Accessing Databases using the Python DBAPI-2.0

Federico Di Gregorio
fog@debian.org

version 0.1

Contents

Foreword

This document is a short but extensive introduction to the (easy) task of accessing relational databases using the Python programming language and any database adapter module using the DBAPI-2.0 API.

About half of the questions on the psycopg[1] mailing list are about correctly using the API and not about problems with psycopg itself. Thinking that users of other modules can have the same problems, I wrote this short document, hoping to do something usefull to the community.

While quite simple, I found the DBAPI-2.0 a powerfull tool for accessing datatbases from the Python language. I hope this short introduction will help you starting faster and writing better (free) software.

Have you got everything?

Before starting experimenting with Python and the DBAPI-2.0, you will need to install your favourite database, the Python interpreter and libraries and the database adapter module of your choice. I can't help you in doing that: just read carefully the documentation and make sure everything is setup correctly before writing to the adapter's author that "your module just doesn't work".

About the examples

All the code used in this document is available in source form in the `examples/` directory and is distributed under the GNU General Public License. For more information about the GNU GPL refer to the COPYING file that comes with the example code. Moreover, all the examples use the psycopg adapter, but should work by just importing the right module and changing every occurance of the word 'psycopg' to the name of your favourite database adapter. If you don't understand what I am saying here, just make sure to carefully read chapter 1 and especially section 1.1.

A prerequisite for executing the provided examples is to create a test database and maybe a user to connect to it. All the examples use the name `test` for both the database and the user (and a null password: this is not a good habit, but simplifies the examples a little bit.) Your best option is to create the `test` database from scratch, and simply destory it at the end of this book. If you don't know how to create a new database, look at the documentation that comes with it, or ask a friend or the system administrator to do it for you.

[1] psycopg is just one of the many database adapter available for the Python language, but it is the author's favourite one.

For the module implementor

Writing a Python database adapter is never an easy task. Fixing bugs and making it a *good* and *stable* adapter is even harder. And what about lots of people sending email and reporting 'bugs' that are only programming error or misinterpretations of the API used by the vast majority of the Python database modules?[2] But it is *our* fault, not users'. Documentation is a must, especially when most of the users are green programmers and the API definition was written by more seasoned ones.

This document, while written mostly with the psycopg module in mind, is a generic introduction of Python programming using the DBAPI-2.0 to access relational databases. It is distributed under the Free Documentation License, to allow for modifications and inclusion in all the other modules package's, hoping it to be usefull to other users than just the psycopg ones. If you decide to distribute this document with your module, I suggest you to:

1. change every reference to 'psycopg' in the text to your module name, to make the reader more confortable while passing from this document to the editor;
2. modify (or rewrite, if necessary) chapter 1, especially the *Importing the module* and *Opening the connection* sections to match your module peculiarities;
3. add a section about the `connect()` function as implemented in your module to chapter 2;
4. add a new appendix describing your module's extensions to the DBAPI-2.0 or other particular features and quirks.

For the example code your best option is to modify it to work with your adapter (usually that means just modifying the `connect()` function.) You can also modify the example code showed in this book to match your modifications to the example files. If you included a completely new appendinx about your module, it is a very good idea to put some example code in a subdirectory of the `examples/` directory: just name it after your module. If you do all that, please, send the changes back to me: by doing so, you will add a great value to both versions of this manual.

If you make any improvments to (or find errors in) this document, please send me a patch (removing any module-specific stuff, if you did such modifications.)

Thank you very much,

Federico Di Gregorio
`fog@debian.org`

[2] The Python DBAPI, version 2.0 documented in the document.

Chapter 1

Introducing the DBAPI-2.0

This chapter will take you on a fast tour of the main features of the DBAPI-2.0, showing how to:

- open a connection to the database;
- create a cursor; and
- execute a query.

In the following chapters we'll analyze in-depth every part of the DBAPI-2.0, starting from the basic, like opening a connection to the database and ending with some exotic fruits like cursor isolation and exceptions. Let's start the tour.

1.1 Importing the module

Importing the module is, by definiton, module dependent. With psycopg you only need to add an `import` line before connecting to the database, as follows:

```
import psycopg
```

Other modules may require multiple `import`s or slightly more complex actions. Another import style, used in the examples, explicitly pull some functions (or objects) off the module, as in:

```
from psycopg import connect
```

1.2 Opening the connection

Before a database can be accessed, we need to open a connection and store the *connection object* in a variable. All the programming examples through this text (and in the `examples/` directory) will use `conn` as the connection's name.

A connection is opened calling the `connect()` function, giving it a *Data Source Name* (DSN from now on) and some other optional arguments. The DSN and optional arguments are module-dependent and the better thing you can do is to consult your favourite module documentation. psycopg uses a DSN built from keywords (`dbname`, `host`, `port`, `user`, `password`) and values, as in the following example:

```
conn = psycopg.connect("dbname=test user=test")
```

If `host` is not specified, it defaults to the local host, using UNIX sockets (unless `port` is specified.) If either `port` or `host` are given, the connection will be over TCP/IP.

1.3 Executing queries on the database

After connecting to the database, we're ready to create a *cursor* and use it to send queries to the database. We'll talk more about cursors in the *Cursors* section, for now it suffices to know how to create one (note that in all the examples the first cursor created is named `curs`, when more than one cursor is needed they are called `curs1`, `curs2`, etc.)

```
curs = conn.cursor()
```

Quite simple, isn't it? Now we can use the cursor to send *any* SQL query to the database. This is done using the `execute()` function, for example

```
curs.execute("SELECT * FROM atable")
```

will select every column of every row from `atable`. But... we still don't have the `atable` table! If you read carefully what i wrote before this example, you can find the answer by yourself. I wrote that we can use the cursor to send **any** SQL query to the database, even a `"CREATE TABLE"`.

The next example is a complete program, that creates the table that will be used for almost all the example code in this text. Just substitute your DSN and run it to create a table and fill it with a couple of rows.

create_table.py

```
from psycopg import connect

conn = connect("dbname=test user=test")
curs = conn.cursor()

curs.execute("""CREATE TABLE atable (
                  name  char(8),
                  value float)""")
curs.execute("INSERT INTO test VALUES ('one', 1.0)")
curs.execute("INSERT INTO test VALUES ('two', 2.0)")

conn.commit()
```

This example, apart from being the first complete program we encounter, also introduces a new cursor method: `commit()`, used to commit, i.e., make permanent, the changes you did to the database.

Note that if you run the `create_table.py` example two times in a row, you will get an error; something like:

```
Traceback (innermost last):
  File "create_table.py", line 2, in __main__
psycopg.ProgrammingError: ERROR:  Relation 'atable' already exists
```

The exact error you get is database- and module-dependant, but we can anticipate what went wrong: the second run tried to create an already-existing table. A DBAPI-2.0 compliant module will raise an exception for every database error (and for some module errors too.) For now we will just ignore such exceptions: to know more about them jump at the *Advanced topics* chapter, near the end of this book.

And to finish this fast tour of the DBAPI-2.0, a very short program that read back (and prints) the two rows inserted by `create_table.py`.

read_table.py

```
from psycopg import connect

conn = connect("dbname=test user=test")
curs = conn.cursor()

curs.execute("SELECT * FROM atable")

rows = curs.fetchall()

for i in range(len(rows)):
    print "Row", i, "name", rows[i][0], "value", rows[i][1]
```

`.fetchall()` is just one of the cursor methods that can be used to retrieve data from the database after a `SELECT` statement; for now we just note that the result is a list of tuples with each tuple big enough to hold an entire row of data.

Remember that you can always experiment by running python in interactive mode: as a final example, let's remove the table we created while running `create_table.py` (`$` is the shell prompt, `>>>` python's one):

```
$ python
Python 2.1.2 (#1, Jan 18 2002, 18:05:45)
[GCC 2.95.4  (Debian prerelease)] on linux2
Type "copyright", "credits" or "license" for more information.
>>> import psycopg
>>> conn = psycopg.connect("dbname=test user=test")
>>> curs = conn.cursor("DROP TABLE atable")
>>> conn.commit()
>>> ^D
$
```

Now, let's move to the next chapter and to a more sistematic approach.

Chapter 2

Managing connections

In the DBAPI-2.0 paradigm, connections to the database are represented by *connection objects*, created by calling the `connect()` function. It is possible to open any number of connections to different databases and, obviously, multiple connections to the same database. A connection object is an abstraction and if the database supports transactions, each connection is guaranteed to be independent (from the transactional point of view) from other connections to the same database. Also note that a connection object does not map directly to a *phisical connection*; even if most modules implement a one-to-one correspondance, some use pools of multiple physical connections to better support concurrent execution and threading.

2.1 The `connect()` function

The prototype for the `connect()` function call is given below. `dsn` is the *Data Source Name*, a module-dependent argument. `connect()` accept a varying number of additional arguents, all of them *very* module-dependent. Fortunately, `connect()` is the only function to present such variability and about 99% of the DBAPI-2.0 is module-independent (that's what the DBAPI-2.0 document was written for, to begin.) Just note that the following sections can or cannot apply to your database driver.

Here is the `connect()` prototype from the original DBAPI-2.0 document: all parameters are optional but the form presented here is the suggested one:

```
connect(dsn, user, password, host, database, ...)
```

The rest of this chapter is dedicated to expose the different syntaxes of the `connect()` functions used by the most common modules.

2.1.1 psycopg

Here is the `connect()` function, as defined by the psycopg module:

```
connect(dsn, maxconn, minconn, serialize)
```

where `user`, `password`, `host` and `database` are missing because they are part of the `dsn` string. The `dsn` string is built as a list of `key=value` pairs, separated by white space. All the available keys are described in the following table:

Key	Description
`dbname`	this is the database name, as given in the PostgreSQL `CREATE DATABASE` statement
`user`	the user name used to connect to the database
`password`	the plain password (or some other kind of authorization token) used to gain access to the database
`host`	the host name of the machine where the database resides; if missing the connection will be to the local host using UNIX sockets if possible
`port`	the port to connect to on the remote machine; if missing defaults to 5432 (PostgreSQL default port) if host is specified, else the connection goesover UNIX sockets

Here are some examples of `dsn` strings that can be used in psycopg's `connect()` calls:

```
"dbname=test"
```

connect to the database `test` on the local machine over UNIX sockets; user is the same as the currently logged-in user and password is empty. Next one is identical but connect over TCP/IP using the default port:

```
"dbname=test host=localhost"
```

Then let's add a user (`fog`) and a password (`dust`, the name of my ferret, never do that in a production environment!) and connect to a different machine:

```
"dbname=test host=db.initd.org user=fog password=dust"
```

The psycopg adapter can use a pool of connections to speed-up connection times when using multiple *cursors* in multithreaded applications. This is called *putting the connection in non-serialized mode* and is accomplished using the `serialize` parameter. Non-serialized connections (and the `serialize`, `maxconn` and `minconn` parameters) are discussed in detail in appendix **??**, page **??**.

2.2 Transactions

The DBAPI-2.0 document specifies that transactions should be worked out at connection level (when the underlying database provides transactions; if this is not the case all the transaction-related functions are no-ops.)

If the database supports transactions, all the "work" done via a connection (i.e., all the changes done to the database) is not "saved" until you commit it to the database. You can do that by calling the `.commit()` method on the connection object (**note**: on the connection object, not on the cursor object!)

If your program encounters an error or ends up in an unusual situation (like an exception) and you don't want to commit your changes to tha database but start from scratch with the same connection, you can undo all the changes (from the last commit on) using the `.rollback()` method. Just remember that a relational database does not have unlimited undo levels, only *one*. Calling `.rollback()` again and again or calling it after a `.commit()` simply will not work.

2.3 Closing the connection

A connection can be explicitly closed by invoking the `.close()` method on the connection object, by explicitly destroying the connection object using the `del` operator or

by letting the Python garbage collector do its work. An important point is that if the module supports the transaction model, closing a connection does an implicit rollback on it. That simply means that you'll need to call `.commit()` one last time before letting the connection die.

2.4 Putting it all togheter

Here is a little example that just show how to open a connection getting the `dsn` string from the command line arguments. It surely needs to be modified to work with your driver of choice: twisting it until you get the '`Connected to the DB!`' message is a very good exercise to understand how the `connect()` function works for you. Try it out with something similar to

```
python open_connection.py dbname=test fog dust
```

or, supposing your database is located on a remote host:

```
python open_connection.py 'dbname=test host=db.initd.org' fog dust
```

The example contains some extra code, to manage exceptions (you're supposed to fail your first connection, after all): don't bother too much about it, we'll talk extensively about exceptions raised by the database drivers in a later chapter.

open_connection.py

```
import sys
from psycopg import connect

## just check we have enough arguments
if len(sys.argv) != 4:
    print "Usage: %s dsn user password" % sys.argv[0]
    sys.exit(1)

## if you're not using psycopg your connect call may
## differ (e.g., connect(dsn, user, password))
dsn += " user=%s password=%s" % (user, password)
try:
    conn = connect(dsn)
except StandardError, err:
    print "Something went bad during the connection:\n", err
    sys.exit(2)

## lets close it and print a congratulation message
conn.close()
print "Connected to the DB! Kudos..."
```

Chapter 3

Executing Queries

Just as you need a *connection oject* to manage your database connections, you will use *cursor objects* to manage the execution of queries and to retrieve data. Some databases (like PostgreSQL, Oracle and others) support cursors natively and have exotic functionality[1] accessible only when using them. DBAPI-2.0 cursor objects, just like connection objects, share *nothing* with (real) database cursors, they are completely different stuff.

The DBAPI-2.0 defines a set of well-defined cursor methods and attributes and, even if almost every DBAPI-2.0 compliant module extends the cursor interface somewhat, the basics work the same for *every* module—especially if compared with the variance of the `connect()` call. But take care; the SQL code you send to the database *can* contain non-standard features or features not supported by every database backend. The DBAPI-2.0 is there to help you, but cannot guarantee your code will run unmodified on every database.

A cursor object is always associated with an open connection and is created by calling the `.cursor()` method on the connection. As per the DBAPI-2.0, the `.cursor()` method takes no arguments and returns a new cursor object, as in the following example:

```
curs = conn.cursor()
```

A new cursor is valid and can be used to send queries to the backend until the cursor itself or its parent connection are closed; after that trying to use the cursor shall raise an exception.

3.1 The `.execute()` method

`.execute()` is surely the most important cursor method and the only way to send SQL queries to the database. It takes one mandatory argument (the query string) and an optional list (or dictionary) of *bound variables*:

```
.execute(query, vars=None)
```

`.execute()` makes really easy to have the database execute literal SQL code, as in the following examples:

Until now we simply built the query string directly but there is a much better way to

[1] Like binary access to data or row scrolling just to name two.

3.2 Retrieving data

3.3 Other cursor attributes and methods

Chapter 4

Advanced topics

4.1 Cursor isolation

4.2 Threading

Appendix A

GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

A.1 Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains noth-

ing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A.2 Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

A.3 Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover,

and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

A.4 Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

- List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).

- State on the Title page the name of the publisher of the Modified Version, as the publisher.

- Preserve all the copyright notices of the Document.

- Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

- Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

- Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- Include an unaltered copy of this License.
- Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

A.5 Combining Documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

A.6 Collections of Documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

A.7 Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

A.8 Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include

translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

A.9 Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.10 Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections" instead of saying which ones are invariant. If you have no Front-Cover Texts, write "no Front-Cover Texts" instead of "Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.