres `ANALYZE` operation against a target database. The optimizer consults the statistics collected by the `ANALYZE` operation, using the information to construct efficient query plans.

`-vacuumAnalyze`

Include the `-vacuumAnalyze` option to invoke both the `VACUUM` and `ANALYZE` operations against a target database. The optimizer consults the statistics collected by the `ANALYZE` operation, using the information to construct efficient query plans. The `VACUUM` operation reclaims any storage space occupied by dead tuples in the target database.

`-copyDelimiter`

Specify a single character to use as a delimiter in the `COPY` command when loading table data. The default value is `'\t'` (tab).

`-batchSize`

Specify the batch size of bulk inserts. Valid values are 1 to 1000. The default batch size is 1000. Reduce the value of `-batchSize` if Out of Memory exceptions occur.

`-cpBatchSize`

Specify the batch size in MB to use in the `COPY` command. Any value greater than 0 is valid. The default batch size is 8 MB.

`-lobBatchSize`

Specify the number of rows to load in a batch for LOB data types. The data migration for a table containing a large object type (LOB) column such as `BYTEA`, `BLOB`, or `CLOB`, etc., is performed one row at a time by default. This is to avoid an out-of-heap-space error in case an individual LOB column holds hundreds of megabytes of data. In case the LOB column average data size is at a lower end, you can customize the LOB batch size by specifying the number of rows in each batch with any value greater than 0.

`-fetchSize`

Use the `-fetchSize` option to specify the number of rows fetched in a result set. If the designated `-fetchSize` is too large, you might encounter Out of Memory exceptions. Include the `-fetchSize` option to avoid this pitfall when migrating large tables. The default fetch size is specific to the JDBC driver implementation and varies by database.

MySQL users note: By default, the MySQL JDBC driver fetches all of the rows in a table into the Migration Toolkit in a single network round trip. This behavior can easily exceed available memory for large tables. If you encounter an out-of-heap-space error, specify `-fetchSize 1` as a command line argument to force Migration Toolkit to load the table data one row at a time.

`-filterProp <file_name>`

`<file_name>` specifies the name of a file that contains constraints in key=value pairs. Each record read from the database is evaluated against the constraints. Those that satisfy the constraints are migrated. The left side of the pair lists a table name. The table name must not be schema qualified. The right side specifies a condition that must be true for each row migrated. For example, including the following constraints in the property file

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

migrates only those countries with a `country_id` value that is not equal to `AR`. This constraint applies to the `countries` table.

You can also specify conditions for multiple tables. However, the condition for each table must be on a new line in the property file.

Example:

Following entries in the properties file migrate only the relevant data from `EMPLOYEES` and the `DEPARTMENTS` tables.

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

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

```
-customColTypeMapping <column_list>
```

Use custom type mapping to change the data type of migrated columns. The left side of each pair specifies the columns with a regular expression. The right side of each pair names the data type for that column to assume. You can include multiple pairs in a semi-colon separated list for `<column_list>`. For example, to map any column whose name ends in `ID` to type `INTEGER`, use the following custom mapping entry:

```
.*ID=INTEGER
```

Custom mapping is applied to all table columns that match the criteria unless the column is table qualified.

The `'\\'` characters act as an escape string. Since `'.'` is a reserved character in regular expressions, on Linux use `'\\.'` to represent the `'.'` character. For example, to use custom mapping to select rows from the `EMP_ID` column in the `EMP` table, specify the following custom mapping entry:

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

On Windows, use `'\.'` to represent the `'.'` character:

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

```
-customColTypeMappingFile <property_file>
```

You can include multiple custom type mappings in `<property_file>`. Specify each entry in the file on a separate line, in a key=value pair. The left side of each pair selects the columns with a regular expression. The right side of each pair names the data type for that column to assume.

7.6 Migration options for parallel data loading

When dealing with a relatively large number of tables or large size tables, normal data migration takes a long time. As a solution to this problem, Migration Toolkit has a parallel data loading algorithm that allows migrating data in parallel at the schema level and at the table level.

Parallel data migration at the schema level uses multiple threads to migrate several tables at the same time. Parallel data loading at the table level uses multiple threads to migrate a single table from the source database to the target database more efficiently and in a shorter time.

!!! Note * - Parallel loading is useful for large tables only. For small tables, parallel threads might take more time. Thus we recommend using parallel data loading at the table level for relatively large tables.

- – Parallel data loading doesn't apply to tables without a primary/unique key.
- – `fastCopy` parameter can't be used with parallel data loading.
- – Parallel loading functionality applies only to the source Oracle, EDB Postgres Advanced Server, and PostgreSQL databases.

Following options control the way MTK migrates data in parallel.

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

Use the `-loaderCount` option to specify the number of parallel threads available in the pool to perform data load in parallel. This option is particularly useful if the host system that is running Migration Toolkit has high-end CPU and RAM resources. While `value` can be any nonzero, positive number, we recommend that `value` not exceed the number of CPU cores. For example, a dual-core CPU has an optimal value of 2. The default is 1.

!!! Note * When multiple threads are used, depending on the actual number of threads and table data size, you might need to adjust the memory heap size for the Migration Toolkit.

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

Use the `-parallelLoadRowLimit` option to specify the minimum number of rows required in a table to perform data load in parallel at the table level. The `<value>` parameter must be greater than 0. The default is 100,000.

For example, if the value of `-parallelLoadRowLimit` is 10,000, only tables with the number of rows 10,000 and above are loaded in parallel at the table level. The other tables migrate in sequential mode or at the schema level.

`-tableLoaderLimit [<value>]`

Use the `-tableLoaderLimit` option to specify the maximum number of threads to assign from the thread pool to load a single table data in parallel chunks at the table level. The `<value>` parameter must not be greater than the value of `-loaderCount`. The default value is equal to the value of the `-loaderCount` option. This option is implicitly changed based on the nondefault custom value of `-loaderCount`.

For example, if the `-loaderCount` is set to 4 and `-tableLoaderLimit` isn't specified, the `-tableLoaderLimit` is also set to 4. Hence, if all threads aren't meant to be used for a single table load, reduce the `-tableLoaderLimit` to a lower value.

7.6.1 High-level overview of the parallel data-loading algorithm

The algorithm works in three steps:

1. A list of tables and table chunks is created from the ordered list of tables to be migrated, sorted by the table name. The list is created based on the `-parallelLoadRowLimit` and `-tableLoaderLimit` option values.
2. A thread pool is created based on the `-loaderCount` option value.
3. The parallel data-loading algorithm creates tasks for each table/chunk from the list of tables and table chunks and executes them in the thread pool. As many tasks as the number of threads are executed in parallel. As soon as a task execution completes, its thread starts executing the next task from the list. Parallel data loading completes as soon as the last task from the list completes.

Example:

In this scenario, migrate four tables—`tab1`, `tab2`, `tab3`, `tab4`—with parallel data loading:

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

Table `tab1` has 50,000 rows, table `tab2` has 9,500 rows, table `tab3` has 50,001 rows and table `tab4` has 9,999 rows. The list of tables for step 1 is `tab1`, `tab2`, `tab3`, and `tab4`.

Only tables `tab1` and `tab3` qualify for parallel loading on table level (`-parallelLoadRowLimit 10000`).

Since the value of the `-tableLoaderLimit` option is 2, table `tab1` is loaded in two chunks, `tab1_chunk1` and `tab1_chunk2`. The number of rows in each chunk is determined based on the `totalRowCount/tableLoaderLimit`. In this case, both chunks have 25,000 rows ($50,000/2$).

Table `tab3` is also loaded in two chunks, `tab3_chunk1` and `tab3_chunk2`. As the total number of rows in table `tab3` is odd (50,001), the first chunk accommodates an additional row, so `tab3_chunk1` has 25,001 rows, and `tab3_chunk2` has 25,000 rows.

Here is the the list of tables and table chunks created:

```
*  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)
```

Thread pool with the four threads T1, T2, T3, T4 is created.

Tasks migrating tables/chunks from the list created in step 1 are executed in the thread pool:

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

Thread T3 finishes first and starts executing tab3_chunk2, the next chunk in the list:

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

Thread T1 finishes second and starts executing tab4:

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

This way, four threads are used to migrate four tables in parallel at the mixed schema and table level.

If you change command options to set `-tableLoaderLimit` to 1:

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

You can have pure schema level parallelism. Each table is processed simultaneously in one thread:

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

7.6.2 System resource recommendations and constraints**

Choose the number of threads depending on the CPU and RAM resources available on the host system running MTK. As there is a certain overhead on the source database in terms of CPU, IO, and disk utilization, exceeding parallel thread count beyond a certain threshold degrades the performance of MTK.

7.7 Oracle-specific options

The following options apply only when the source database is Oracle.

-objectTypes

Import the user-defined object types from the schema list specified at the end of the `runMTK.sh` command.

-allUsers

Import all users and roles from the source database. The `-allUsers` option is supported only when migrating from an Oracle database to an EDB Postgres Advanced Server database.

-users <user_list>

Import the selected users or roles from the source Oracle database. <user_list> is a comma-separated list of user/role names with no intervening space characters (e.g., `-users MTK, SAMPLE, acctg`). The `-users` option is supported only when migrating from an Oracle database to an EDB Postgres Advanced Server database.

-allProfiles

Import all custom (that is, user-created) profiles from the source database. Other Oracle noncustom profiles such as `DEFAULT` and `MONITORING_PROFILE` aren't imported.

For the imported profiles, only the following password parameters associated with the profiles are imported:

`FAILED_LOGIN_ATTEMPTS`

`PASSWORD_LIFE_TIME`

`PASSWORD_REUSE_TIME`

`PASSWORD_REUSE_MAX`

`PASSWORD_LOCK_TIME`

`PASSWORD_GRACE_TIME`

`PASSWORD_VERIFY_FUNCTION`

All other profile parameters, such as the Oracle resource parameters, aren't imported. The Oracle database user specified by `SRC_DB_USER` must have `SELECT` privilege on the Oracle data dictionary view `DBA_PROFILES`.

!!! Note * The `-allProfiles` option is supported only when migrating from an Oracle database to an EDB Postgres Advanced Server database.

-profiles <profile_list>

Import the selected, custom (that is, user-created) profiles from the source Oracle database. `profile_list` is a comma-separated list of profile names with no intervening space characters (e.g., `-profiles ADMIN_PROFILE, USER_PROFILE`). Oracle noncustom profiles such as `DEFAULT` and `MONITORING_PROFILE` aren't imported.

As with the `-allProfiles` option, only the password parameters are imported. The Oracle database user specified by `SRC_DB_USER` must have `SELECT` privilege on the Oracle data dictionary view `DBA_PROFILES`.

!!! Note * The `-profiles` option is supported only when migrating from an Oracle database to an EDB Postgres Advanced Server database.

-importPartitionAsTable <table_list>

Include the `-importPartitionAsTable` parameter to import the contents of a partitioned table that resides on an Oracle host into a single nonpartitioned table. `table_list` is a comma-separated list of table names with no intervening space characters (e.g., `-importPartitionAsTable emp,dept,acctg`).

-copyViaDBLinkOra

The `dblink_ora` module provides EDB Postgres Advanced Server-to-Oracle connectivity at the SQL level. `dblink_ora` is bundled and installed as part of the EDB Postgres Advanced Server database installation. `dblink_ora` uses the `COPY` API method to transfer data between databases. This method is considerably faster than the `JDBC COPY` method.

This example uses the `dblink_ora` `COPY` API to migrate all tables from the `HR` schema:

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

The target EDB Postgres Advanced Server database must have `dblink_ora` installed and configured. For information about `dblink_ora`, see [Database Compatibility for Oracle Developers](#).

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

Choose this option to migrate Oracle database links. The password information for each link connection in the source database is encrypted so, unless specified, the dummy password edb is substituted.

To migrate all database links using edb as the dummy password for the connected user:

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

You can alternatively specify the password for each of the database links through a comma-separated list of name=value pairs with no intervening space characters. Specify the link name on the left side of the pair and the password value on the right side.

To migrate all database links with the actual passwords specified on the command line:

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

Migration Toolkit migrates only the database link types that are currently supported by EnterpriseDB. This includes fixed user links of public and private type.

-allSynonyms

Include the **-allSynonyms** option to migrate all public and private synonyms from an Oracle database to an EDB Postgres Advanced Server database. If a synonym with the same name already exists in the target database, the existing synonym is replaced with the migrated version.

-allPublicSynonyms

Include the **-allPublicSynonyms** option to migrate all public synonyms from an Oracle database to an EDB Postgres Advanced Server database. If a synonym with the same name already exists in the target database, the existing synonym is replaced with the migrated version.

-allPrivateSynonyms

Include the **-allPrivateSynonyms** option to migrate all private synonyms from an Oracle database to an EDB Postgres Advanced Server database. If a synonym with the same name already exists in the target database, the existing synonym is replaced with the migrated version.

-useOraCase

Include the **-useOraCase** option to preserve the Oracle default, uppercase naming convention for all database objects when migrating from an Oracle database to an EDB Postgres Advanced Server database.

The uppercase naming convention is preserved for tables, views, sequences, procedures, functions, triggers, packages, and so on. For these database objects, the uppercase naming convention is applied to:- The names of the database objects- The column names, key names, index names, constraint names, and so on, of the tables and views- The SELECT column list for a view- The parameter names that are part of the procedure or function header

!!! Note * In the procedural code body of a procedure, function, trigger or package, you might have to manually edit identifier references for the program to execute without an error. Such corrections are in regard to the proper case conversion of identifier references that might have occurred.

!!! Note * When you specify the **-useOraCase** option, you might need to specify the **-skipUserSchemaCreation** option as well. For information, see the description of the **-skipUserSchemaCreation** option.

The default behavior of the Migration Toolkit without the **-useOraCase** option is that database object names are extracted from Oracle without enclosing quotation marks unless the database object was explicitly created in Oracle with enclosing quotation marks. The following is a portion of a table command generated by the Migration Toolkit with the **-offlineMigration** option:

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

(continues on next page)

(continued from previous page)

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

When you then migrate this table and create it in EDB Postgres Advanced Server, all unquoted object names are converted to lowercase letters, so the table appears in EDB Postgres Advanced Server as follows:

```
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)
```

If your EDB Postgres Advanced Server applications are referencing the migrated database objects using quoted uppercase identifiers, the applications fail since the database object names are now in lower case.

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

If your application uses quoted uppercase identifiers, perform the migration with the `-useOraCase` option. The DDL encloses all database object names in quotes:

```
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");
```

When you then migrate this table and create it in EDB Postgres Advanced Server, all object names are maintained in uppercase letters, so the table appears in EDB Postgres Advanced Server as follows:

```
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")
```

Applications can then access the object using quoted uppercase names.

```
useoracase=# SELECT * FROM "DEPT";
DEPTNO | DNAME | LOC
-----+-----+-----
```

(continues on next page)

(continued from previous page)

10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
(4 rows)		

-skipUserSchemaCreation

When an Oracle user is migrated, a role (that is, a user name) is created in the target database server for the Oracle user if the role doesn't already exist. The role name is created in lowercase letters. When a new role is created, a schema with the same name is also created in lowercase letters.

Specifying the **-skipUserSchemaCreation** option prevents the automatic schema creation for a migrated Oracle user name. This option is particularly useful when the **-useOraCase** option is specified to prevent creating two schemas with the same name where only the case is different. Specifying the **-useOraCase** option results in creating a schema in the Oracle naming convention of uppercase letters for the source schema specified following the options list when Migration Toolkit is invoked.

Thus, if the **-useOraCase** option is specified without the **-skipUserSchemaCreation** option, the target database results in having two identically named schemas with one in lowercase letters and the other in uppercase letters. If the **-useOraCase** option is specified along with the **-skipUserSchemaCreation** option, the target database has only the schema in uppercase letters.

7.8 Miscellaneous options

Use these migration options view Migration Toolkit help and version information. You can also use these options to control Migration Toolkit feedback and logging options.

-help

Display the application command-line usage information.

-logDir <log_path>

Include this option to specify where to write the log files. **<log_path>** is the location for saving log files. By default, on Linux log files are written to:

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

On Windows, the log files are saved to:

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

-logFileCount <file_count>

Include this option to specify the number of files used in log file rotation. Specify a value of **0** to disable log file rotation and create a single log file. This file is truncated when it reaches the value specified using the **logFileSize** option. **<file_count>** must be greater than or equal to 0. The default is 20.

-logFileSize <file_size>

Include this option to specify the maximum file size limit in MB before rotating to a new log file. **file_size** must be greater than 0. The default is 50.

-logBadSQL

Include this option to have the schema definition (DDL script) of any failed objects saved to a file. The file is saved under the same path used for the error logs and is named in the format:

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

Where `schema_name` is the name of the schema and `timestamp` is the timestamp of the Migration Toolkit run.

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

Display application log messages on standard output. By default, verbose is on.

```
-version
```

Display the Migration Toolkit version.

7.9 Example

This example performs an Oracle to EDB Postgres Advanced Server migration.

The following is the content of the `toolkit.properties` file.

```
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
```

The following command invokes Migration Toolkit:

```
$ ./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
-- ^
```

(continues on next page)

(continued from previous page)

```

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" ...
["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_CK
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

```

(continues on next page)

(continued from previous page)

```

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
-----
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)
*****

```

Skipped and unsupported database objects are omitted. The migration information is summarized in the Migration Summary at the end of the run.

MIGRATION ERRORS

During the migration process, the migration summary displays the progress of the migration. If an error occurs, the summary displays information about the error. The migration summary is also written to a log file.

On Linux, the default location for the log files is:

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

On Windows, the log files are saved to:

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

You can specify an alternative log file location with the `-logdir <log_path>` option in Migration Toolkit.

8.1 Connection errors

Migration Toolkit uses information from the `toolkit.properties` file to connect to the source and target databases. Most of the connection errors that occur when using Migration Toolkit are related to the information specified in the `toolkit.properties` file.

8.1.1 Invalid username/password

- When I try to perform a migration from an Oracle database with Migration Toolkit, I get the following error:*

```
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.
```

To resolve this error, edit the `toolkit.properties` file, specifying the name and password of a valid user with privileges to perform the migration in the `SRC_DB_USER` and `SRC_DB_PASSWORD` properties.

8.1.2 Connection rejected: FATAL: password

- When I try to perform a migration with Migration Toolkit, I get the following error:*

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

The user name or password specified in the `toolkit.properties` file is not valid to use to connect to the Postgres database.

To resolve this error, edit the `toolkit.properties` file, specifying the name and password of a valid user with privileges to perform the migration in the `TARGET_DB_USER` and `TARGET_DB_PASSWORD` properties.

8.1.3 Exception: ORA-28000: the account is locked

- When I try to perform a migration from an Oracle database with Migration Toolkit, I get the following error message:*

```
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.
```

To resolve this error, you can either unlock the user account on the Oracle server or edit the `toolkit.properties` file, specifying the name and password of a valid user with privileges to perform the migration in the `SRC_DB_USER` and `SRC_DB_PASSWORD` parameters.

8.1.4 Exception: oracle.jdbc.driver.OracleDriver

- When I try to perform a migration with Migration Toolkit, the migration fails and I get the error message:*

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

Before using Migration Toolkit, you must download and install the appropriate JDBC driver for the database that you are migrating from. See [Installing Source-Specific Drivers](#) for complete instructions.

8.1.5 I/O exception: The Network Adapter could not establish the connection

- When I try to perform a migration with Migration Toolkit, I get the following error:*

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

The JDBC URL for the source database specified in the `toolkit.properties` file contains invalid connection properties.

To resolve this error, edit the `toolkit.properties` file, specifying valid connection information for the source database in the `SRC_DB_URL` property. The `SRC_DB_URL` value is database specific. See [Building the toolkit-properties file](#) for detailed information for your source database type.

8.1.6 Exception: The URL specified for the “target” database is invalid

- When I try to perform a migration with Migration Toolkit, I get the following error:*

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

The JDBC URL for the target database (EDB Postgres Advanced Server) specified in the `toolkit.properties` file contains invalid connection properties.

To resolve this error, edit the `toolkit.properties` file, specifying valid connection information for the target database in the `TARGET_DB_URL` property. For information about forming a JDBC URL for EDB Postgres Advanced Server, see [Defining an EDB Postgres Advanced Server URL](#).

8.2 Migration errors

The following errors can occur after Migration Toolkit has successfully connected to the target and source database servers.

8.2.1 ERROR: Extra data after last expected column

- When migrating a table online, I get the error message:*

```
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"
```

This error occurs when the data in a column in `table_name` includes the delimiter character. To correct this error, change the delimiter character to a character not found in the table contents.

!!! Note * In this example, the pipe character (|) occurs in the text, `LOS|ANGELES`, intended for insertion into the last column, and the Migration Toolkit is run using the `-copyDelimiter '|'` option, which results in the error.

8.2.2 Error loading data into table: *TABLE_NAME*

- When performing a data-only migration, I get the following error:*

```
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
```

You must create a table to receive the data in the target database before you can migrate the data. Verify that a table with a name of `TABLE_NAME` exists in the target database. Create the table if necessary, and retry the data migration.

8.2.3 Error creating constraint CONS_NAME_FK

- When I perform a table migration that includes indexes and constraints, I get the following error message:*

```
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
```

The table you're migrating includes a foreign key constraint on a table that doesn't exist in the target database. Migration Toolkit creates the table, omitting the foreign key constraint.

You can avoid generating the error message by including the `-skipFKConst` option in the Migration Toolkit command.

8.2.4 Error Loading Data into Table

- I've already migrated the table definition; when I try to migrate the data into the table, I get an error:*

```
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
```

(continues on next page)

(continued from previous page)

```
[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.
```

The table definition in the target database doesn't match the migrated data. If you altered the target or source table, confirm that the table definition and the data are compatible.

ERROR: value too long for type

- I've already migrated the table definition. When I try to migrate the data into the table, I get the following error:*

```
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)
```

A column in the target database is not large enough to receive the migrated data. This problem can occur if the table definition is altered after migration. The column name (in this example, `location_id`) is identified in the line that begins with `Where:`.

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

To correct the problem, adjust the column size and retry the migration.

ERROR: Exception in thread:OutOfMemoryError

- When migrating from a MySQL database, I encounter the following error:*

```
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)
```

(continues on next page)

(continued from previous page)

```
at com.edb.DataLoader.run(DataLoader.java:186)
at java.lang.Thread.run(Thread.java:722)
```

By default, the MySQL JDBC driver fetches all of the rows in a table into Migration Toolkit in a single network round trip. This behavior can easily exceed available memory for large tables.

To correct the problem, specify `-fetchSize 1` as a command-line argument when you retry the migration.

ERROR CODES

When Migration Toolkit encounters a problem, it issues a notification in the form of an error code and a message.

Each error code begins with the prefix MTK- followed by five digits. The first two digits denote the error class, which is a general classification of the error. The last three digits represent the specific error condition.

The table lists the error classes.

Error Class	Description
00	Successful completion
01	Information
02	Warning
03	General drror
0x	Reserved
10	Invalid user input
11	Configuration error
12	Insufficient privileges
13	Unsupported feature
14	Missing object
15	Schema migration
16	Procedural language migration
17	Data loading

If there's an error reported back by a specific database server, this error message is prefixed with DB-. For example, if table creation fails due to an existing table in a Postgres database server, the error code 42P07 is returned by the database server. The specific error in the Migration Toolkit log appears as DB-42P07.

9.1 Error code summary

In the following tables, the column **Error code** lists the Migration Toolkit error codes. The **Message and resolution** column contains the message displayed with the error code. The message explains the cause of the error and how to resolve it.

In the **Message and resolution** column, \$NAME is a placeholder for information that is substituted at runtime with the appropriate value.

9.1.1 Class 02 - Warning

This class represents the warning messages.

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 Class 10 - Invalid user input

This class represents invalid user input values.

9.1.3 Class 11 - Configuration issues

This class represents invalid configuration settings in the `toolkit.properties` file or in any other configuration file used by the Migration Toolkit.

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 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-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: <i>\$REASON</i>

continues on next page

Table 3 – continued from previous page

Error code	Message and resolution

9.1.4 Class 12 - Insufficient privileges

This class represents insufficient privilege errors for loading database user/role information.

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 Class 13 - Unsupported features

This class represents errors related to the migration of unsupported objects and clauses. Either the target database has not implemented the given object, or the Migration Toolkit doesn't handle its migration.

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.
MTK-13002	For \$DATABASE views migration is not supported.
MTK-13003	For \$DATABASE 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 \$DATABASE_LINK. EnterpriseDB currently does not support this type of Database Link.

continues on next page

Table 5 – continued from previous page

Error code	Message and resolution
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> .
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 Class 14 - Missing objects

This class represents failures to find metadata information in the source database.

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.

continues on next page

Table 6 – continued from previous page

Error code	Message and resolution
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 Class 15 - Schema migration

The class represents migration issues related to nonprocedural database objects such as tables, constraints, indexes, synonyms, views, users, and roles.

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> .
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 Class 16 - Procedural language migration

The class represents migration issues related to database objects that are based on procedural languages such as procedures, functions, packages, anonymous blocks, triggers, and rules.

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> .
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 Class 17 - Data loading

This class represents errors that can occur while copying data from the source to the target database.

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> .

UNSUPPORTED FEATURES

EDB Postgres Advanced Server offers complete support for some Oracle features and partial support for others. Migration Toolkit can't migrate any object that uses an unsupported feature.

In some cases, Migration Toolkit can migrate objects that use features that offer partial compatibility. In other cases, EDB Postgres Advanced Server supports suitable workarounds.

Full-text search is an example of functionality that isn't fully compatible with Oracle. The EDB Postgres Advanced Server database has included support for full-text search for quite some time, but the implementation is different from Oracle's. Migration Toolkit can't migrate objects that use this feature.

EDB Postgres Advanced Server doesn't yet support other features. Features in this category include Automated Storage Management, table compression, and external tables. You can often implement a successful workaround:

- You can replace Automated Storage Management **with** system-specific volume management `↪ software`.
- You can implement table compression by storing data **in** a tablespace that resides on `↪ a compressed filesystem`.
- External tables don't exist in EDB Postgres Advanced Server, but you can load flat `↪ text files into staging tables in the database. We recommend using the EDB\ ↪ *Loader utility to load the data into an EDB Postgres Advanced Server database ↪ quickly`.
- When migrating multiple profiles **from an** Oracle database into EDB Postgres Advanced `↪ Server`, you must manually assign the profile to a user **or** users when the `↪ migration completes`.

10.1 Unsupported Postgres features

Migration Toolkit doesn't support migration of the following Postgres features:

- ``OPERATOR CLASS``
- ``OPERATOR FAMILY``

For information about `OPERATOR CLASS` and `OPERATOR FAMILY`, see the [PostgreSQL core documentation](#).

10.2 Frequently asked questions

Does Migration Toolkit support the migration of packages?

Migration Toolkit supports the migration of packages from an Oracle database into EDB Postgres Advanced Server. See [Functionality overview](#) for information about the migration support offered by EDB Postgres Advanced Server.

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

Yes. Data transfer is supported as part of an online or offline migration.

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

Migration Toolkit does support migration of tables containing data of the CLOB type.

Does EDB Postgres Advanced Server support the enum data type?

EDB Postgres Advanced Server doesn't currently support the enum data type but will support it in future releases. Until then, you can use a check constraint to restrict the data added to an EDB Postgres Advanced Server database. A check constraint defines a list of valid values that a column can take.

The following code sample includes a simple example of a check constraint that restricts the value of a column to one of three dept types; sales, admin or technical.

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

If you test the check constraint by entering a valid dept type, the INSERT statement works without error:

```
test=# INSERT INTO emp VALUES (7324, 'sales');  
  
INSERT 0 1
```

If you try to insert a value not included in the constraint (support), EDB Postgres Advanced Server returns an error:

```
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 doesn't support materialized views compatible with Oracle databases. To set up a materialized view/summary table in Postgres you must manually create the triggers that maintain the summary table. Automatic query rewrite isn't currently supported. You must make the application aware of the summary table's existence.

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?

Postgres doesn't have a problem storing TIME data types as long as the value of the hour component isn't greater than 24.

Unlike Postgres, the MySQL TIME data type allows you to store a value that represents either a TIME or an INTERVAL value. A value stored in a MySQL TIME column that represents an INTERVAL value can potentially be out of the accepted range of a valid Postgres TIMESTAMP value. If, during the migration process, Postgres encounters a value stored in a TIME data column that it perceives as out of range, it returns an error.

RELEASE NOTES

The Migration Toolkit documentation describes the latest version of Migration Toolkit 55 including minor releases and patches. The release notes in this section provide information on what was new in each release. For new functionality introduced in a minor or patch release, there are also indicators within the content about what release introduced the feature.

Version	Release Date
[55.2.0](07_mkt_552_rel_notes)	23 Dec 2021
[55.1.0](08_mkt_551_rel_notes)	26 Sep 2021
[55.0.0](09_mkt_55_rel_notes)	19 Mar 2020

11.1 Version 55.2.0

New features, enhancements, bug fixes, and other changes in Migration Toolkit 55.2.0 include:

Type	Description
Enhancement	Migration Toolkit is certified to support PostgreSQL 14.
Enhancement	Added support for EPAS 14, which includes some of the new Oracle compatibility features.
Enhancement	Partition tables with the LIST BY PARTITION clause followed by the AUTOMATIC keyword can now be migrated to target EPAS 13 and 14 databases.
Enhancement	Migration Toolkit now successfully migrates partition tables with the SUBPARTITION TEMPLATE clause to target EPAS 14 databases.
Bug Fix	Fixed the ORA-01795 issue to allow successful migration of more than a thousand users.
Bug Fix	Fixed the ORA-01795 issue to allow successful migration of more than a thousand partitions.

11.2 Version 55.1.0

New features, enhancements, bug fixes, and other changes in Migration Toolkit 55.1.0 include:

Type	Description
Enhancement	Migration Toolkit is enhanced to support migration from MySQL 5.7 and 8.0.
Bug Fix	For MySQL to PostgreSQL migration, the boolean data type was not migrating in LO mode. This issue is fixed.
Bug Fix	Fixed an issue to allow migration of table names starting with a number from MySQL to PostgreSQL.
Bug Fix	Fixed an issue where the TIMESTAMP value changes within the same time zone when migrating from MYSQL TO PostgreSQL.
Bug Fix	While migrating from MySQL to PostgreSQL, enum types were not being created on the target database with fastcopy mode. This issue is fixed.
Bug Fix	Fixed failures related to the migration of BIT(n) data type from MySQL to PostgreSQL when n > 1.
Bug Fix	Fixed an issue wherein while migrating from Oracle to PostgreSQL, a number format exception was reported (72863).
Bug Fix	Migration of the PostgreSQL table fails when the associated OID value exceeds the INTEGER range. This issue is fixed (72694).
Bug Fix	Fixed an issue related to the migration of JSON data type from PostgreSQL to Oracle.
Bug Fix	For Oracle 19c to EDB Postgres Advanced Server, certain system users/schemas were incorrectly picked for migration. This issue is fixed.
Bug Fix	Fixed an issue to allow migration of case-sensitive schema/table names in parallel loading mode from Oracle to EDB Postgres Advanced Server.
Bug Fix	Certain corner cases that were related to the use of the Custom Column Type mappings option, are fixed.

11.3 Version 55.0.0

New features, enhancements, bug fixes, and other changes in Migration Toolkit 55.0.0 include:

Type	Description
Enhancement	Support for parallel data loading at the table level. This allows you to use multiple threads to load data in parallel from the source database table and apply in parallel on the target database table. This significantly improves the data load time when dealing with relatively large size tables. The lab benchmark results have revealed a 25-60% reduction in the data load duration for a large table when using multiple parallel threads. This feature is supported for source PostgreSQL, EDB Postgres Advanced Server, and Oracle databases. See [Migration Options for Parallel Data Loading](http://localhost:8000/migration_toolkit/latest/08_mtk_command_options/#migration-options-for-parallel-data-loading).
Enhancement	As part of the table-level parallel data loading enhancement, the <i>tableLoaderLimit</i> and <i>parallelLoadRowLimit</i> options have been added to the command-line options list.
Enhancement	Support of schema and data migration from Oracle versions 18c and 19c.
Enhancement	Total migration time for the the schema and data migration session is reported via the <i>Total Elapsed Migration Time</i> attribute.
Bug Fix	Summary report does not accurately report status when data migration fails for table(s).
Bug Fix	Summary report does not remove redundant invalid objects.
Bug Fix	Data size for a table reported in exponential format.
Bug Fix	Avoid failures when a duplicate table name is specified via '-tables' list.
Bug Fix	Reflect small table sizes that were previously displaying as 0.0 MB.