# Overview

Installers are used for deploying software. An installer can be created by several ways as listed below. All these involve an overhead such as installation of special software package to create installers.

1.       Using MSVS .vdproj

2.       WIX toolset

3.       Installshield

EASYInstaller is a standalone application that enables creating installers (.MSI files) from any PC. Some of the advantages are:

§  Standalone application – only .Net 3.5 is required

§  Create layout file containing files, directories and sub directories etc by drag drop from windows explorer

§  Multiple destination folders and installation parameters can be defined

§  Inputs for app.config of deployed assemblies can be provided.

§  Custom file operations can be defined.

§  Pre and Post deployment operations supported.

§  Creates Installers (.MSI files) that support silent installation.

§  Command line operation is supported for build scripts

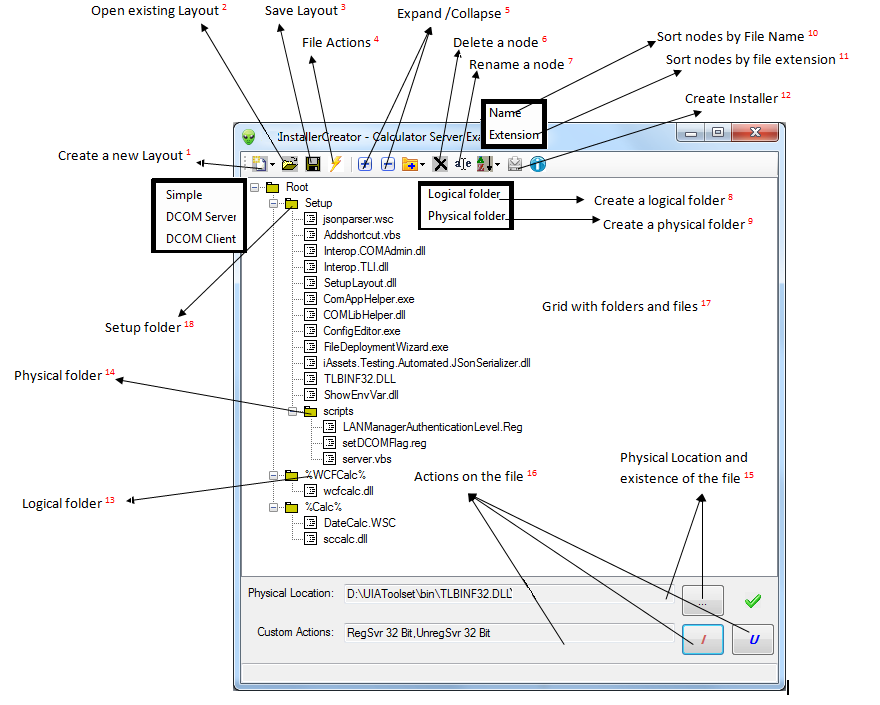
One of the primary goals is to define a file layout for deployment. Layouts are essentially hierarchical file structures with folders and files.

Each file can be associated with one or more file commands. Layouts are also associated with pre and post installation scripts. The name, version for the installer can also be defined.

Each layout results as an installer.  A new package code, product code, and upgrade code are applied when the layout is created. Layouts are stored as files with .layout extension in JSON format.

# User Interface

As discussed earlier, one of the primary goals of the user interface is to define a file layout for deployment. Following describes this in detail.



Create a new Layout 1

This creates a new .layout file. Files and folders can be added at root by drag and drop from windows explorer. Three options are available.

Simple – Creates a basic installer.

DCOM Server – Facilitates installing COM components in COM+ application also provides server side DCOM patching. Users can customize by editing server.vbs.

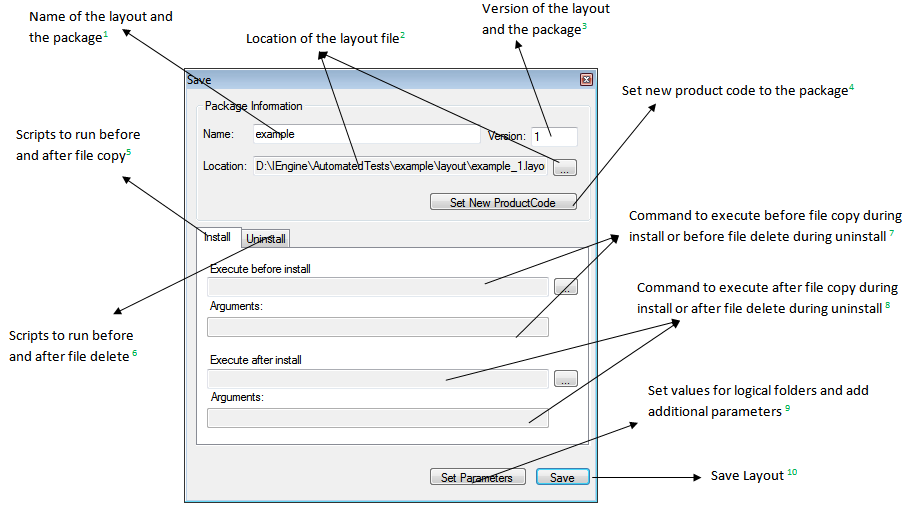
 DCOM Client - Provides client side DCOM patching. Users can customize by editing client.vbs.

Open existing Layout 2

This opens an existing layout. The tree view is populated with the file structure.

Save Layout 3

After files are added to a new layout, the layout needs to be saved to create the installer. Note that a new package code is created for every time a layout is saved. Following describes the user interface in detail.



Name of the layout and the package1

Location of the layout file2

Version of the layout and the package3

The name and the version of the layout file. This will be embedded into the package.  The location of the layout file specifics the directory.

Set new product code to the package 4

It lets a new product code to be defined for the package. This will allow creation of a new product in the families of products. The upgrade code will remain same.

Scripts to run before and after the file copy 5

Users can configure operations to be executed before and after file copy of an installation.

Scripts to run before and after file copy 6

Users can configure operations to be executed before and after file delete of an uninstallation.

Command to execute before file copy during install or before file delete during uninstall 7

These commands typically do checks to be performed before install or uninstall. Upon exceptions, StopInstall environment variable must be set to 1 to abort.

Command to execute after file copy during install or after file delete during uninstall 8

These commands typically do post install and uninstall operation. Upon exceptions, StopInstall environment variable must be set to 1 to rollback.

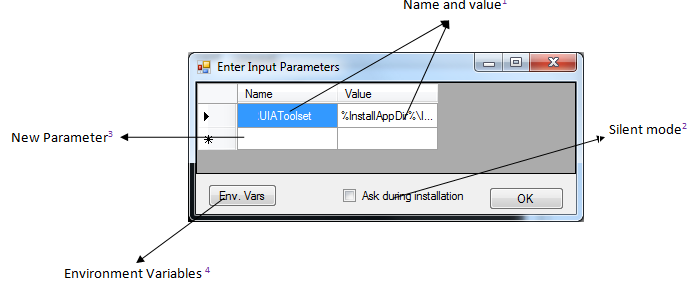
Save Layout 10

This saves the current layout and settings to disk. The new layout will be associated with new package code, product code, and upgrade codes. The file name will be name+version.layout. Example, example\_1.layout.

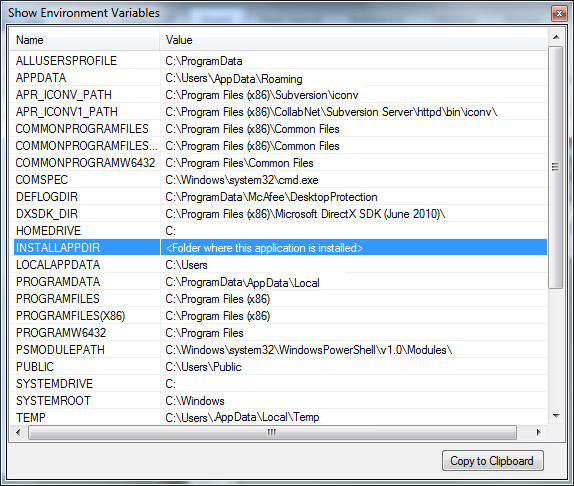
The pre post commands for install and uninstall commands are also stored. By default, the only files specified in the commands are added to the setup folder discussed later. Additional dependencies such as assemblies must be added to setup folder by drag drop from explorer.

Set values for logical folders and add additional parameters 9

When defining a layout, logical folders can be defined whose path is inputted during installation. However, if the path is already known it can be entered ash shown below. All the logical folders defined in the layout are listed. If the input is already defined, these appear as default values during installation. The values can be entered in the second column1. Also new parameters can also be defined by entering the name and value 3. For silent installation mode, asking user input can be turned off by checking off the ask input option 2.  Note that during installation and uninstallation, an environment variable will be created for each of the parameters.

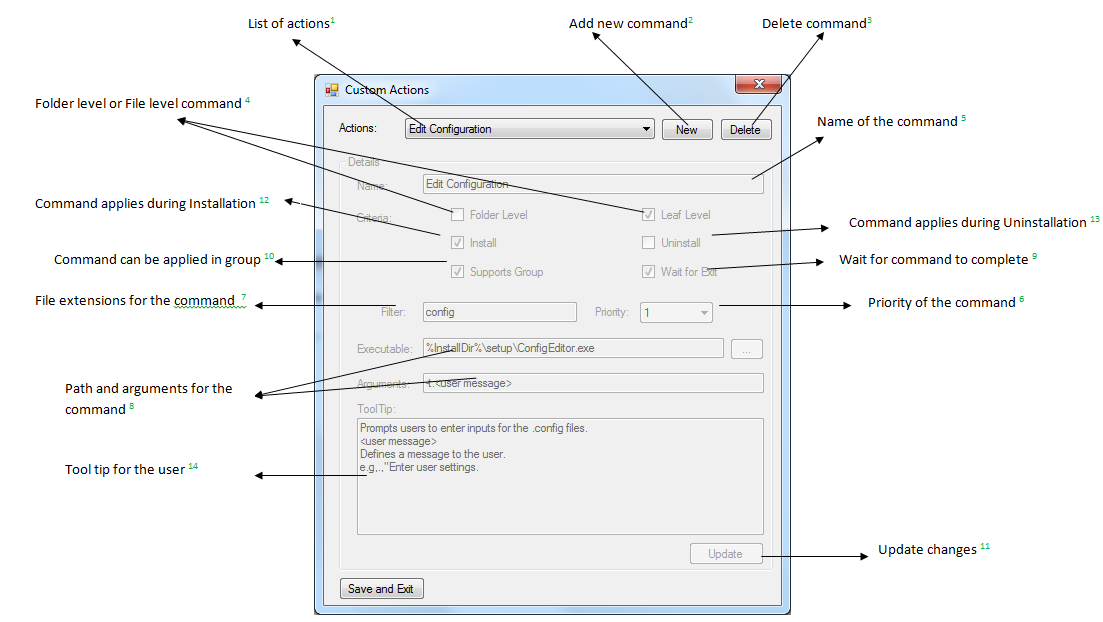
[](https://www.blogger.com/blog/post/edit/5849398651633108988/1231046908734448141)

A list of existing environment variables can be listed. 4



File Actions 4

This dialog enables defining a new custom action or edit an existing. There are 18 predefined commands. Refer to the appendix for the details. These commands cannot be edited or deleted.  New user defined commands can be defined that can also be deleted. During installation these commands are executed after the files are copied. During uninstallation, the commands are executed before deleting the files.



List of actions 1

This is a list of all actions including built in and user defined commands.

Folder level or File level command 4

This indicates if a command is applicable only to a folder or only a file or both to file and folder. For example, regasm is applicable to a file. Whereas create new is applicable to a folder.

Name of the command 5

This indicates the name of the command. The name cannot be changed once the command created. It can be specified only when a new command is created.

Priority of the command 6

This indicates the priority of the command. 0 is highest and 5 is the least. Commands with highest priority are run first.

Applicable File extensions for the command 7

This indicates the file extensions of the file to which this command is applicable. \* indicates for all file extensions. Multiple extension can be applied separated by ;. E.g., dll;exe

Path and arguments for the command 8

The “Executable” indicates the path of the executable with full path. The Arguments indicates the applicable arguments. The installer passes the actual file name as last argument to the command. If Group 10 option is supported, the filenames will be combined with other files executing the same command and the arguments. The arguments within <> needs to be edited.

Wait for command to complete 9

This indicates that the installation or uninstallation waits till the commands are completed.

Command can be applied in group 10

This indicates that this command takes multiple files are arguments. This command is executed it will include all the files that indicate execution of this command and having same command arguments.

Command applies during Installation 12

This indicates that this command is available only for installation.

Command applies during Uninstallation 13

This indicates that this command is available only for Uninstallation.

Tool tip for the user 14

Provides useful information to the user

Update 11

This will commit changes done to the command .

Add new command 2

This will add a new command. Following describe adding a new command notepad.exe.

1.       Click New2  button.

2.       Enter name5 as Notepad

3.       Check Leaf level 4

4.       Enter \* for the filter 7

5.       Select 1 for priority 6

6.       Enter %windir%\notepad.exe for Executable 8

7.       Enter filename(s) for Arguments 8

8.       Leave Wait for Exit 9 and Support group 10 unchecked

9.       Click the Update 11 button

10.   Click Save and Exit button to save the command

Note that all the fields except the name can be edited.

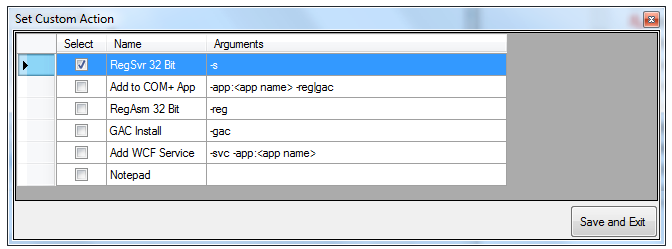
Delete command5

This will delete the selected command.

Actions on the File 16

This sets actions on the individual file and folder items. Installation actions are called after files are copied. Uninstallation actions are called before deleting the files. The physical file path is passed as last argument to the command. Based on the group action setting 6, the action can be combined with other files.  Arguments with <> require user inputs. For example, <app name> represents the name of the COM+ App.

Refer to the appendix for the details.



# Appendix

## Predefined Commands

Note that all command must have filename(s) argument. This will be replaced by actual file name during installation or uninstallation.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Folder** | Leaf | **Executable** | **Arguments** | **Filter** | **Group** | **Priority** | **Install** | **Uninstall** | **Wait** | **Description** |
| Edit Configuration | false | true | ConfigEditor.exe | -t:<user message> | Config | true | 0 | true | false | true | Prompts users to enter inputs for the .config files.  –t defines a message to the user. |
| RegSvr 32 Bit | false | true | Internal | -s | dll,wsc,exe | true | 0 | true | false | true | Registers COM components to 32 bit registry. |
| RegEdit 32 Bit | false | true | Internal | -s | Reg | true | 0 | true | false | true | Update file contents to  32 bit registry. |
| CScript 32 | false | true | Internal |  | .vbs,.js | false | 0 | true | true | true | Executes  the script file |
| Create Folder | true | false |  |  | \* | false | 0 | true | false | true | Creates the folder. |
| Create COM+ App | true | false | ComAppHelper.exe | app:<appname> - -runforever | \* | false | 0 | true | false | true | Create the COM+ application indicated by  <appname>. –runforever indicates that the com+ app runs for ever. |
| Add to COM+ App | false | true | ComAppHelper.exe | -app:<appname> -reg|gac | dll,wsc | false | 1 | true | false | true | Adds the COM component to COM+ application.  <appname> indicates name of the com+ application. reg indicates component is from registry. Gac indicates GAC. |
| RegAsm 32 Bit | false | true | ComAppHelper.exe | -reg | dll,exe | false | 0 | true | false | true | Registers the component information to registry. |
| GAC Install | false | true | ComAppHelper.exe | -gac | dll,exe | false | 0 | true | false | true | Installs the component to GAC. |
| Add WCF Service | false | true | ComAppHelper.exe | -svc -app:<appname> | Dll | true | 1 | true | false | true | Creates a wcf service for the com+ application passed  in <appname> |
| UnregSvr 32 Bit | false | True | Internal | -s -u | dll,wsc,exe | true | 0 | false | true | true | Unregisters the component information to registry. |
| Delete Folder | true | False |  |  | \* | false | 2 | false | true | true | Deletes the folder. |
| Remove COM+ App | false | True | ComAppHelper.exe | -app:<app name> | dll,wsc | false | 1 | false | true | true | Delete the com+ application. |
| UnRegAsm 32 Bit | false | True | ComAppHelper.exe | -unreg | dll,exe | false | 0 | false | true | true | Unregisters the component |
| GAC Uninstall | false | True | ComAppHelper.exe | -ungac | dll,exe | false | 0 | false | true | true | Uninstalls the component from GAC |
| Notepad | false | True | Notepad.exe |  | \* | false | 0 | true | true | false | Opens the file in notepad |
| Add Shortcut | false | True | AddShortcut.vbs | -i | \* | false | 0 | true | false | true | Adds short cu on the current user’s desktop |
| Remove  Shortcut | false | True | AddShortcut.vbs | -u | \* | false | 0 | false | true | true | removes short cu on the current user’s desktop |

## Storyboard: Creating an installer from scratch

The following discusses steps for creating a sample installer that covers all the aspects.

### Overview

The “example” demonstrates using EASYUIAFramework  and EASYUIAToolset to automate calculator (calc.exe) to compute unit conversions. Two installers are created – server and client.

### Server

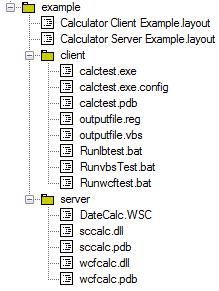
Using EASYUIAFramework ,  a C# based component “sccalc” is created(sccalc.dll). This uses calculator to compute kg to lb conversion. A VBScript based component “DateCalc”  (dateCalc.WSC) is also created to compute date difference. Using Latebinding Code generator, a wrapper class is generated for sccalc and DateCalc components for wcf service (wcfcalc.dll).

### Client

Using Latebinding Code generator, late binding code for managed client is generated. Wcf references for sccalc and datecalc are also added to this client (calctest.exe and calctest.exe.config).  A vbscript client (outputfile.vbs) and registry file for remote generation (outputfile.vbs) are also generated. Command files runlbtest.bat runvbstest.bat and runwcftest.bat are created.

### Deployment

The client and server files are embedded into the EASYUIAToolset\_Win7.layout layout file as example.zip whose structure is shown below.  Once the EASYUIAToolset\_Win7.msi  installer is installed, the next section explains how to create a layout file for server and client. For your reference, these layout files are also included as shown in the layout folder (Calucalator\_Client\_Example.layout and Calucalator \_Server\_Example.layout). After executing the installer EASYUIAToolset\_Win7.msi , the UIA Framework and Toolset are deployed. The examples can be found in C:\Program Files (x86)\UIAToolset\_1.0.0\UIAToolset\Example folder.



### Creating Server Installer

Launch EasyInstallerCreator.exe from  C:\Program Files (x86)\ UIAToolset\_1.0.0\UIAToolset\bin folder.  Execute following steps:

1.       Click new and select DCOM Server

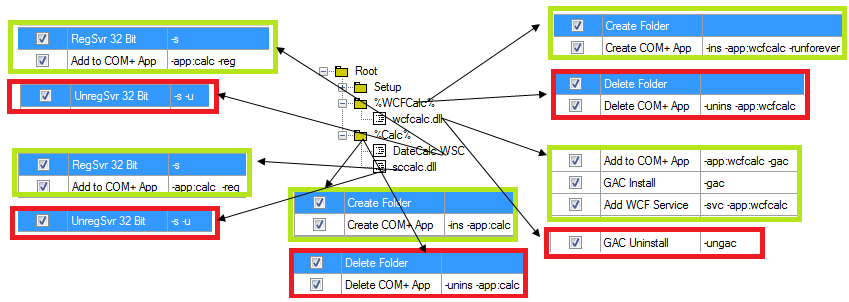
2.       Add a logical folders “WCFCalc. “

3.       Rename “%Components%” to “%Calc%”

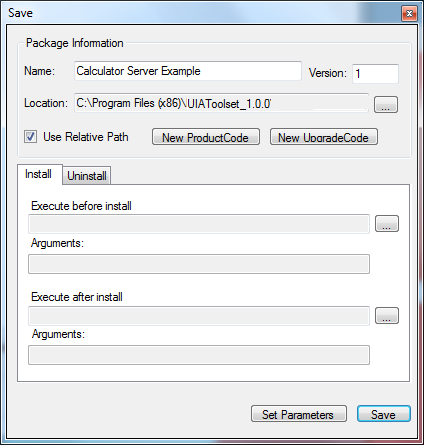
4.       Add a physical folder “server “ under example.

5.       Drag and drop files from C:\Program Files (x86)\ UIAToolset\_1.0.0\UIAToolset\Example\Server

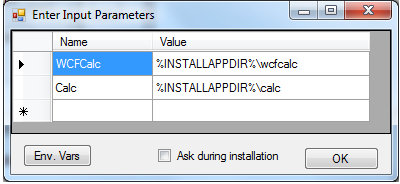
6.       Define custom actions as below.



Save the layout file as below.



Click Set parameters and Save as below:



1.       Create the installer

### Creating Client Installer

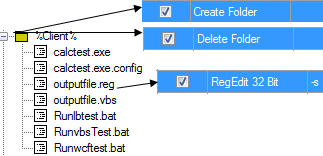
Launch EasyInstallerCreator.exe from C:\Program Files (x86)\UIAToolset\_1.0.0\UIAToolset\bin folder.  Execute following steps:

1.       Click new  and select DCOM client

2.       Add a logical folder “Client “

3.       Drag and drop files from C:\Program Files (x86)\UIAToolset\_1.0.0\UIAToolset\Example\Client

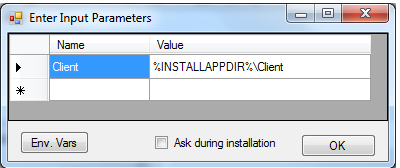
4.       Define custom actions as below.



5.       Save the layout file and define a after install task as below.



Click on “setParameters” button and save as below.



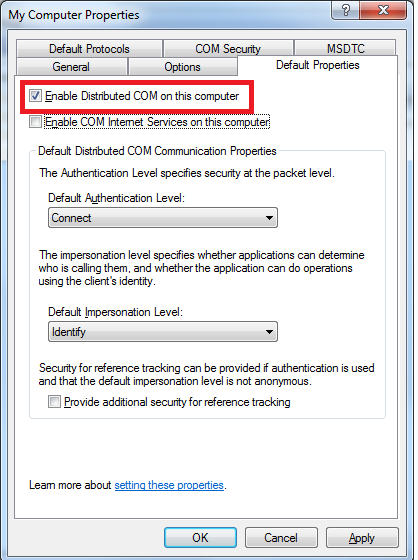
6.       Create the installer

## Setup Checklist

This discusses verification after installation in TAF, XV and Simulator PCs.

### Client PC

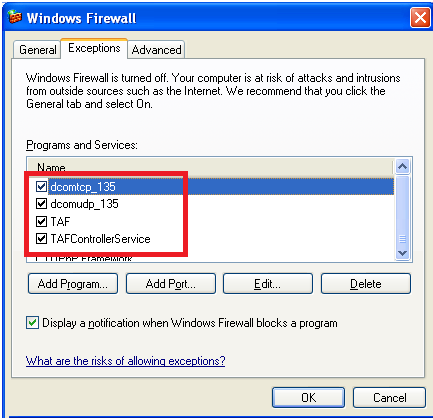
1.       Enable DCOM



2.       FSE user



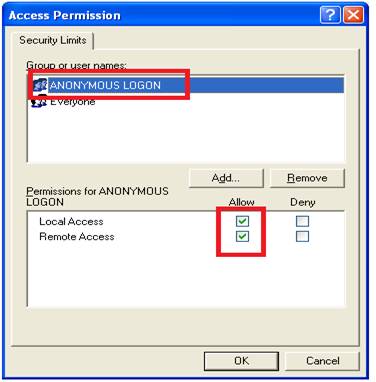
3.       Firewall Exceptions



4.       LAN  Manager Authentication

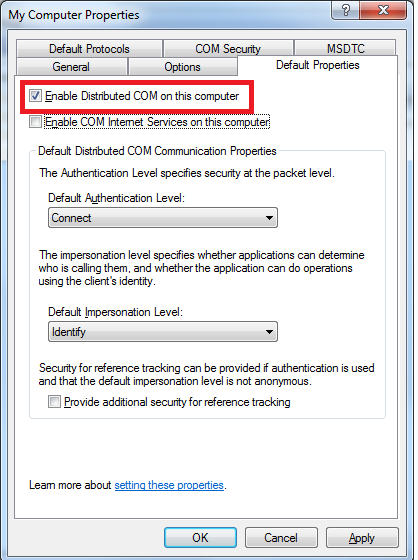


5.       Remote Access



### Server PC

1.       Enable DCOM



2.       FSE user



3.       LAN  Manager Authentication



4.       Firewall configuration

