# Windows Setup

## Installing the Support Software

* Install Windows 2008 R2 (64-bit).
* Add .NET Framework 3.5.1:
  + Server Manager-->Features
  + Add Features (on the right side)
  + Choose NET Framework 3.5.1 (first option)
  + Accept all of the defaults
  + Click "Add Required Role Services"
  + Click Next, Next, Install
* Download and Install Visual Studio. Two options:
  + Download from outside Windows, then make the ISO available through the VM. Double-click on the "Setup" program on the ISO.
  + You can download it from inside Windows, where it looks like it could start the Install process after downloading.
  + Click on "Install"
    - Accept all of the defaults (making sure you select the C# language libraries you run setup, the other languages, Visual Basic and C++ are not necessary)
* Install Msysgit (<https://code.google.com/p/msysgit/downloads/list?q=full+installer+official+git>)
* Copy [telnet.exe](https://remote.akirisol.com:5045/trac/palette/files/telnet.exe) from the Shared Files area on this [DevBox](https://www.assembla.com/wiki/show/palette-software/DevBox" \o "DevBox) into the bin directory of Msysgit: **C:\Program Files (x86)\Git\bin**.
* Install python 2.7.6 (<http://www.python.org/getit/releases/2.7.6/>) x86-64, not x86.
* Install tortoisegit (<https://code.google.com/p/tortoisegit/wiki/Download>) (for 64-bit OS)
  + Accept all defaults for everything starting with: Choose "[TortoiseGitPink](https://www.assembla.com/wiki/show/palette-software/TortoiseGitPink" \o "TortoiseGitPink), coming from Putty, integrates with Windows better".
* Get ssh working between Windows and the Palette [DevBox](https://www.assembla.com/wiki/show/palette-software/DevBox" \o "DevBox) so "git clone", etc. will work.
* Start-->All Programs-->Git-->Git Bash and then create an **.ssh** directory in the home directory:
* mkdir .ssh
  + Run [PuTTY](https://www.assembla.com/wiki/show/palette-software/PuTTY" \o "PuTTY) Key Generator: Start-->All Programs-->[TortoiseGit](https://www.assembla.com/wiki/show/palette-software/TortoiseGit" \o "TortoiseGit)-->Puttygen
  + Under "Actions", to the right of "Generate a public/private key pair", click "Generate". After random mousing:
    - In [PuTTY](https://www.assembla.com/wiki/show/palette-software/PuTTY" \o "PuTTY) Key Generator Click "Conversions-->Export [OpenSSH](https://www.assembla.com/wiki/show/palette-software/OpenSSH" \o "OpenSSH) Key" and save to **.ssh\id\_rsa** (no extension) in the home directory (e.g. C:\Users\Administrator\.ssh).
    - In the top area under "Public key for pasting into [OpenSSH](https://www.assembla.com/wiki/show/palette-software/OpenSSH" \o "OpenSSH) authorized\_keys file", select all of the key and then click the right mouse button and Copy to copy the public key.
      * Login to the Palette [DevBox](https://www.assembla.com/wiki/show/palette-software/DevBox" \o "DevBox), click **Settings** then paste the public key into the text area and click **Add Key**.
    - Click **Save private key** (bottom right) and save anywhere under any name in case you need to use it again. (You can use it again in Putty Key Generator File-->Load Private Key).
  + Test to see if the private key is working:
  + ssh <your-devbox-username>@remote.akirisol.com -p 5046
* Install 7zip (<http://www.7-zip.org/download.html>). (64-bit x64 version)
* Git clone the 3 repositories from the Palette [DevBox](https://www.assembla.com/wiki/show/palette-software/DevBox" \o "DevBox) (agent, controller, and lib).  Make sure these 3 repositories are at the same directory level on your machine.

### Visual Studio Configuration and Solution Build

* Configure Windows Explorer to not hide file name extensions:
  + Start Windows Explorer, you can do this by opening up any folder.
  + Click Tools, and then click Folder Options. If you don't see the Tools menu, click: Organize-->Layout-->Menu Bar.
  + Scroll down and then click Folder and search options.
  + Click the View tab.
  + Scroll down until you notice Hide extensions for known file types, un-check this line by clicking the check box.
  + Click OK
* In the same parent folder as the "Agent" Visual Studio Solution, there should be a "lib" folder.  The following files should be there:
  + log4net.dll
  + fastJSON.dll
* If not already open from the previous step, open the Visual Studio project: With Windows Explorer, navigate to your palette [DevBox](https://www.assembla.com/wiki/show/palette-software/DevBox" \o "DevBox) repository for "agent" and open agent/[WindowsAgent](https://www.assembla.com/wiki/show/palette-software/WindowsAgent" \o "WindowsAgent).soln
  + Choose Visual C# Development Settings
  + Select Build->Clean Solution
  + Select Build->Build Solution

### If installing a controller on windows

### Postgres Configuration

* Install pgadmin: <http://www.pgadmin.org/download/windows.php>
* Start pgadmin. You'll need to create a superuser named "palette" and a new database named "paldb".
* First connect to the Postgres database:
  + File-->Add Server...
    - Name: tableau
    - Host: localhost
    - Port: 8060
    - Username: tblwgadmin
    - Password: tblwgadmin
    - Click OK
  + Double Click to expand "tableau (localhost:8060)" in the left "Object browser" area.
* Create the 'palette' super-user:
  + Right click on "Login Roles(3)" and select "New Login Role..."
    - For Role name, enter "palette"
    - Click the "Definition" tab and enter the password **palpass** twice.
    - Click the "Role privileges" tab and click/enable **Superuser** and **Can create databases**.
    - Click OK.
* Create the 'paldb' database:
  + Back at the left "Object browser" right click on "Databases(3)" and select "New Database..."
    - For "Name", enter "paldb".
    - For "Owner" enter "tblwgadmin"
    - Click "OK"
* Install sqlalchemy (<http://www.sqlalchemy.org/download.html>)
  + Download the SQLalchemy-X.X.X.tar.gz file.
  + Start-->Git Bash
    - Set your environment to find python:

export PATH=$PATH:/c/Python27

* Then cd to the directory where SQLAlchemy was downloaded, extract the files from the SQLAlchemy tar file, cd to it and install it:
* cd Downloads
* tar xvfz SQLAlchemy-0.9.1
* cd SQLAlchemy-0.9.1
* python setup.py install
* Install the pyscopg python package: <http://www.stickpeople.com/projects/python/win-psycopg/> Choose (For Python 2.7 amd64) (64bit Windows)

### Run it on Windows

* Start the controller (if you followed the additional controller-on-windows instructions, above):
  + Start a Git Bash window, 'cd' to the repo controller directory and type:
  + export PATH=$PATH:/c/Python27
  + python controller.py
* Start the agent:
  + From Visual Studio, build the agent by pressing F6
  + Run the agent by pressing F5
* Telnet to the controller and send it commands:
  + Start a git Bash window and type:
  + telnet localhost 9000

status

## To install the Windows Service using Visual Studio

* Make sure you have the current solution from the Repository (see "Visual Studio Configuration and Solution Build" above)
* Start Visual Studio as Administrator (right click on the shortcut and choose "run as administrator")
* If you haven't done so, Clean and Build the project (Build->Clean and then Build->Build)
* To install the Windows service, right-click the setup project ("[ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent" \o "ServiceAgent) Setup") in the Solution Explorer and select Install (note you can uninstall the same way)
* A Wizard will appear to walk you through the installation.  The default installation folder is C:/Palette.  If you choose another folder, the service will be installed there but currently the supporting files will be installed in C:/Palette (there is a ticket to change this so everything is installed in the same folder)

## To install the Windows Service/Console App using the .msi file

* Go to the location of the .msi file - if you are building using VS the it will be in the Debug or Release subfolder of [ServiceAgentSetup](https://www.assembla.com/wiki/show/palette-software/ServiceAgentSetup" \o "ServiceAgentSetup).
* Right click on [ServiceAgentSetup](https://www.assembla.com/wiki/show/palette-software/ServiceAgentSetup).msi and choose "Install" (note you can uninstall the same way)
* A Wizard will appear to walk you through the installation.  Use the default folder.

### To start and stop the service

* Open the Services Control Manager by doing the following:
* Open the Windows Control Panel
* Click on “System and Security”
* Click on “Administrative Tools”
* Double click on the “Services” icon
* Look for a Service with the name “[ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent" \o "ServiceAgent)”.  It should have a “Manual” startup type
* Right click on the Service and click “Properties”.  On the “Log On” tab, check “Allow Service to interact with desktop” (see below)
* Right click the service and select “start”
* To stop the service, right click and select “stop”

## To uninstall the Service and Console App

* In Visual Studio, right-click on the [ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent" \o "ServiceAgent) Setup project and select "Uninstall" (or) Right click on the [ServiceAgentSetup](https://www.assembla.com/wiki/show/palette-software/ServiceAgentSetup).msi file in [ServiceAgentSetup](https://www.assembla.com/wiki/show/palette-software/ServiceAgentSetup" \o "ServiceAgentSetup)\Debug and click "uninstall" (or) go to Control panel, Programs->Uninstall Program and select "[ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent" \o "ServiceAgent)".
* The Service will stop and be removed, all of the installation files will be removed except for any created at runtime.  We are working on removing these for 0.1.

To uninstall the Service manually if process fails during uninstallThis is unlikely to happen but if it does difficult to fix without doing the following:

* Open a console window
* Type "sc delete [ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent" \o "ServiceAgent)"
* You should see the message: "[SC] [DeleteService](https://www.assembla.com/wiki/show/palette-software/DeleteService" \o "DeleteService) SUCCESS"
* Delete the folder that the service was installed to (the default is C:/Palette/[ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent))
* If you can't install or uninstall the [ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent" \o "ServiceAgent) because of a failed uninstall, go to "HKEY\_USERS" in Registry, search for "[ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent" \o "ServiceAgent)". Delete the folder containing a reference to it in \Software\Microsoft\Installer\Products (the folder has a guid-type name so it may be different for any particular installation).  After doing this you can re-install.

### To run the [ConsoleAgent](https://www.assembla.com/wiki/show/palette-software/ConsoleAgent" \o "ConsoleAgent) in Visual Studio

* In the Solution Explorer, right click on the "[ConsoleAgent](https://www.assembla.com/wiki/show/palette-software/ConsoleAgent" \o "ConsoleAgent)" project and select "Set as Start up Project".  The text for this project will become **BOLD**
* Right Click on the project and select "Properties".  Under the "Debug" tab, put in the location of your desired .ini file under "Command Line Arguments".  The default is "C:\Palette\agent\conf\primary.ini".
* Select Debug->Start Without Debugging  
  To run the [ConsoleAgent](https://www.assembla.com/wiki/show/palette-software/ConsoleAgent" \o "ConsoleAgent) in a Console Window without Visual Studio
* Open a Console Window and go to the folder you have installed the [ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent" \o "ServiceAgent) into (by default "C:\Palette\[ServiceAgent](https://www.assembla.com/wiki/show/palette-software/ServiceAgent)" if you chose the default install)
* Type "[ConsoleAgent](https://www.assembla.com/wiki/show/palette-software/ConsoleAgent).exe [ini file]" where "ini file" is the name of your chosen .ini file.  The default is C:\Palette\conf\primary.ini.

## Logging

The Agent uses log4net to write output to a log file.  The binary for this is in the "lib" repository, but to download the latest version of this this, go to http://logging.apache.org/log4net/download\_log4net.cgi and download the latest version.  To change the project to use this version,

* Open the zip file under C:\Palette
* In the [WindowsAgent](https://www.assembla.com/wiki/show/palette-software/WindowsAgent" \o "WindowsAgent) solution, go to the [ConsoleAgent](https://www.assembla.com/wiki/show/palette-software/ConsoleAgent" \o "ConsoleAgent) project in the Solution Explorer, and right click References
* Select "Add Reference"
* Under the browse tab, select C:/Palette/log4net-X.XX.XX/bin/net/3.5/release/log4net.dll  (where "log4net-X.XX.XX" is the latest version

The log file is stored by default in c:/Palette/log but can be configured in the ini file

There are three types of logging defined in the file, [FileAppender](https://www.assembla.com/wiki/show/palette-software/FileAppender" \o "FileAppender), [RollingFileAppender](https://www.assembla.com/wiki/show/palette-software/RollingFileAppender" \o "RollingFileAppender), and [ConsoleAppender](https://www.assembla.com/wiki/show/palette-software/ConsoleAppender" \o "ConsoleAppender).  Currently we are using the [RollingFileAppender](https://www.assembla.com/wiki/show/palette-software/RollingFileAppender" \o "RollingFileAppender) for the service and both the[RollingFileAppender](https://www.assembla.com/wiki/show/palette-software/RollingFileAppender) and [ConsoleAppender](https://www.assembla.com/wiki/show/palette-software/ConsoleAppender" \o "ConsoleAppender) for the [ConsoleAgent](https://www.assembla.com/wiki/show/palette-software/ConsoleAgent" \o "ConsoleAgent).  The [RollingFileAppender](https://www.assembla.com/wiki/show/palette-software/RollingFileAppender" \o "RollingFileAppender) will write a log of "[MaxLogSize](https://www.assembla.com/wiki/show/palette-software/MaxLogSize" \o "MaxLogSize)" (set in the .ini file) before creating a new file.

There are seven logging levels, five of which can be called in the code. They are as follows (with the highest being at the top of the list):

*OFF - nothing gets logged (cannot be called)*

*FATAL*

*ERROR*

*WARN*

*INFO*

*DEBUG*

*ALL - everything gets logged (cannot be called)*

Each Appender has a min and max level defined.  Currently they are set at min = INFO and max = FATAL.  No debug messages will show unless this is changed to min = DEBUG.