Sparx Systems Enterprise Architect as an unattended Windows Service on Windows Server 2008R2 and higher

VERSION 1.0

LieberLieber Software GmbH
www.lieberlieber.com

INHALT

SPARX SYSTEMS ENTERPRISE ARCHITECT AS AN UNATTENDED WINDOWS SERVICE ON WINDOWS SERVER 2008R2 AND HIGHER

Author: Peter Lieber, August 2012

EXECUTIVE SUMMARY

This whitepaper describes how to run Enterprise Architect as an **unattended** Windows Service on Windows Server 2008R2 and higher. Before that Server Version it was much easier to run any automation interface based application on a Windows Server environment – but at the end running such applications that are not designed and developed for servers are not only dangerous for instable usage of that application itself – it is even dangerous for the complete server system that may fail because of such constellations.

If you know that risk and you can handle the consequences, then go on reading this document and handle with care and document carefully what you have done – otherwise hands off.

Read this first: http://support.microsoft.com/kb/257757/EN-US
For Windows Server constellations before Windows Server 2008R2 – just do that:
http://blog.lieberlieber.com/2009/09/16/running-enterprise-architect-on-a-server/

PREPARATIONS

Ok – you have reached this line: this document is a visual guideline and the required environment is (all configurations, paths are related to the default installation):

- Windows Server 2008R2 x32/x64 or higher
  - Application Server Role installed (needed for DCOM/COM+)
  - optional: Web Server Role installed
- Enterprise Architect 8.0 or higher
  - Installed for **all** users
  - with a valid **named user** license
- Optional: Visual Studio 2008 Professional or higher – you need the capability to develop Windows (NT) Services

STEP BY STEP ASSISTANCE

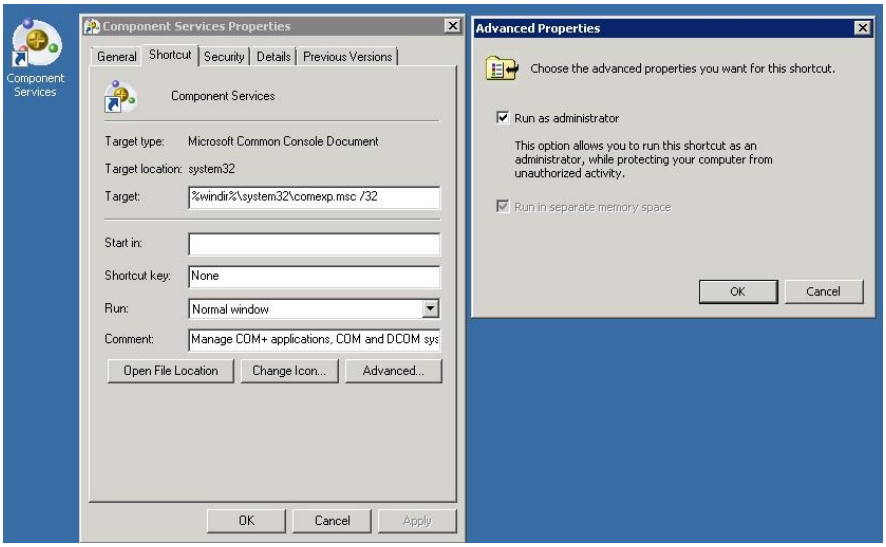
STEP 0: OPERATING SYSTEMS, PATHS AND APPLICATIONS

Generally the following paths are used:

PATH	OS	DESCRIPTION
`C:\Windows\System32`	X32	Hosts any of the OS related applications for 32 bit
`C:\Windows\SysWOW64`	X64	Hosts any of the OS related applications for 64 bit
`C:\Windows\Microsoft.NET\`	Any	Hosts all the .NET related Framework stuff and helpers of the .NET Framework
`C:\Windows\Microsoft.NET\Framework\[Version]` (here we typically use v2.0.50727)	Any	There you can find the `installutil.exe` that we have to use later on

For easier usage create desktop links including administrative permissions to:

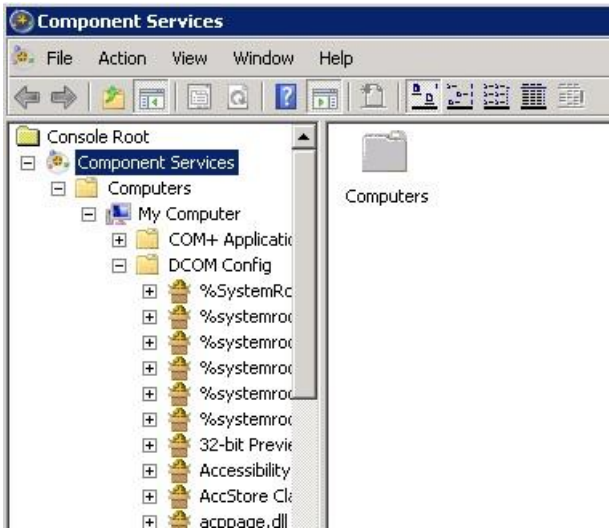
APPLICATION	OS	PATH COMMAND LINE	DESCRIPTION
Component Services	X32 X64	`comexp.exe` `comexp.exe /32`	We need the 32 bit Version of the Component Services, because the COM Interface of EA is 32 bit
Command Line	Any	`cmd.exe`	
.NET Windows (NT) Service Install Util	.NET	`installutil.exe`	You can use any `installutil.exe` of any .NET framework version



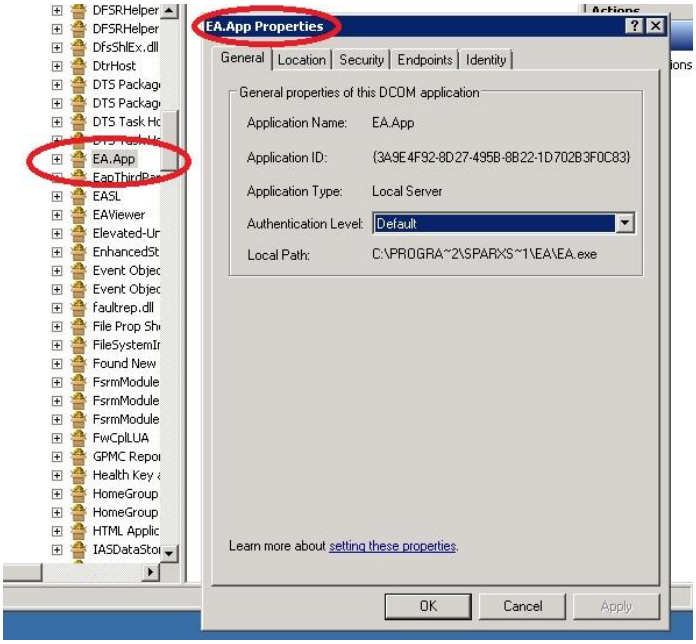
Shortcut on desktop for using comexp with administrative rights (Advanced…|Advanced Properties|Run as administrator).

STEP 1: CONFIGURATION OF DCOM/COM+ FOR EA.APP

Start compexp.exe and open the treeview to DCOM Config.



Then search for **EA.App** and open the properties dialog (right mouse button|properties)

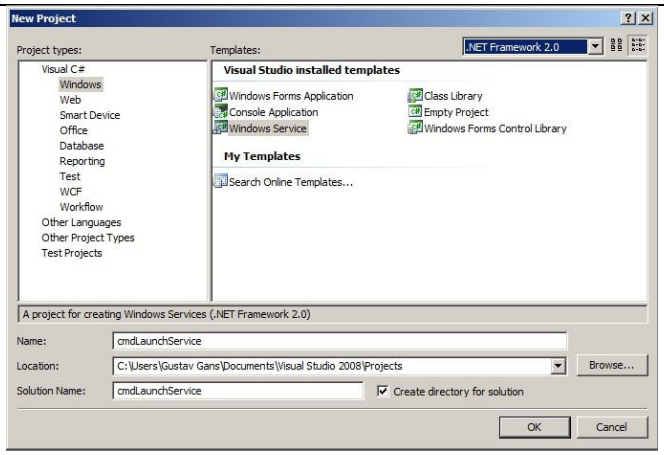
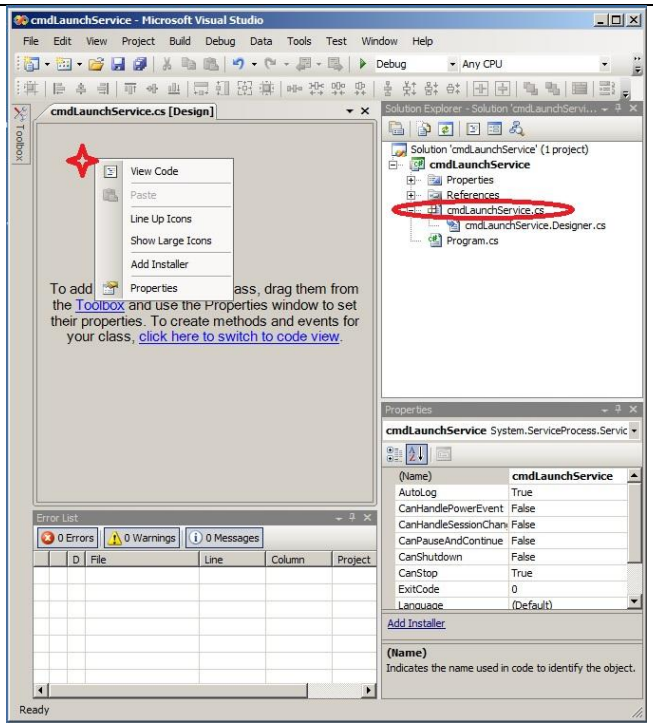


By default you can access EA already (if you change the **identity** to **interactive user**), but you must be logged on (since Windows Server 2008R2) and this is not really what we want for a server environment – to achieve an **unattended** server environment is what we want.

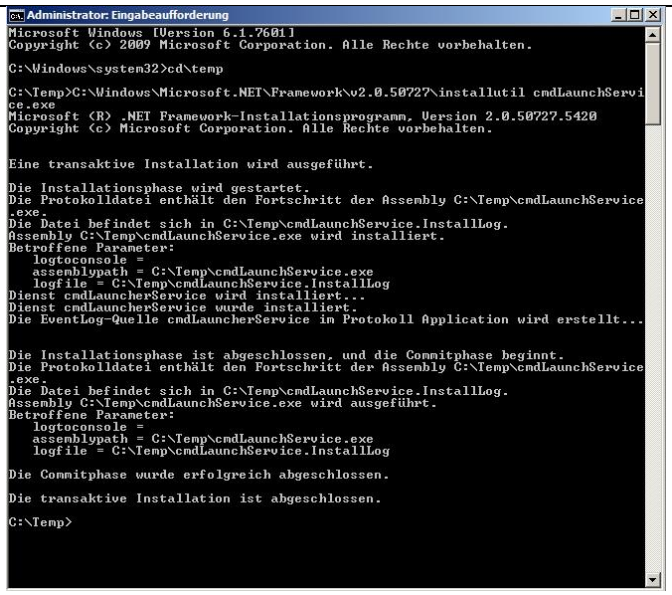
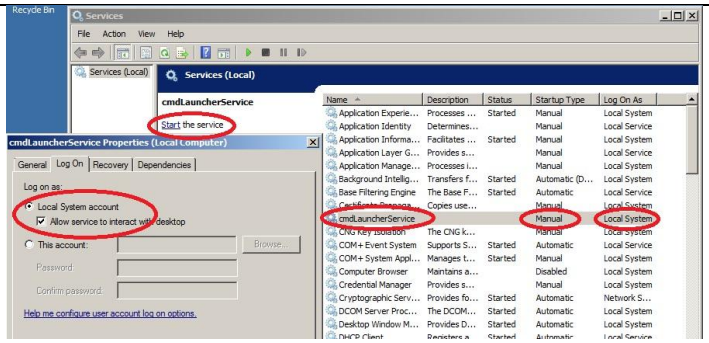
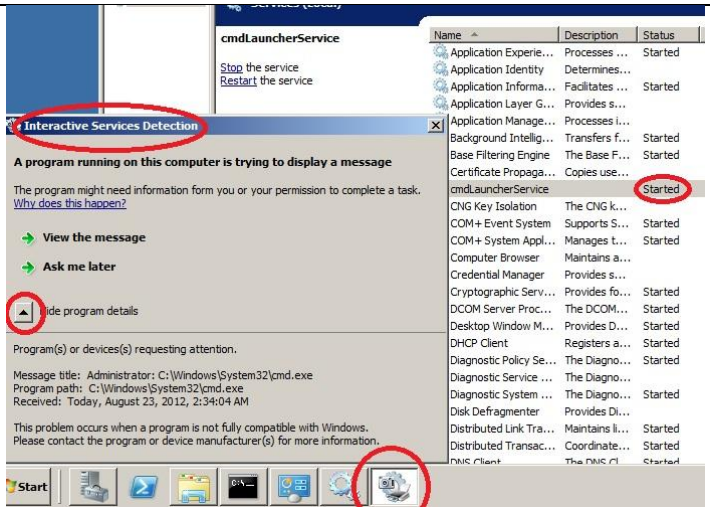
In this case we don't have to change anything, because we are going to run EA in a Local System context and this is the highest security level – even higher than administrator, but you should know – where to configure the access/launching rights.

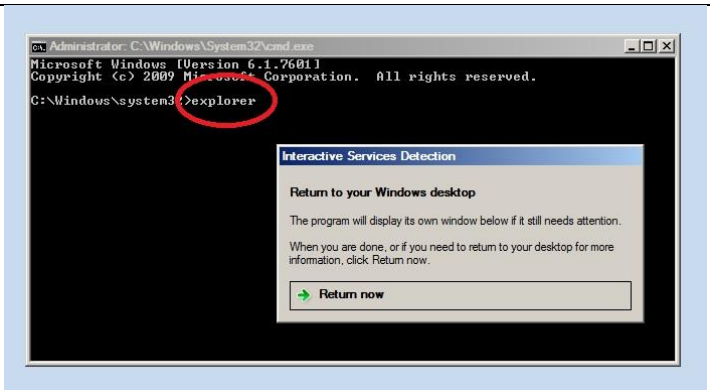
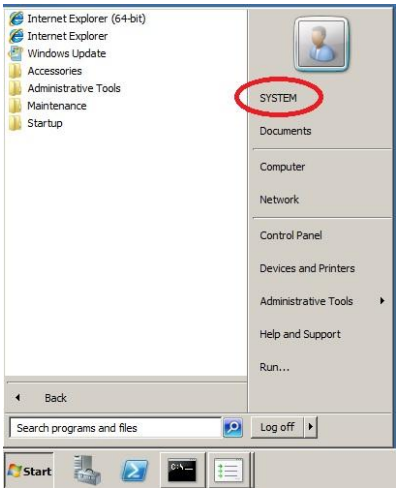
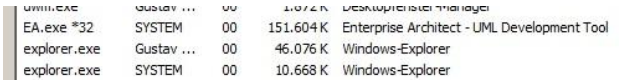
STEP 2: CREATE A WINDOWS SERVICE WITH VISUAL STUDIO

Please read very carefully and don't leave a step out. If you don't want to touch Visual Studio – we can provide the `cmdLaunchService.exe` for any kind of donation – just send an email to: sales@lieberlieber.com.

DESCRIPTION	SCREENSHOT
Launch Visual Studio to create a new windows service	
• File|New|Project… - opens the create new project dialog • Choose C#|Windows|Windows Service – to create a windows service • Choose the .NET Framework (here we take 2.0 because it is the minimum requirement) • Enter cmdLaunchService as name • OK	
• Then rename `Service1.cs` to `cmdLaunchService.cs` (don't forget the .cs) • accept that VS will rename all relations to that name. • Right mouse button in the Design Area of the `cmdLauncService.cs` (marked with a red cross) • Add Installer • Now you have a `ProjectInstaller.cs` • **Set the property "Account" of serviceProcessInstaller to "LocalSystem" (this is the security context of the Service)** • Set the property "ServiceName" of serviceInstaller to "cmdLauncherService"	

• Open the code file of the `cmdLaunchService.cs` (this code will execute on Service's startup using the given security context of the Service)	`protected override void OnStart(string[] args)`
	`{`
	`  Process cmdTool = new Process();`
	`  cmdTool.StartInfo.FileName = "cmd.exe";`
	`  cmdTool.Start();`
	`}`

• Build solution – by default you will find the exe in the following path: `C:\Users\[User]\-Documents\Visual Studio 2008\Projects-\cmdLaunchService-\cmdLaunchService-\bin\Debug` • Copy the `cmdLaunchService.exe` to the Server (here: `c:\temp`) • Open `CMD.EXE` (of course with administrative rights) • Goto the path using `CD c:\temp` • Run `[.NET Framework Path]\Installutil cmdLaunchService.exe`	
• Goto `Services` • Set Properties of `cmdLaunchService` to allow the service to be interactive	
• Start `cmdLaunchService` • Accept the Windows Warning (View the message), that there is an interactive command • Now you have a `CMD` running in Local System security context	

• To get the complete explorer for Local System – you just type in the `CMD` Window: `explorer` • Verify if the explorer runs in Local System context	 
• Launch Enterprise Architect and enter the license key of a named user (that license key allows to use EA in that context) • Go through the wizards (Interface customization wizard) of EA (whatever you might want as primary configuration is ok)	Windows Task Manager shows the security context of EA: 
• Close `EA`, `CMD` • Quit the `cmdLauncherService` in Services	

STEP 2: A QUICK AND DIRTY VERIFICATION

EA is running now in the Local System context. To access EA you need an environment that will also run under Local System – one possibility is to develop a classic Windows Service – hey we have one: For that story we can reuse our `cmdLauncherService` and change the `OnStart` behaviour to do something with EA.

For demonstrating that EA is working just reference the `EA.App` (either via COM or using `EA.tlb` Import or provided by Sparx Systems you can also use `Interop.EA.dll`) and use that sample code:

```
// instance of EA and a repository to get the ProjectGUID
EA.App a = new EA.AppClass();
EA.Repository r = a.Repository;

string ProjectGuid;

if (r.OpenFile("c:\\temp\\eaexample.eap"))
{
  ProjectGuid = r.ProjectGUID;
  r.CloseFile();
}

// clean up
GC.Collect();
GC.WaitForPendingFinalizers();

TextWriter tw = new StreamWriter("c:\\temp\\guids.txt");

// write a line of text to the file, just to have a easy to access info
tw.WriteLine(DateTime.Now);
tw.WriteLine(ProjectGuid);
// close the stream
tw.Close();
```

So requisites are to have a `c:\temp` folder and the `eaexample.eap` there.

You must take care in your code, that EA is not accessed by more than one calling instance at the same time. In this sample it is not relevant – but keeps in mind that you have to define/design/develop such a queuing scenario.

Then change the load behaviour of your service in the service configuration to automatic.
To verify a standalone run – enable access to the `c:\temp` (as file share) from another machine and then reboot the server.
After reboot watch the directory `c:\temp` and voilà the `guids.txt` will raise.

STEP 3: DEVELOP A WCF SERVICE TO ACCESS ENTERPRISE ARCHITECT OVER THE WIRE

tbc

STEP 4: CONFIGURING THE WCF SERVICE

tbc

STEP 5: DEVELOPING A WCF CLIENT APPLICATION

tbc

Welcome Enterprise Architect to run on Windows Server 2008R2 as a windows service

END OF DOCUMENT

