EDB Postgres Language Pack

Version 4

Built at 2024-05-03T15:17:29

1 EDB Postgres Language Pack 3
2 Installing Language Packs 3
2.1 Installing Language Pack on Linux 3
2.2 Installing Language Pack on Windows 4
2.3 Installing Language Pack on MacOS 6
3 Uninstalling Language Pack 7
4 Using the procedural languages 8

1 EDB Postgres Language Pack

Learn how to install, configure, and use the procedural languages (PL/Perl, PL/Python, and PL/Tcl).

Language pack installers install the PL/Perl, PL/Python, and PL/Tcl procedural languages that can be used with EDB Postgres Advanced Server and PostgreSQL. The language pack installers allow you to install Perl, Tcl/TK, and Python without installing supporting software from third-party vendors.

The Language Pack 4.1 installer includes:

- Tcl with TK version 8.6.13
- Perl version 5.38.2
- Python version 3.11.7

Language Pack 4.1 contains the `cpan` package manager, and Python contains `pip` and `easy_install` package managers. There is no package manager for Tcl/TK.

In previous Postgres releases, `plpython` was statically linked with ActiveState's python library. The Language Pack Installer dynamically links with our shared object for python. In ActiveState Linux installers for Python, there is no dynamic library. As a result of these changes, `plpython` no longer works with ActiveState installers.

> **Note**
>
> The term *Postgres* refers to either PostgreSQL or EDB Postgres Advanced Server.

2 Installing Language Packs

Select a link to access the applicable installation information:

- Linux
- Windows
- MacOS

2.1 Installing Language Pack on Linux

Installing with EDB Postgres Advanced Server

Installing EDB Postgres Advanced Server on any Linux platform also installs language packs for Perl, Python, and TcL. See the installation documentation for more information on accessing the EDB repository and installing EDB Postgres Advanced Server.

Installing with PostgreSQL

When using PostgreSQL, language packs are optional. By default they are not required, but they are necessary for some build options. For more information on language pack requirements, see PostgreSQL requirements.

2.2 Installing Language Pack on Windows

On Windows, EDB provides a graphical interactive installer for Language Pack. Use Stack Builder (with PostgreSQL) or StackBuilder Plus (with EDB Postgres Advanced Server) to download the EDB installer package and then invoke the graphical installer for Language Pack. See Using Stack Builder or StackBuilder Plus.

Using Stack Builder or StackBuilder Plus

If you're using PostgreSQL, you can invoke the graphical installer for Language Pack with Stack Builder. SeeUsing Stack Builder.

If you're using EDB Postgres Advanced Server, you can invoke the graphical installer for Language Pack with StackBuilder Plus. SeeUsing StackBuilder Plus.

1. In Stack Builder or StackBuilder Plus, follow the prompts until you get to the module selection page.

   On the Welcome page, 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, select **Proxy servers** and specify a server. Select **Next**.

2. Expand the **Add-ons, tools, and utilities** node.

3. Select **EDB Language Pack** and select **Next**.

4. Browse to a directory where you want to install the language pack, or leave the directory set to the default location. Select**Next**.

5. When the installer states that all installation files have been downloaded, select **Next**.

6. Proceed to Using the graphical installer.

Using the graphical installer

1. Select the installation language and select **OK**.

2. On the Setup Language Pack page, select**Next**.

   The Ready to Install window displays the Language Pack installation directory.

   ```
   C:/edb/languagepack/<version>
   ```

   You cannot modify the installation directory.

3. Select **Next**.

   An information box shows the installation progress of the selected components.

4. When the installation is complete, select **Finish**.

Configuring Language Pack with EDB Postgres Advanced Server on Windows

When using EPAS on Windows, the Language Pack installer places the languages in:

```
C:\edb\as\languagepack\<version>
```

> **Note**
>
> Check that the version of Language Pack you are running matches the version in the file path and in the following commands.

After installing Language Pack, you must set the following variables:

```
set PYTHONHOME=C:\edb\as\languagepack\<version>\Python-3.7
```

where `<version>` is the current version of Language Pack.

Follow these steps to add Python, Perl, and Tcl to your Language Pack:

1. Go to **This PC > Properties > Advanced System Settings > Environment Variables**.
2. Edit the system-wide PATH variable.

   To add Python, enter the following:

   ```
   C:\edb\as\languagepack\<version>\Python-3.7\
   C:\edb\as\languagepack\<version>\Python-3.7\Scripts
   ```

   To add Perl and Tcl, enter the following:

   ```
   C:\edb\as\languagepack\<version>\Perl-5.26\bin
   C:\edb\as\languagepack\<version>\Tcl-8.6\bin
   ```

3. Close the dialog box.
4. Use the Windows `Services` applet to restart the EDB Postgres Advanced Server.

Configuring Language Pack with PostgreSQL on Windows

After installing Language Pack, you must set the following variables:

```
set PYTHONHOME=C:\edb\languagepack\v2\Python-3.7
```

Follow these steps to add Python, Perl, and Tcl to your Language Pack:

1. Go to **This PC > Properties > Advanced System Settings > Environment Variables**.
2. Edit the system-wide PATH variable.

   To add Python, enter the following:

```
C:\edb\languagepack\v2\Python-3.7\
C:\edb\languagepack\v2\Python-3.7\Scripts
```

To add Perl and Tcl, enter the following:

```
C:\edb\languagepack\v2\Perl-5.26\bin
C:\edb\languagepack\v2\Tcl-8.6\bin
```

3. Close the dialog box.

4. Restart the database server.

2.3 Installing Language Pack on MacOS

On MacOS, EDB provides a graphical interactive installer for Language Pack. To access the graphical installer, install Stack Builder as part of a PostgreSQL installation, and then use Stack Builder to invoke the graphical installer.

Using Stack Builder

Using PostgreSQL, you can invoke the graphical installer with Stack Builder. See Using Stack Builder.

1. In Stack Builder, follow the prompts until you get to the module selection page.

   On the Welcome page, 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, select **Proxy servers** and specify a server. Select **Next**.

2. Expand the **Add-ons, tools, and utilities** node.

3. Select **EDB Language Pack** and select **Next**.

4. Browse to a directory where you want to install the language pack, or leave the directory set to the default location. Select **Next**.

5. When the installer states that all installation files have been downloaded, select **Next**.

6. Proceed to Using the graphical installer.

Using the graphical installer

1. Select the installation language and select **OK**.

2. On the Setup Language Pack page, select **Next**.

   The Ready to Install window displays the Language Pack installation directory.

   ```
   /Library/languagepack/<version>
   ```

You cannot modify the installation directory.

3. Select **Next**.

   An information box shows the installation progress of the selected components.

4. When the installation is complete, select **Finish**.

Configuring on MacOS

To simplify setting the value of `PATH` or `LD_LIBRARY_PATH`, you can create environment variables that identify the installation location:

```
 PERLHOME=/Library/edb/languagepack/<version>/Perl-5.26
PYTHONHOME=/Library/edb/languagepack/<version>/Python-3.7
TCLHOME=/Library/edb/languagepack/<version>/Tcl-8.6
```

where `<version>` is the current version of Language Pack.

Then, execute the following command to instruct the Python interpreter where to find Python:

```
export PYTHONHOME
```

You can use the same environment variables when setting the value of `PATH`:

```
 export PATH=$PYTHONHOME/bin:
$PERLHOME/bin:
$TCLHOME/bin:$PATH
```

Lastly, set the following variables to instruct OSX where to find the shared libraries:

```
 export DYLD_LIBRARY_PATH=$PYTHONHOME/lib:
$PERLHOME/lib/CORE:$TCLHOME/lib:
$DYLD_LIBRARY_PATH
```

3 Uninstalling Language Pack

The Language Pack graphical installer creates an uninstaller that you can use to remove Language Pack. The uninstaller is created in the installation directory.

To uninstall Language Pack:

1. Navigate into the directory that contains the uninstaller and assume superuser privileges. Open the uninstaller and select**Yes** to begin uninstalling Language Pack.

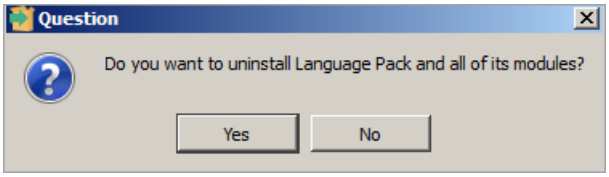

Fig. 1: The Language Pack Uninstaller

2. The uninstallation process begins. Select **OK** when the uninstallation completes.

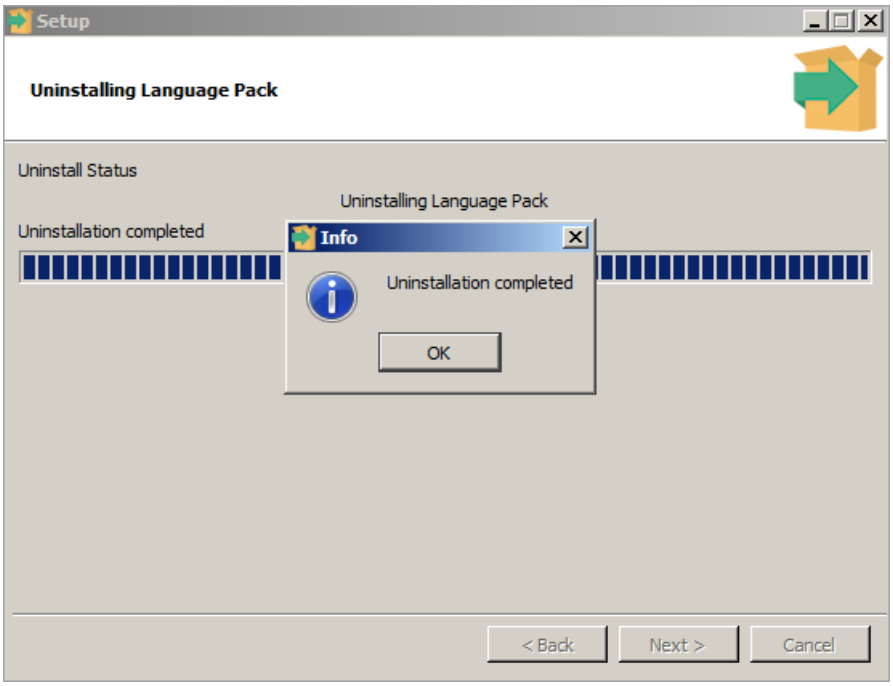

Fig. 2: Uninstalling Language Pack

4 Using the procedural languages

The Postgres procedural languages (PL/Perl, PL/Python, and PL/Tcl) are installed by the Language Pack installer. You can also use a native package to add procedural language functionality to your EDB Postgres Advanced Server installation.

PL/Perl

The PL/Perl procedural language allows you to use Perl functions in Postgres applications.

You must install PL/Perl in each database (or in a template database) before creating a PL/Perl function. Use the `CREATE LANGUAGE` command at the EDB-PSQL command line to install PL/Perl. Open the EDB-PSQL client, establish a connection to the database in which you wish to install PL/Perl, and enter the command:

```
CREATE EXTENSION plperl;
```

You can now use a Postgres client application to access the features of the PL/Perl language. The following PL/Perl example creates a function named `perl_max` that returns the larger of two integer values:

```
CREATE OR REPLACE FUNCTION perl_max (integer, integer) RETURNS integer
AS
$$
if ($_[0] >
$_[1])
{ return $_[0];
}
return
$_[1];
$$ LANGUAGE
plperl;
```

Pass two values when calling the function:

```
SELECT perl_max(1, 2);
```

The server returns:

```
 __OUTPUT__
perl_max
---------
       2
(1 row)
```

For more information about using the Perl procedural language, consult the official PostgreSQL documentation.

PL/Python

The PL/Python procedural language allows you to create and execute functions written in Python within Postgres applications. The version of PL/Python used by EDB Postgres Advanced Server and PostgreSQL is untrusted (`plpython3u`); it offers no restrictions on users to prevent potential security risks.

Install PL/Python in each database (or in a template database) before creating a PL/Python function. You can use the `CREATE LANGUAGE` command at the EDB-PSQL command line to install PL/Python. Use EDB-PSQL to connect to the database in which you wish to install PL/Python, and enter the command:

```
CREATE EXTENSION plpython3u;
```

After installing PL/Python in your database, you can use the features of the PL/Python language.

> **Note**
>
> The indentation shown in the following example must be included as you enter the sample function in EDB-PSQL.

The following PL/Python example creates a function named `pymax` that returns the larger of two integer values:

```
CREATE OR REPLACE FUNCTION pymax (a integer, b integer) RETURNS integer
AS
$$
if a >
b:
return
a
return
b
```

```
$$ LANGUAGE
plpython3u;
```

When calling the `pymax` function, pass two values as shown below:

```
SELECT pymax(12,
3);
```

The server returns:

```
__OUTPUT__
pymax
-------
    12
(1 row)
```

For more information about using the Python procedural language, consult the official PostgreSQL documentation.

PL/Tcl

The PL/Tcl procedural language allows you to use Tcl/Tk functions in applications.

You must install PL/Tcl in each database (or in a template database) before creating a PL/Tcl function. Use the `CREATE LANGUAGE` command at the EDB-PSQL command line to install PL/Tcl. Use the psql client to connect to the database in which you wish to install PL/Tcl, and enter the command:

```
CREATE EXTENSION pltcl;
```

After creating the `pltcl` language, you can use the features of the PL/Tcl language from within your Postgres server.

The following PL/Tcl example creates a function named `tcl_max` that returns the larger of two integer values:

```
CREATE OR REPLACE FUNCTION tcl_max(integer, integer) RETURNS integer AS
$$
if {[argisnull 1]}
{
if {[argisnull 2]} { return_null
}
return
$2
}
if {[argisnull 2]} { return $1
}
if {$1 > $2} {return
$1}
return
$2
$$ LANGUAGE
pltcl;
```

Pass two values when calling the function:

```
SELECT tcl_max(1,
2);
```

The server returns:

```
 __OUTPUT__
tcl_max
--------
       2
(1 row)
```

For more information about using the Tcl procedural language, consult the official PostgreSQL documentation.