Delphi4QM

A Delphi & Free Pascal Wrapper for the QMClient C API

Version 2.0.5

Trident Information Systems, Inc

Table of Contents

Part I

Introduction

Introduction

Overview

What is it?

Delphi4QM consists of two Pascal units that allow you to access the functionality provided by the QMClient C API (qmclilib.dll on Windows and libqmclient.a on Linux, OSX/BSD) from within your Delphi & Free Pascal (FPC) applications.

The QMClient API allows you to work with many aspects of your OpenQM database, such as session management, file handling, dynamic array manipulation, string manipulation, command and subroutine execution.

Until now, without a lot of work, the QMClient API was only available to developers using Visual Basic, PureBasic, or C. Now, with this API wrapper specifically written in Pascal, Delphi & FPC developers can also enjoy easy to use connectivity between their applications and OpenQM databases.

This API wrapper is written to ensure ease of use. No complicated pointer management or variable typecasting is required. Your calls to the QMClient API functions use and return only three variable types - Boolean, Integer, and AnsiString!

Adding QMClient connectivity to your Delphi & FPC applications is simple. Just add a single reference to the *Uses* clause of a Delphi/FPC unit, and the full QMClient API becomes available to that unit.

Features

- Compatible with Delphi & Free Pascal on Windows, Linux, and OSX/BSD
- Complete access to the full QMClient API
- No pointer management or typecasting. Use standard Pascal types
- Only a single unit reference added to your projects

Compatibility

Delphi: Tested D7, D2010, DXE. Should work with all versions between D7 and DXE. May work with older versions.
FPC: Tested 2.2.4 in compiler modes *Object Pascal* (-Mobjfpc) and *Delphi* (-Mdelphi). May work with older versions and other modes.
OS: Tested with Delphi/FPC on Win32 and with FPC on OSX & Linux.
QM: Tested versions 2.8-6 - 2.10-4 (Not all API functions are available for all versions. Check API documentation for your version of OpenQM)

If you experience problems, please contact support.

License

Delphi4QM is distributed under a BSD style license, the full text of which can be read here.

Home Page

Product information and download links are available here.

Quick Start

The following is a short guide to get you up and running quickly Those with a good understanding of Delphi or Free Pascal (FPC) should find it easy to follow.

Important

On Windows, the QMClient C API is provided by the dynamic link library **qmclilib.dll**. This dll must be available to any programs that use the QMClient C API. The easiest way to do this, is to make sure the dll is in a folder that resides on your workstation's execution *Path.* For simplicity, most people put this dll in the *\Windows\System32* folder.

On an OpenQM for Windows installation, *qmclilib.dll* is found in the subfolder *\QMSYS\bin*, under the installation folder.

On Linux, OSX and BSD, the QMClient C API is provided by linking the object library **libqmclient.a** to your FPC application. This library must be available to the FPC linker when you build your project, but does not have to be available to actually run your compiled application. On such OpenQM installations, *libqmclient.a* is found in the folder */usr/qmsys/bin*. Ensure that it is in your library path when building your projects.

No Unicode Support

OpenQM: As of version 2.10-4, OpenQM does not yet support Unicode. For that reason, and to maintain compatibility across all supported versions of Delphi and FPC, the default string type used by *Delphi4QM* is **AnsiString**.

Delphi: For versions of Delphi below 2009, the string types *AnsiString* and *String* are exactly the same. However, for Delphi versions 2009 and up, the default type for *String* is a *Unicode String*. If you attempt to use a variable of type *Unicode String* where an API return value type is *AnsiString*, it will be cast to a *Unicode String*, likely without any problems. However, if you use a *Unicode String* for an API parameter that expects an *AnsiString*, it will be cast to an *AnsiString* with possible data loss.

FPC: The Free Pascal type *String* is a generic type that either represents a *Short String* or an *AnisString*, depending on the compiler directives {$MODE} and {$LONGSTRINGS} or {$H}. For details, see the FPC on-line documentation using this link.

Installation & Use

Delphi4QM is provided as two Pascal units (text source files), named:

tisQMClient.pas (details)

tisQMClientAPI.pas (details)

You only need to add the unit reference *tisQMClient* to the *uses* clause of each project unit that you wish the QMClient API to be available. However, it is important to note that this unit requires *tisQMClientAPI*.

That's all there is to it. Use your IDE's Code Explorer/Browser to see a complete listing of all the functions and procedures found in *tisQMClient*. For usage details of each func/Proc, refer to the section on the QMClient API in the OpenQM reference manual.

Using the QMClient API

Once you've added *tisQMClient* to your project, you're ready to use the QMClient API.

Example:

This very simple example shows a function that connects to the QM Server, opens a QM file, reads a record, and extracts the first attribute from the record.

```
Function GetCustName(CustNo: AnsiString): AnsiString;
var
  err: Integer;
  CustFile: Integer;
  CustRecord: AnsiString;
begin

// Open a connection to the QM Server
If not QMConnect('ServerName',-1,'UserID','UserPass','QMAcctName') then begin
    showmessage('failed to connect');
    Exit;
  end;
  CustFile    := QMOpen('Customer-File')     ;// Open customer file
  CustRecord  := QMRead(CustFile,CustNo,err) ;// Read Customer record
  Result      := QMExtract(CustRecord,1,0,0));// Return 1st attribute to caller
end;
```

Note: QMClient sessions execute the LOGON paragraph for the account (if present), but not the MASTER.LOGON paragraph. You should not have commands within the LOGON paragraph that require user input/interaction. Refer to *QMConnect* under the *QMClient API* section of the *QM Reference Manual* for details.

Support

No Official Support

Delphi4QM **does not include any official support.**

However, questions, suggestions, and bug reports may be sent to support@tridentinfosys.com. You might not receive an immediate response to e-mail sent to that address. As stated, this software is not officially supported by Trident Information Systems.

Also, please do not use any official OpenQM support channels, such as the OpenQM Google newsgroup, to request support for this software.

Compatibility

Other versions of Delphi, Free Pascal, and OpenQM may work. If you experience problems, please contact support.

License

This software is distributed under a BSD style license, the full text of which can be read here.

Home Page

Product information and download links are available here.

License

Delphi4QM is distributed under a BSD style license.

The following copyright, terms and conditions of the license are also found in the source code.

Version History

Current Version

2.0.5

Release Notes

2.0.5 / March 2013
- Project renamed to Delphi4QM
- Updates to documentation and license text to reflect new project name
- No functional changes

2.0.4 / October 2010
- Updated QM Client API for QM version 2.10-4
- Delphi XE compatibility verification
- Corrected memory deallocation problem with QMError() API call.

2.0.3 / April 2010
- Corrected 3 API calls not linking on FPC due to method name casing errors.

2.0.2 / April 2010
- FPC compatibility update. Removed use of "Result" variables in functions for better FPC compiler mode compatibility.

2.0.1 / April 2010
- Minor changes to some method parameter types to prevent intermittent FPC compiler warnings.

2.0.0 / April 2010
- Updated for compatibility with:
  Delphi 7-2010 (Older versions may also work)
  Free Pascal 2.2.4 (Older versions may also work)

- Updated with all API items as of QM version 2.10-2

1.1.0 / November 2008
- Updated to include new API items:
  QMLogTo()
  QMRecordLock()
  QMTrapCallAbort()

1.0.0 / October 2006
- First Release

Part II

QMClient API

QMClient API

The OpenQM reference manual provides complete information on the QMClient API, potential security issues, and details of API function usage. There really is no need to duplicate that information here. However, the following API functions deserve some attention.

QMCall

QMConnect

QMCall

QMCall is defined as an overloaded procedure 22 times in the tisQMClient unit. The reason for this is twofold. First, it allows you to use the function with a variable argument list. Second, it gives you the flexibility to pass individual AnsiString arguments, or all arguments within a single AnsiString array.

Using QMCall with No Argument List

Syntax

QMCall(*SubrName***);**

where

SubrName	is an AnsiString value representing the name of a catalogued subroutine to call on the server

No arguments are passed to the subroutine and none are returned. Calling a subroutine that does not exist, or with a mismatched number of arguments, results in a failed call with no warnings or errors raised.

Using QMCall with 1-20 Individual Arguments

Syntax

QMCall(*SubrName, BuffSize, Arg1 [, Arg2] ...***);**

where

SubrName	is an AnsiString value representing the name of a catalogued subroutine to call on the server
BuffSize	is an integer value used to allocate character space for any values returned by the subroutine
Arg1[, Arg2]...	is a list of 1-20 AnsiString arguments, separated by commas. At least 1 argument is required.

BuffSize is applied to all arguments, but will never be smaller than the size of an argument passed to the subroutine. If you set *BuffSize* smaller than the size of one of your arguments, it is automatically increased to the size of that argument, *for that argument only*. Taking this into consideration, if your subroutine returns values that are larger than the value of *BuffSize,* You will corrupt memory and your program is likely to crash.

Calling a subroutine that does not exist, or with a mismatched number of arguments, results in a failed call with no warnings or errors raised.

Using QMCall with an AnsiString Array

Syntax

QMCall(*SubrName, BuffSize, sArray***);**

where

SubrName	is a AnsiString value representing the name of a catalogued subroutine to call on the server
BuffSize	is an integer value used to allocate character space for any values returned by the subroutine
sArray	is a one dimension array of AnsiStrings representing 1-20 arguments, indexed from 0 to 19

BuffSize is applied to all arguments (*sArray* elements), but will never be smaller than the size of an argument (*sArray* element) passed to the subroutine. If you set *BuffSize* smaller than the size of one of your arguments (*sArray* elements), it is automatically increased to the size of that argument/element, *for that argument/element only*. Taking this into consideration, if your subroutine returns values that are larger than the value of *BuffSize,* You will corrupt memory and your program is likely to crash.

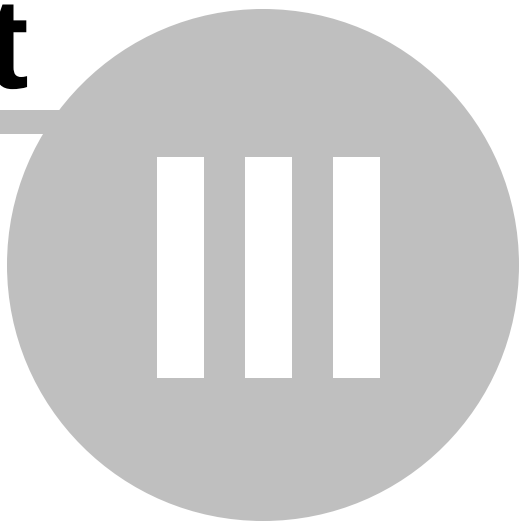
Calling a subroutine that does not exist, or with a mismatched number of arguments, results in a failed call with no warnings or errors raised.

QMConnect

QMConnect - QMClient sessions execute the LOGON paragraph (if present) for the account, but not the MASTER.LOGON paragraph. You should not have commands within the LOGON paragraph that require user input/interaction. Refer to *QMConnect* under the *QMClient API* section of the *QM Reference Manua*l for details.

Part

Units



Units

Delphi4QM consists of two Pascal units (.pas source files).

tisQMClient	A wrapper for tisQMClientAPI. Provides clean, consistent interface to the developer. Hides tedious implementation details.
tisQMClientAPI	A wrapper for the QMClient C API. Contains external declarations for QMClient API functions. Exported by **qmclilib.dll** and **libqmclient.a** on Linux, and OSX/ BSD.

tisQMClient

tisQMClient is a Pascal unit source file (tisQMClient.pas) that is a wrapper for tisQMClientAPI.

This unit hides tedious implementation details, such as pointers/typecasting and function overloading, to provide a clean, consistent, and easy to use interface to the QMClient C API.

Under most circumstances, a developer should reference only this unit in their Delphi or Free Pascal project in order to access the QMClient API.

Defined Constants

```
SV_OK = 0;          // * Action successful
SV_ON_ERROR = 1; // * Action took on ERROR clause
SV_ELSE = 2;        // * Action took else clause
SV_ERROR = 3;       // * Action failed.  Error text available
SV_LOCKED = 4;      // * Action took LOCKED clause
SV_PROMPT = 5;      // * Server requesting input

QMDateOffset = 24837; // * Add to QM Dates to get TDate value
```

Function & Procedure Declarations

```
procedure QMCall( const SubrName: AnsiString); overload;
procedure QMCall( const SubrName: AnsiString;  const BuffSize: Integer;  var a1: AnsiString); overload;
procedure QMCall( const SubrName: AnsiString;  const BuffSize: Integer;  var a1, a2: AnsiString);overload;
```

[Additional definitions for QMCall with 3 to 20 arguments not listed]

```
procedure QMCall(const SubrName: AnsiString; const BuffSize: Integer; var Args: array of AnsiString); overloa
function QMChange(const SrcString, OldString, NewString: AnsiString; const Occurences, Start: Integer): AnsiSt
procedure QMClearSelect(const ListNo: Integer);
procedure QMClose(const FileNo: Integer);
function QMConnect(const Host: AnsiString; PortNo: Integer; const UserName, Password, Account: AnsiString): Bo
function QMConnected: Boolean;
procedure QMConnectionType(const ConType: Integer);
function QMConnectLocal(const Account: AnsiString): Boolean;
function QMDcount(const SrcString, Delimiter: AnsiString): Integer;
function QMDel(const SrcString: AnsiString; const FieldNo, ValeuNo, SubValueNo: Integer): AnsiString;
procedure QMDelete(const FileNo: Integer; const ItemID: AnsiString);
procedure QMDeleteU(const FileNo: Integer; const ItemID: AnsiString);
procedure QMDisconnect;
procedure QMDisconnectAll;
procedure QMEndCommand;
function QMError: AnsiString;
function QMExecute(const Command: AnsiString; var ErrNo: Integer): AnsiString;
function QMExtract(const SrcString: AnsiString; const FieldNo, ValueNo, SubValueNo: Integer): AnsiString;
function QMField(const SrcString, Delimiter: AnsiString; const First, Occurences: Integer): AnsiString;
function QMGetSession: Integer;
function QMIns(const SrcString: AnsiString; FieldNo, ValueNo, SubValueNo: Integer; const NewString: AnsiStrin
function QMLocate(const Value, SrcString: AnsiString; FieldNo, ValueNo, SubValueNo: Integer; var Pos: Integer;
function QMLogto(const Account: AnsiString): Boolean;
procedure QMMarkMapping(const FileNo, State: Integer);
function QMMatch(const SrcString, Pattern: AnsiString): Boolean;
function QMMatchField(const SrcString, Pattern: AnsiString; const Component: Integer): AnsiString;
function QMNextPartial(const ListNo: Integer): AnsiString;
function QMOpen(const FileName: AnsiString): Integer;
function QMRead(const FileNo: Integer; ItemID: AnsiString; var ErrNo: Integer): AnsiString;
function QMReadL(const FileNo: Integer; const ItemID: AnsiString; const Wait: Boolean; var ErrNo: Integer): An
function QMReadList(const ListNo: Integer): AnsiString;
```

```
function QMReadNext(const ListNo: Integer): AnsiString;
function QMReadU(const FileNo: Integer; const ItemID: AnsiString; const Wait: Boolean; var ErrNo: Integer): An
procedure QMRecordLock(const FileNo: Integer; const ItemID: AnsiString; const Update, Wait: boolean);
procedure QMRelease(const FileNo: Integer; ItemID: AnsiString);
function QMReplace(const SrcString: AnsiString; const FieldNo, ValueNo, SubValueNo: Integer; const NewString:
function QMRespond(const Response: AnsiString; var ErrNo: Integer): AnsiString;
procedure QMSelect(const FileNo, ListNo: Integer);
procedure QMSelectIndex(const FileNo: Integer; const IndexName, IndexValue: AnsiString; const ListNo: Integer)
function QMSelectLeft(const FileNo: Integer; const IndexName: AnsiString; const ListNo: Integer): AnsiString;
function QMSelectPartial(const FileNo, ListNo: Integer): AnsiString;
function QMSelectRight(const FileNo: Integer; const IndexName: AnsiString; const ListNo: Integer): AnsiString;
procedure QMSetLeft(const FileNo: Integer; const IndexName: AnsiString);
procedure QMSetRight(const FileNo: Integer; const IndexName: AnsiString);
function QMSetSession(const SessionNo: Integer): Boolean;
function QMStatus: Integer;
procedure QMTrapCallAbort(const Mode: Boolean);
procedure QMWrite(const FileNo: Integer; const ItemID, Item: AnsiString);
procedure QMWriteU(const FileNo: Integer; const ItemID, Item: AnsiString);
```

tisQMClientAPI

tisQMClientAPI is an Pascal unit source file (tisQMClientAPI.pas) that is a wrapper for the QMClient C API. It contains external declarations for QMClient API functions. On Windows platforms, these are exported by **qmclilib.dll**. On Linux and OSX/BSD platforms these are contained within the library file **libqmclient.a**.

This unit *can* be used directly in a project, but is usually not. It is much easier to add QMClient API functionality to your projects by using the unit tisQMClient, which wraps the API functions exposed by this unit into a clean, consistent, and easy to use interface.

Function & Procedure Declarations

```
procedure QMCall(subrname: PAnsiChar; argc: ShortInt);overload; cdecl;
procedure QMCall(subrname: PAnsiChar; argc: ShortInt; a1: PAnsiChar);overload; cdecl;
procedure QMCall(subrname: PAnsiChar; argc: ShortInt; a1, a2: PAnsiChar); overload; cdecl;
```

[Additional definitions for _QMCall with 3 to 20 arguments not listed]

```
function QMChange(src, old_string, new_string: PAnsiChar; occurrences, start: Integer): PAnsiChar; cdecl;
procedure QMClearSelect(listno: Integer); cdecl;
procedure QMClose(fno: Integer); cdecl;
function QMConnect(host: PAnsiChar; port: Integer; username, password, account: PAnsiChar): Integer; cdecl;
function QMConnected: Integer; cdecl;
procedure QMConnectionType(con_type: Integer); cdecl;
function QMConnectLocal(account: PAnsiChar): Integer; cdecl;
function QMDcount(src, delim: PAnsiChar): Integer; cdecl;
function QMDel(src: PAnsiChar; fno, vno, svno: Integer): PAnsiChar; cdecl;
procedure QMDelete(fno: Integer; id: PAnsiChar); cdecl;
procedure QMDeleteu(fno: Integer; id: PAnsiChar); cdecl;
procedure QMDisconnect; cdecl;
procedure QMDisconnectAll; cdecl;
procedure QMEndCommand; cdecl;
function QMError: PAnsiChar; cdecl;
function QMExecute(cmnd: PAnsiChar; var err: Integer): PAnsiChar; cdecl;
function QMExtract(src: PAnsiChar; fno, vno, svno: Integer): PAnsiChar; cdecl;
function QMField(src, delim: PAnsiChar; frist, occurences: Integer): PAnsiChar; cdecl;
procedure QMFree(p: PAnsiChar); cdecl;
function QMGetSession: Integer; cdecl;
function QMIns(src: PAnsiChar; fno, vno, svno: Integer; new_string: PAnsiChar): PAnsiChar; cdecl;
function QMLocate(item, src: PAnsiChar; fno, vno, svno: Integer; var pos: Integer; order: PAnsiChar): Integer;
function QMLogto(account: PAnsiChar): Integer; cdecl;
procedure QMMarkMapping(fno, state: Integer); cdecl;
function QMMatch(str, pattern: PAnsiChar): Integer; cdecl;
function QMMatchField(str, pattern: PAnsiChar; component: Integer): PAnsiChar; cdecl;
function QMNextPartial(listno: Integer): PAnsiChar; cdecl;
function QMOpen(filename: PAnsiChar): Integer; cdecl;
function QMRead(fno: Integer; id: PAnsiChar; var err: Integer): PAnsiChar; cdecl;
function QMReadL(fno: Integer; id: PAnsiChar; wait: Integer; var err: Integer): PAnsiChar; cdecl;
function QMReadList(listno: Integer): PAnsiChar; cdecl;
function QMReadNext(listno: Integer): PAnsiChar; cdecl;
function QMReadu(fno: Integer; id: PAnsiChar; wait: Integer; var err: Integer): PAnsiChar; cdecl;
procedure QMRecordLock(fno: Integer; id: PAnsiChar; update: Integer; wait: Integer); cdecl;
procedure QMRelease(fno: Integer; id: PAnsiChar); cdecl;
function QMReplace(src: PAnsiChar; fno, vno, svno: Integer; new_string: PAnsiChar): PAnsiChar; cdecl;
function QMRespond(response: PAnsiChar; var err: Integer): PAnsiChar; cdecl;
procedure QMSelect(fno, listno: Integer); cdecl;
procedure QMSelectIndex(fno: Integer; indexname, indexvalue: PAnsiChar; listno: Integer); cdecl;
function QMSelectLeft(fno: Integer; indexname: PAnsiChar; listno: Integer): PAnsiChar; cdecl;
function QMSelectPartial(fno, listno: Integer): PAnsiChar; cdecl;
function QMSelectRight(fno: Integer; indexname: PAnsiChar; listno: Integer): PAnsiChar; cdecl;
```

```
procedure QMSetLeft(fno: Integer; indexname: PAnsiChar); cdecl;
procedure QMSetRight(fno: Integer; indexname: PAnsiChar); cdecl;
function QMSetSession(session: Integer): Integer; cdecl;
function QMStatus: Integer; cdecl;
procedure QMTrapCallAbort(mode: Integer); cdecl;
procedure QMWrite(fno: Integer; id, data: PAnsiChar); cdecl;
procedure QMWriteu(fno: Integer; id, data: PAnsiChar); cdecl;
```

Index