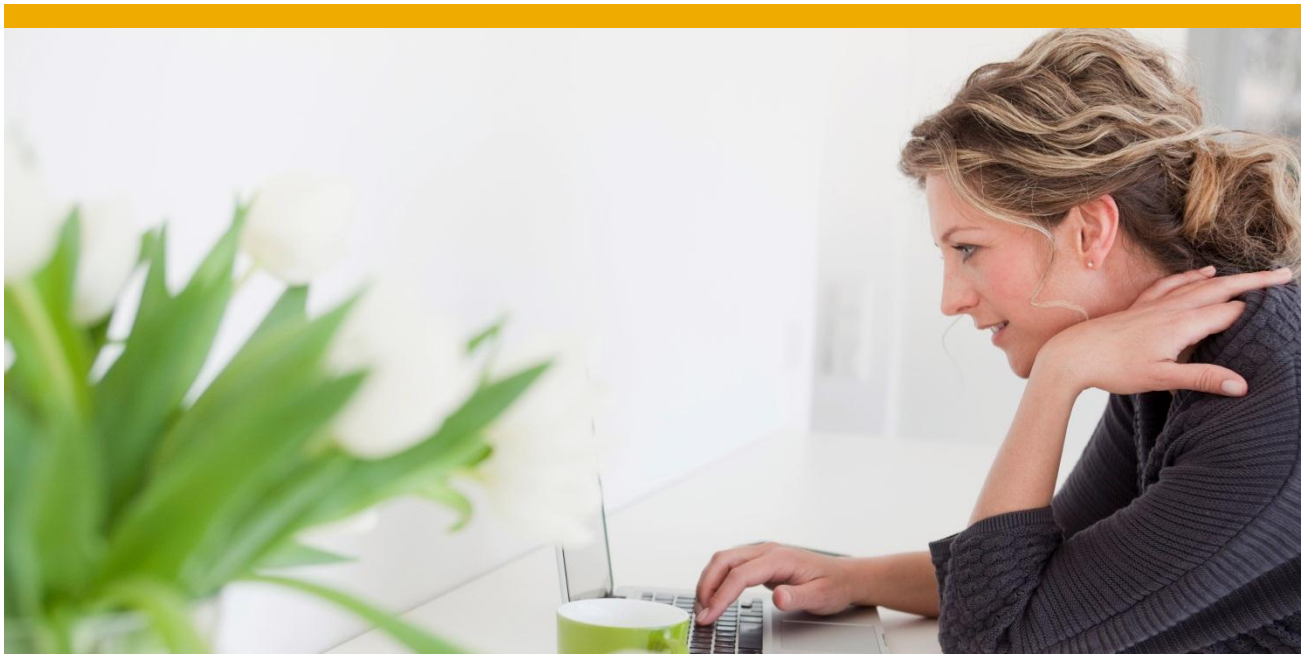


Developing Python application with Sybase ASE

Getting Started Guide for Sybase ASE and Python

TABLE OF CONTENTS

INTRODUCTION

ASE Developer Edition as well as other editions come bundled with a client software development called Open Client Software Development kit or OCS in short) which contains database drivers and extension module for different programming languages

Sybase ASE supports Python by providing extension module- called sybpydb- which lets Python connect to Sybase ASE database, Perform queries and retrieve results from database.

This guide will teach how to setup Python environment to start developing applications which use Sybase ASE

ASE CONFIGURATION STEPS

For simplicity, this tutorial assumes that Windows 7 is used for application development, and same machine is used for database installation as well as client application.

Here is typical install file structure for Sybase ASE Developer Edition on Windows 7 box.

C:\Sybase\
C:\sybase\ase-15_0 corresponds to actual ASE database installation
C:\Sybase\ocs-15_0 corresponds to bundled client software (called OCS in Sybase ASE parlance) development kit. It is interesting to note that that updates to OCS can be downloaded from software.sybase.com (or other release mechanism if support contract purchased) independent of updates to Database.

Set following environment variables

- %SYBASE% to c:\syabse
- %SYBASE_OCS% to %SYBASE%\ocs-15_0

PYTHON REQUIREMENTS

Sybase ASE extension module **sybpydb** currently supports 2.6, 2.7 and 3.1. Make sure that 64 bit version of language installed as Sybase supports only 64 bit platform for all OS. These python version are availably publicly from internet.

Extension module for 2.6, 2.7, or 3.1 will be located in corresponding %SYBASE_OCS%\python directory

Check version of Open Client SDK by %SYBASE_OCS%\ bin\isql –v. It should be 15.7 or more.

Platform	Default Installation Path	Python Version
Windows	• `%SYBASE%\%SYBASE_OCS%\python\python26_64\dll`	2.6
Windows	• `SYBASE%\%SYBASE_OCS%\python\python27_64\dll`	2.7
Windows	• `SYBASE%\%SYBASE_OCS%\python\python31_64\dll`	3.1
All Other Platforms	• `$SYBASE/$SYBASE_OCS/python/python26_64r/lib`	2.6,2.7
All Other Platforms	`$SYBASE/$SYBASE_OCS/python/python31_64r/lib`	3.1

To use the Adaptive Server Enterprise extension module for Python in an application, there are two ways.

- Set PYTHONPATH,
    - In console window, set environment variable PYTHONPATH to Sybase ASE's Python Extension dll.
        - For example, if your development environment is using Python 3.1 64 bit, PYTHONPATH would be set to %SYBASE_OCS%\python\31_64\dll
        - set PYTHONPATH=c:\Sybase\OCS-15_0\python\python6_64\dll
        - A simple Python program to verify if sybpydb is in path

          ```
          import sys

          for d in sys.path:
              print(d);
          ```

- Set Python variable sys.path to one of the following directory paths in table above inside your Python code.

Using Extension Module for Python

Connecting to ASE Server

Use the import statement to load the extension module **sybpydb** for Python by including this line at the top of the python script.

```
import sybpydb
```

sybpydb connect method can take username, password and optional servername as parameter to connect to Sybase ASE database. If servername is not specified, then it is read from DSQUERY environment variable atabase name can be specified either in

```
# Create a connection
```

- `conn = sybpydb.connect(user='john', password='sybase')`
- `conn = sybpydb.connect(user='john', password='sybase', servername='localhost')`

Note: If neither DSQUERY environment variable is set, nor servername is passed as parameter, then default "SYBASE" is selected.

Detail Sybase Python API reference is available here

http://infocenter.sybase.com/help/topic/com.sybase.infocenter.dc01692.1570/doc/html/car1309464822398.html

Sample Python Program to query and display rows

- Open connection by calling sybpydb.connect method, save connection object
- After a connection is established, get cursor from connection object, to manage the context of a fetch operation
- Execute native Sybase ASE SQL or call stored procedure using cursor object
- Fetch result row by calling cursor's fetchall method
- Perform operations on result row
- Close cursor, and connection object

It is possible to pass input and output parameters to stored procedure as well as perform aggregate computations on result rows retrieved.

Here is complete end to end example:

```
import sybpydb


   #Create a connection.
   conn = sybpydb.connect(user='sa',password='abcdefgh')


   # Create a cursor object.
   cur = conn.cursor()


   cur.execute("drop table footab")
   cur.execute("create table footab ( id integer, first char(20) null, last char(50)
null)")
   cur.execute("insert into footab values( ?, ?, ? )", (1, "John", "Doe"))
   cur.execute("select * from footab")
   rows = cur.fetchall()
   for row in rows:
       print "-" * 55
       for col in range (len(row)):
          print "%s" % (row[col]),


   #Close the cursor object
   cur.close()


   #Close the connection
   conn.close()
```

Sample Program to insert, update rows

RUN SAMPLE PROGRAMS

Sybase ASE comes bundled with Python sample program based on sample database “pubs3”- both sample program and database is included with Developer Edition.

- Start Sybase ASE server by going to Control Panel→ Administrative Services-→ Start Sybase ASE Service
- Set %SYBASE%, %SYBASE_OCS%, %PYTHONPATH% environment variable as mentioned in configuration step above
- cd %SYBASE_OCS%\ sample\python
- Test environment is correctly setup and ASE server is up and running by executing “python test.py” first.

```
import sybpydb

conn = sybpydb.connect(user='sa', password='')
print("Successfully established connection, exiting.")
conn.close()
```

- Other programs are self-explanatory.

Output from starting Python Program

cd %SYBASE_OCS%\sample\python

```
c:\Sybase5\OCS-15_0\sample\python>c:\Python31\python.exe firstapp.py
White: Menlo Park
Green: Oakland
Carson: Berkeley
O'Leary: San Jose
Straight: Oakland
Bennet: Berkeley
Dull: Palo Alto
Gringlesby: Covelo
Locksley: San Francisco
Yokomoto: Walnut Creek
Stringer: Oakland
MacFeather: Oakland
Karsen: Oakland
Hunter: Palo Alto
McBadden: Vacaville
```

SYBASE SUPPORT FOR PYTHON ODBC DRIVER

PyODBC is not officially developed by Sybase ASE but it is possible to access Sybase ASE database through Sybase ODBC driver and PyODBC combination.

pyodbc is a Python 2.x and 3.x module that allows you to use ODBC to connect to almost any database from Windows, Linux, OS/X, and more.

pyodbc is licensed using an MIT license, so it is free for commercial and personal use

PyODBC can be downloaded from here http://code.google.com/p/pyodbc/ . PyODBC installation requires right Python version to be installed already before installation of PyODBC can begin.

Platform specific download http://code.google.com/p/pyodbc/downloads/list . So far, Python 2.6 and Python 2.7 are supported.

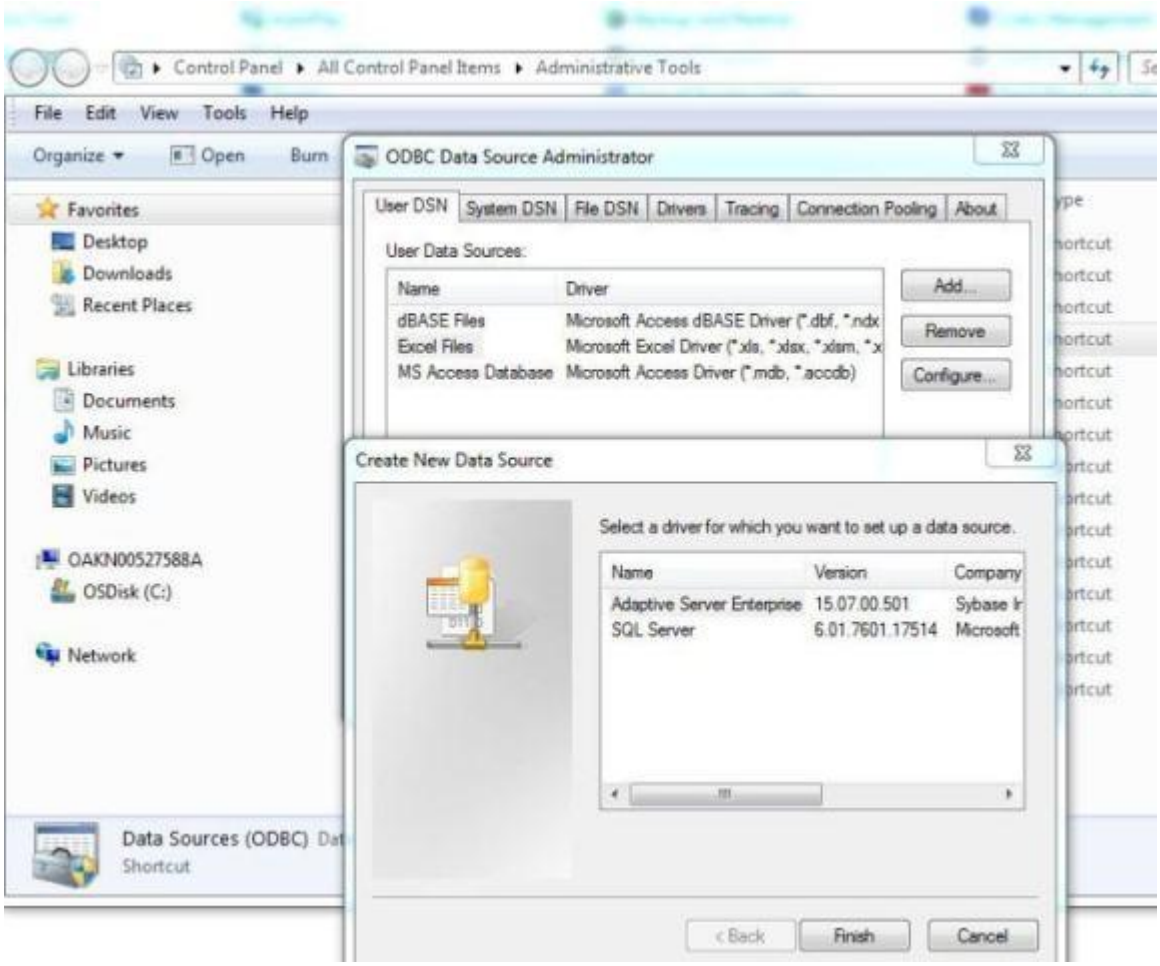
Connection String for PyODBC

The most important thing to know is that pyodbc lets any generic ODBC connection string documentation should be valid. And anything ODBC supports, pyodbc supports, including DSN-less connections and FILEDSN

Here is sample DSN for Sybase ODBC Driver on Windows 7.

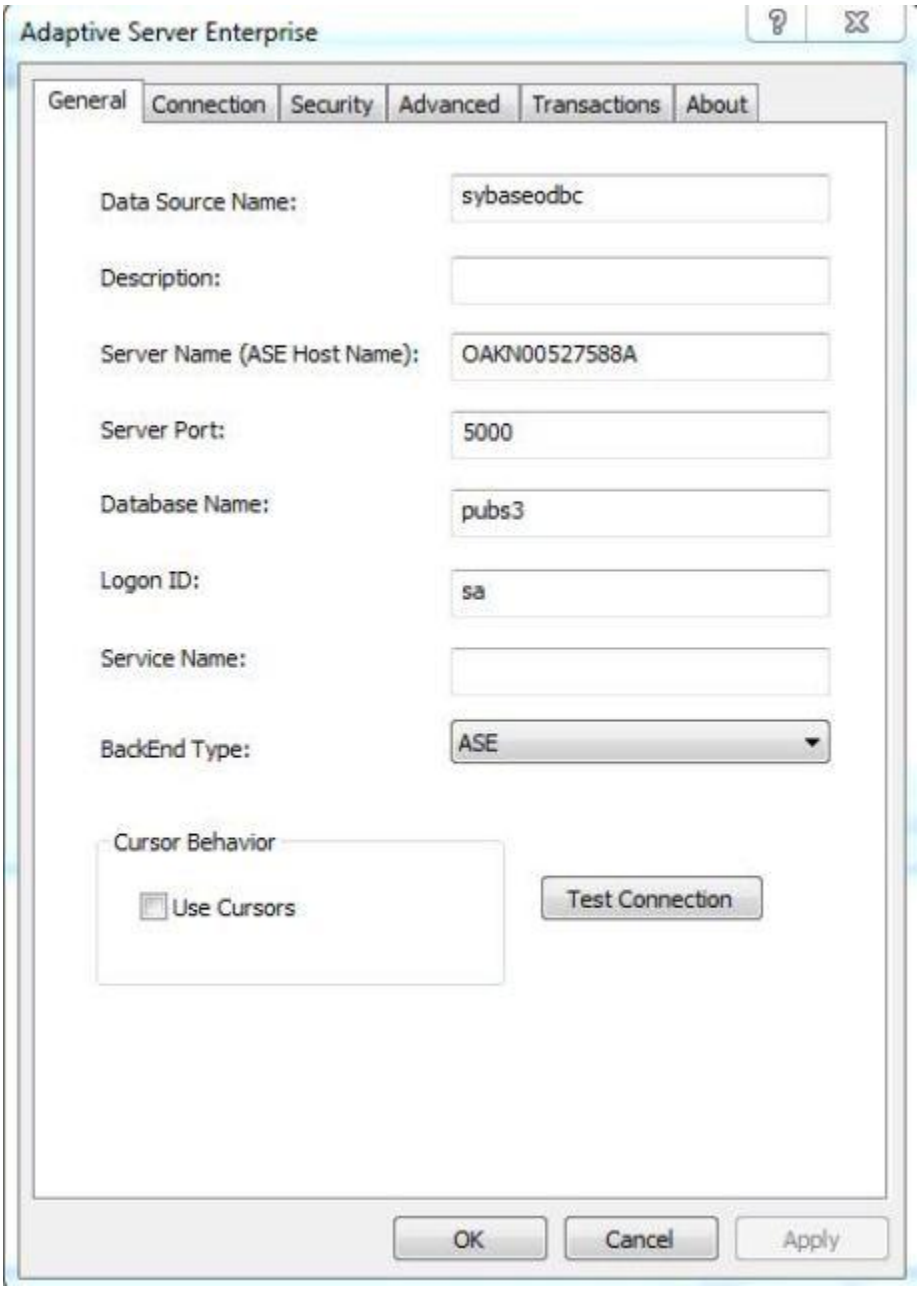
Control Panel -→ Administrative Tools -→DataSources (ODBC)

Step 1.

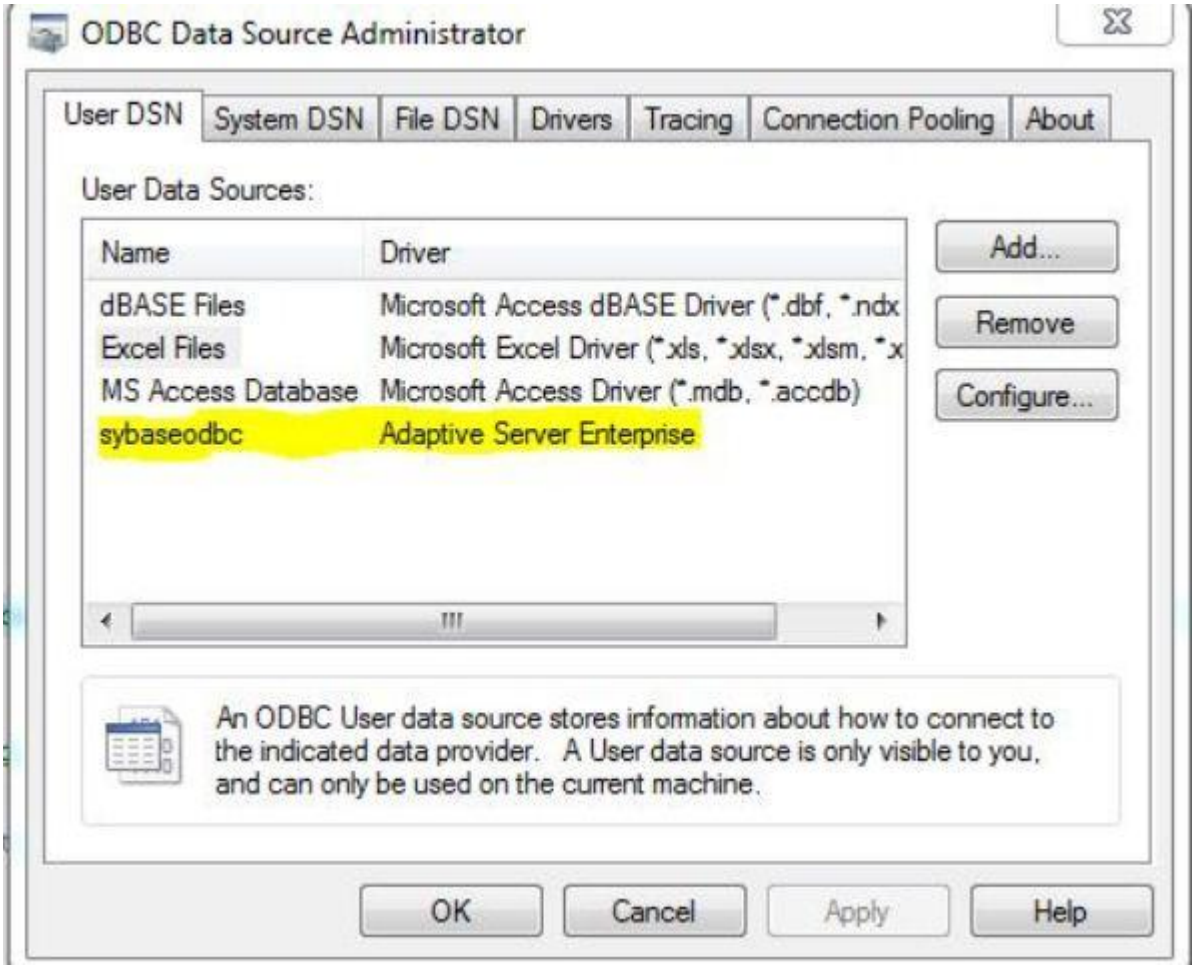


Step 2.

Enter Servername, port , database etc. Password is not stored in DSN , required just for testing connection to database.


Adaptive Server Enterprise
General
Connection
Security
Advanced
Transactions
About
Data Source Name:
sybaseodbc
Description:
Server Name (ASE Host Name):
OAKN00527588A
Server Port:
5000
Database Name:
pubs3
Logon ID:
sa
Service Name:
BackEnd Type:
ASE
Cursor Behavior
Use Cursors
Test Connection
OK
Cancel
Apply

Step 3.


ODBC Data Source Administrator
User DSN
System DSN
File DSN
Drivers
Tracing
Connection Pooling
About
User Data Sources:
Name
Driver
dBASE Files
Microsoft Access dBASE Driver (*.dbf, *.ndx
Excel Files
Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.x
MS Access Database
Microsoft Access Driver (*.mdb, *.accdb)
sybaseodbc
Adaptive Server Enterprise
Add...
Remove
Configure...
An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.
OK
Cancel
Apply
Help

Using PyODBC
After installing PyODBC module , import pyodbc module in your Python script , connect to Sybase ASE database either through 1) DSN or 2) connection string.

For this example, we will be using DSN “sybaseodbc” just created.

Here is sample program using pyodbc to make connection to sample “pub3” database and display rows from table.

```
import pyodbc

dsn="DSN=sybaseodbc;PWD=admin123"
cnnctn = pyodbc.connect(dsn)
print("Successfully established connection, .")
#To ensure updating (other choises - updating is not sure)
cnnctn.autocommit = True
cur = cnnctn.cursor()
cur.execute("select au_lname, city from pubs2..authors where state = 'CA'")
while True:
    row = cur.fetchone()
    if (not row):
        break
    print("%s: %s" % (row[0], row[1]))

cur.close()
cnnctn.close()
```

Output from program.

```
c:\Sybase5\OCS-15_0\sample\python>c:\Python27\python.exe test1.py
Successfully established connection, .
White: Menlo Park
Green: Oakland
Carson: Berkeley
O'Leary: San Jose
Straight: Oakland
Bennet: Berkeley
Dull: Palo Alto
Gringlesby: Covelo
Locksley: San Francisco
Yokomoto: Walnut Creek
Stringer: Oakland
MacFeather: Oakland
Karsen: Oakland
Hunter: Palo Alto
McBadden: Vacaville
```

Row Modification
Insert

It will be something similar to:

```
cursor.execute("insert into products(id, name) values ('pyodbc', 'awesome library')")
cnxn.commit()
```

Update and Delete

```
cursor.execute("insert into products(id, name) values (?, ?)", 'pyodbc', 'awesome library')
cnxn.commit()
```