# 

# The win32com package

This document describes the win32com package in general terms.

The COM support can be thought of as existing in 2 main portions - the C++ support code (the core PythonCOM module), and helper code, implemented in Python. The total package is known as "win32com".

The win32com support is stand-alone. It does not require Pythonwin.

## The win32com package

To facilitate an orderly framework, the Python "ni" module has been used, and the entire package is known as "win32com". As is normal for such packages, win32com itself does not provide any functionality. Some of the modules are described below:

* **win32com.pythoncom - core C++ support**.  
  This module is rarely used directly by programmers - instead the other "helper" module are used, which themselves draw on the core pythoncom services.
* **win32com.client package**  
  Support for COM clients used by Python. Some of the modules in this package allow for dynamic usage of COM clients, a module for generating .py files for certain COM servers, etc.
* **win32com.server package**  
  Support for COM servers written in Python. The modules in this package provide most of the underlying framework for magically turning Python classes into COM servers, exposing the correct public methods, registering your server in the registry, etc.
* **win32com.axscript**  
  ActiveX Scripting implementation for Python.
* **win32com.axdebug**  
  Active Debugging implementation for Python
* **win32com.mapi**  
  Utilities for working with MAPI and the Microsoft Exchange Server

## The pythoncom module

The pythoncom module is the underlying C++ support for all COM related objects. In general, Python programmers will not use this module directly, but use win32com helper classes and functions.

This module exposes a C++ like interface to COM - there are objects implemented in pythoncom that have methods "QueryInterface()", "Invoke()", just like the C++ API. If you are using COM in C++, you would not call a method directly, you would use pObject->Invoke( �, MethodId, argArray�). Similarly, if you are using pythoncom directly, you must also use the Invoke method to call an object's exposed method.

There are some Python wrappers for hiding this raw interface, meaning you should almost never need to use the pythoncom module directly. These helpers translate a "natural" looking interface (eg, obj.SomeMethod()) into the underlying Invoke call.