**LillTek Developer Setup**

**Default File Install Point**

**C:\LillTek**

**C:\NeonSwitch**

Although in theory, you should be able to locate this folder anywhere in the File System, other projects may depend on this folder being located here and it’s very likely that including blanks in this path may break CMD and Build Action scripts.

**Instructions**

Install: Make sure that **IIS/ASP.NET** is installed using **Control Panel/Programs and Features**. (I **enable all features** except for FTP). Also include the **Internet Information Services Hostable Web Core**.  
  
Make sure that application **pool defaults** are configured for **.NET 4.5**.

Config: Run **iis-sites.cmd** as **administrator** to initialize the LillTek related web sites.

Config: **Windows Update** to update all **Microsoft Products**.

Install: All **Windows Updates** including all **.NET Framework** redistributables.

Install: **Visual Studio 2013 Ultimate**  
  
Professional will work but Ultimate is preferred to pick up additional features such as Profiling.

Install: **WiX Toolset** from here: <http://wixtoolset.org>

Install: **Microsoft Speech Platform** from [here](http://www.microsoft.com/en-us/download/details.aspx?id=27225).

Install: **Microsoft Speech Platform Languages** from [here](http://www.microsoft.com/en-us/download/details.aspx?id=27224).

Config: LillTek projects use the default VS editor settings. It is **very important for all developers** to standardize on these settings to prevent Visual Studio from reformatting large sections of the source code automatically resulting in unnecessary and confusing source control modifications.

Config: Configure **VS2013** to run **as administrator** by:  
  
\* Right-clicking on the VS shortcut in the Start Menu  
 on the desktop.  
\* Select the **Advanced tab**  
\* Check: **Run this program as an Administrator**

Optional: Disable the Visual Studio Debugger Attach Security warning by searching the registry for any **DisableAttachSecurityWarning** values and changing them from **0** them to **1**.

Install: **SQL/Server 2012** or greater. Download this from MSDN. Be sure to enable **Mixed Mode** security and grant the **developer Windows account** must have **administrator rights** to the server as well (go for a walk or something; installation takes forever).

Config: Run **Start\All Programs\SQL Server 2012 R2\Configuration Tools\SQL Server Configuration Manager**.  
  
Double-click on **SQL Server Network Configuration**.  
  
Click on **Protocols for MSSQLSERVER**.  
  
Right-click on **TCP/IP** in the right hand pane and **select Enable TCP/IP**. **Enable Named Pipes** as well. Close the configuration tool.

**Reboot:** Your machine to make sure all pending setup operations have completed.

Enlist: LillTek project management is hosted in Visual Studio Online at **lilltek.visualstudio.com**.  
  
**todo**: add detailed enlistment instructions for both Main and NeonSwitch.

Run: **Right-click** on **C:\LillTek\buildvars.cmd** and **Run as Administrator**. This sets up all of the LillTek related build **environment variables**.

Install: **Windows Azure SDK** from [here](http://www.windowsazure.com/en-us/develop/net/).

Optional: **Fiddler4** from [here](http://fiddler2.com/get-fiddler).

Optional: Configure your **virus checker to disable checking** of the following folders:  
  
**C:\NeonSwitch   
C:\LillTek**

Optional: Open **C:\NeonSWITCH\NeonSWITCH.2012.sln** and **Rebuild** the **Debug** solution configuration.

Install: Microsoft **Sandcastle** Documentation Generator and GUI Tools from [here](http://www.lilltek.com/install/sandcastle.zip) and follow these instructions:

1. Install **Sandcastle.msi**
2. Extract the MSI from **SandcastleBuilderSetup\_1604.zip** and install it.
3. Extract the contents of **PresentationPatch.zip** to the Microsoft Sandcastle installation folder (overwriting any existing files):   
     
   **C:\Program Files (x86)\Sandcastle**

Install: (Optional) **NetMon** (complete) from [here](http://www.microsoft.com/downloads/details.aspx?FamilyID=983b941d-06cb-4658-b7f6-3088333d062f&displaylang=en).

Security: The **NETWORK SERVICE** account does not have rights to access the event log by default. You need to manually grant these rights:

1. Start **RegEdit**
2. Navigate to:  
     
   **HKLM\SYSTEM\CurrentControlSet\Services\EventLog\Application**
3. Right click the Application folder and select **Permissions**…
4. Add **NETWORK SERVICE** to the set of users and click **Full Control**.
5. **Restart** the **World Wide Web** service

Install: **Microsoft Distributed Transaction Coordinator** (MSDTC):

1. Start **CMD.EXE** with **Run as administrator**.
2. Execute: **msdtc –install**
3. Execute: **dcomcnfg**
4. Open **Component** **Services\Computers\My Computer\Distributed Transaction Coordinator** in the left pane.
5. Allow access for the **Windows Firewall** for all networks (if Windows Firewall launches).
6. Start the **Administrative Tools\Services** control panel and configure the **Distributed Transaction Coordinator** for **automatic** start.

IIS: Configure the following local **IIS websites** (making sure that the **application pools are configured for .NET 4.0**):

**127.0.0.1 -- IP address only, no host name**

* **C:\LillTek\Test\Web**

Run: **Windows Update** until there are no more patches are applied.

Run: **Reboot**

**LillTek Environment Variables**

The LillTek build requires several environment variables. Normally, you’ll use **C:\LillTek\Main\buildvars.cmd** batch file to configure these, but can also set these up manually using the **System Control Panel** (Advanced/Environment Variables).

**DOTNETPATH=%WINDIR%\Microsoft.NET\Framework\v4.0.30319  
  
DEV\_WORKSTATION=1**

**LT\_ROOT=C:\LillTek**

**LT\_TOOLBIN=%LT\_ROOT%\ToolBin**

**LT\_TESTBIN=%LT\_ROOT%\TestBin**

**LT\_BUILD=%LT\_ROOT%\Build**

**LT\_BUILD\_DBPACK=1**

**LT\_TEMP=C:\Temp**

**NEONSWITCH=C:\NeonSwitch  
NEONSWITCH\_DEBUGBIN=%NEONSWITCH%\Debug**

**NEONSWITCH\_RELEASEBIN=%NEONSWITCH%\Release  
NEONSWITCH\_DEBUGMANAGED=%NEONSWITCH\_DEBUGBIN%\mod\managed**

**NEONSWITCH\_RELEASEMANAGED=%NEONSWITCH\_RELEASEBIN%\mod\managed**  
 **PATH=%PATH%;%LT\_BIN%;%LT\_BUILD%**

Two optional environment variables can be set for UNIT testing purposes.

**LT\_TEST\_DB** Specifies the connection string to be used for test suites that need access to a SQL Server database. The test suites will default to the local **SQLExpress** instance using the active user account if this variable is not present.

**LT\_TEST\_AD** Specifies the credentials to be used to authenticate against the current active directory. This will be typically set up as a test account. This variable must be set for all test suites to work properly. The credentials must be formatted as:

**servers=domain01;nasSecret=my-secret;  
domain=lilltek.com;account=test;password=mypassword**

where **servers** lists the domain controller host names or IP addresses separated by commas, **nasSecret** is the RADIUS shared secret to be used for IAS authentications, **domain** is the fully qualified name of the domain, **account** is the domain account, and **password** is the account password.

Note that omitting **nasSecret** or setting it to blank will disable AD/IAS tests. Note that this secret will have to be manually configured into IAS for the appropriate RADIUS clients for the tests to function properly.

**Other Steps**

**SQL Server** must be configured for **Mixed Mode** security. This is not the default after installing SQL Express. To set mixed mode security:

1. Start **SQL Server Management Studio**
2. Connect to **SQL**
3. **Right-click** the server in the left pane (the server instance) and select **Properties**.
4. Select **Security** in the left pane of the dialog
5. Select the **SQL Server and Windows Authentication mode** radio button.
6. Click **OK** and **restart** the SQL service.

**NeonSwitch Notes**

The NeonSwitch project is primarily hosted in the LillTek NeonSwitch Git repository. This contains the small number of proprietary modifications we’ve made to FreeSWITCH to support the loading of NeonSwitch .NET applications. This was originally a clone of the master branch of the FreeSWITCH.org Git repository and we will periodically pull updates from FreeSWITCH.org to pick up bug fixes and other enhancements.

Most of the NeonSwitch code is actually hosted in the LillTek TFS repository under **Main\Platform\Telephony**. This code depends on the **FreeSWITCH.Managed.dll** assembly built from the NeonSwitch solution. This assembly includes the proxy code to bridge from the managed NeonSwitch assemblies to unmanaged base FreeSWITCH. The Main projects reference a copy of this assembly from:

**C:\LillTek\Main\External\FreeSWITCH\Lib\FreeSWITCH.Managed.dll**

This means than whenever we pull updated source from FreeSWITCH.org, we should rebuild this assembly from the NeonSwitch solution and copy the results from the folder below to the Main project folder above to make sure both repositories are in sync.

**C:\NeonSwitch\x64\Release\mod\FreeSWITCH.Managed.dll**

I typically work with NeonSwitch applications from the LillTek Platform solution by configuring the debugger to launch the debug NeonSWITCH console application for the assemblies I’m working on:

**C:\NeonSwitch\x64\Debug\NeonSwitchConsole.exe**

Note that the Platform NeonSwitch related projects have build events that copy their build outputs to the appropriate build output folders in the NeonSwitch project so F5 debugging will just work.

Note that FreeSWITCH really works only for 64-bit environments, so that’s all we’re going to support for NeonSwitch as well.

**ConfigOverride.txt File**

LillTek messaging discovers other endpoints using UDP multicasts. This is great when running applications within a data center but this can cause conflicts when multiple developers or testers unknowingly deploy service instances on the same UDP multicast endpoint, also known as a LillTek Messaging cloud. The easiest way to avoid this is by creating a local configuration override file. This file specifies the set of configuration settings that will supersede all settings loaded from application configuration files.

This file is formatted as a standard configuration file. Create the file and then add the **LillTek.ConfigOverride** environment variable, setting its value to the fully qualified path to the override file. Note that this should be a **­system wide** environment variable that is configured for all accounts so that the override settings will be loaded properly by Windows service applications.

The most commonly overridden setting is **MsgRouter.CloudEP**. Each developer, tester, and test lab should be configured with a unique messaging cloud endpoint. Contact [jeff@lilltek.com](mailto:jeff@lilltek.com) for this endpoint. Diagnostic settings are also commonly overridden.

Example Steps:

1. Create the file: **C:\ConfigOverride.txt** with the following contents:

// This file is referenced by the LillTek.ConfigOverride environment variable

// and specifies configuration settings that will override all settings

// loaded by the LillTek.Common.Config class in all applications.

//-----------------------------------------------------------------------------

// Diagnostic settings

#section Diagnostics

-- TraceEnable[0] = 0:DEFAULT

#endsection

//-----------------------------------------------------------------------------

// LillTek Messaging settings

#section MsgRouter

CloudEP = 231.223.000.254:254

#endsection

2. Create the system environment variable:

**LillTek.ConfigOverride=C:\ConfigOverride.txt**