Open QM

Using OpenQM with Python

Overview

The history of the Python language goes back about 30 years but it has become more widely used since the release of version 3. Python shares many concepts with those familiar to QMBasic developers. It supports both the structured programming and object oriented programming paradigms, loads modules dynamically at run time and uses type variant variables.

Developers now have the ability to access the QM database from a Python program, including the ability to run QMBasic programs, execute commands and read/write files. There is full support for locking, transactions, select lists, alternate key indices, instantiation and execution of QM object oriented programs and use of QM's connection pooling system.

Rather than executing the QM functions in the same process as the Python program, the QM implementation is based on the QMClient API. This has the important advantage that the two processes run with totally isolated memory images to ensure that an error in the Python program cannot corrupt internal data belonging to QM which could otherwise cause QM processes, perhaps even those unrelated to the Python activity, to fail in a manner that might be difficult to track down.

QMClient also allows the choice of running both processes on the local system or using a network connection to a remote QM server. A Python program can work with multiple simultaneous QM connections, including the ability for the Python process to be multi-threaded.

Use of the QMClient API also means that the developer can use the Extended Character Set (ECS) version of QM to access files that may be a mixture of 8-bit and Unicode data.

The simple example Python program below connects to the PY account on the local QM system, opens the ORDERS file, builds a select list of orders and then, for each order, prints the order number, customer number and order date.

```python
if qm.ConnectLocal("py"):
    fno = qm.Open("ORDERS")
    qm.Select(fno,1)
    while True:
        id = qm.ReadNext(1)
        if id == "": break
        rec, err = qm.Read(fno, id)
        date = qm.Extract(rec, 1, 0, 0)
        cust = qm.Extract(rec, 2, 0, 0)
        print(id, cust, qm.OConv(date, "D2DMYL[,A3]"))
    qm.Disconnect()
else:
    print("Failed to connect.",QMError())
```