Follow this to Create a new Console application that can be ran as an application as well as a service. This keeps us from needing to maintain a service and test app version like some of the other legacy services that have been built in the past.

1. **Create a new Console application utilizing .net framework**
2. **Add References:**

* Available in Assemblies available
  + System.Configuration
  + System.Configuration.Install
  + System.Deployment
  + System.ServiceModel
  + System.ServiceProcess

1. **Add Required Class Objects:**

**Installer.vb**

* Replace contents with following:

Imports System.ServiceProcess

Imports System.Configuration.Install

Imports System.ComponentModel

<RunInstaller(True)>

Public Class Installer

Inherits ServiceProcessInstaller

Private ProcessInstaller As ServiceProcessInstaller

Private ServiceInstaller As ServiceInstaller

Public Sub New()

ProcessInstaller = New ServiceProcessInstaller

ServiceInstaller = New ServiceInstaller

ProcessInstaller.Account = ServiceAccount.LocalSystem

ServiceInstaller.StartType = ServiceStartMode.Manual

ServiceInstaller.ServiceName = "GWC Service - XXX - XXXXXXXXXXXX"

ServiceInstaller.Description = "XXXXXXXXX"

Installers.Add(ServiceInstaller)

Installers.Add(ProcessInstaller)

End Sub

End Class

* Replace / Update the areas where X’s are listed with appropriate name for the service and the Description

1. **Update Module1:**

* Open code of Module1.
* Replace contents with following

Imports System.Threading

Imports System.Data.SqlClient

Imports System.ServiceProcess

Imports System.Configuration.Install

Imports System.ComponentModel

Imports System.Configuration

Imports System.Reflection

Imports System.IO

Imports System.Configuration.ConfigurationManager

Imports Microsoft.VisualBasic.ApplicationServices

'Namespace to handle the running of app/service

Namespace PIM\_INTEGRATION\_OUT\_SERVICE

Class Program

#Region "Nested classes To support running As service"

'Name of the service

Public Const ServiceName As String = ""

Class Service

Inherits ServiceBase

Public Sub New()

ServiceName = Program.ServiceName

End Sub

Protected Overrides Sub OnStart(args As String())

Program.Start(args)

End Sub

Protected Overrides Sub OnStop()

Program.[Stop]()

End Sub

Protected Overrides Sub OnShutdown()

Program.[Shutdown]()

End Sub

End Class

'The startup procedure that is called in the project.

'This will identify if being ran as service or in userInteractive mode which displays the console.

Public Shared Sub Main(ByVal args As String())

If Not Environment.UserInteractive Then

Using service = New Service()

ServiceBase.Run(service)

End Using

Else

'will run as console application and will run until any key is pressed in the console

Module1.DisplayInConsole = True

Module1.StartService()

Console.WriteLine("Press any key to stop...")

Console.ReadKey(True)

[Stop]()

End If

End Sub

'Handles Start command of service

Private Shared Sub Start(ByVal args As String())

Module1.StartService()

End Sub

'Handles Stop command of Service

Private Shared Sub [Stop]()

Module1.StopService()

End Sub

'Handles shutdown of machine while service is executing

Private Shared Sub [Shutdown]()

Module1.StopService(ShutdownFlag:=True)

End Sub

#End Region

End Class

End Namespace

Module Module1

Public DisplayInConsole As Boolean = False

Private WithEvents tmrProcess As New Timers.Timer

Private ProcessThread As Thread

Private Sleeping As Boolean = False

Public StopProcessing As Boolean = False

'Need set based on settings

Public ProcessStartHour As Integer = 5 '5AM

Public ProcessEndHour As Integer = 10 '10PM

Public Process\_DefaultSleepTimer As Integer = 30000 '5 Minutes

'Events that are structured and can be called during logging

Structure Events

Const ServiceStart = "Service Start"

Const ServiceStop = "Service Stop"

Const ServiceError = "Service Error"

Const ServiceSleep = "Service Sleeping"

End Structure

Sub Main()

End Sub

'procedure to start the service. This will call procedure to load application settings.

'Once settings are loaded, timer is set to sleep for 5000 (5 seconds).

'This will allow the service to show as running.

'The processing will then be handled in the “Timer\_Elapsed” procedure

Public Sub StartService()

ProcessThread = Thread.CurrentThread

Dim c As Int16 = 0

'Load Application Settings. If fails the first time, try up to 3 times before erroring out.

LoadApplicationSettings:

Try

AddLog("", "", Events.ServiceStart, "DPS Service Starting", UpdateStatus:=True)

LoadApplicationSettings()

c = 0

Catch ex As Exception

AddLog("", "", Events.ServiceError, "Exception Starting Service: " & ex.Message, UpdateStatus:=True)

c += 1

Finally

If c >= 1 And c < 3 Then

Threading.Thread.Sleep(5000)

End If

End Try

If c >= 1 And c < 3 Then

GoTo LoadApplicationSettings

ElseIf c >= 3 Then

End

End If

Try

AddLog("", "", Events.ServiceStart, "XXXXX Service Started Timer Enabling 5 Second Sleep", UpdateStatus:=True)

tmrProcess.Interval = 5000

tmrProcess.Enabled = True

Catch ex As Exception

AddLog("", "", Events.ServiceError, "XXXXX Service Error: " & ex.Message)

commonServices.SendErrorEmail(Nothing, "Process Record Exception: " & ex.Message)

EndService("EXCEPTION: " & ex.Message)

End Try

End Sub

'Procedure to stop the actual processing taking place.

'If timer is currently marked as sleeping, then make call to end service

'If timer is not sleeping, the app is processing, then give the app 15 seconds to stop the processing before forcing the shutdown.

Public Sub StopService(Optional ShutdownFlag As Boolean = False)

If Sleeping Then

EndService("Requested Stop (Status: Process Sleeping)")

Else

StopProcessing = True

tmrProcess.Stop()

tmrProcess.Dispose()

'ProcessThread.Join()

Dim c As Int16 = 0

Do Until c = 3

If Sleeping Then Exit Do

Threading.Thread.Sleep(5000)

c += 1

Loop

End If

End Sub

'What to do when ending service. This will would generally be to Log the service is stopping and send error email notification

Private Sub EndService(ReasonMsg As String)

AddLog("", "", Events.ServiceStop, "DPS Service Stop Reason: " & ReasonMsg, UpdateStatus:=True)

commonServices.SendErrorEmail(Nothing, "Service Has Stopped. Reason: " & ReasonMsg)

End Sub

'Procedure to load any application settings as necessary

Private Sub LoadApplicationSettings()

End Sub

'What to do when the timer has elapsed. In this one RunProcessLoop is called where the processing happens.

'Once processing done, check time of day the service is executing and if within the window of execution, set sleep timer for default sleep amount.

'If time is outside processing window, set the timer to sleep until the start of the execution window on the next day.

Private Sub tmrProcess\_Elapsed(Sender As Object, e As Timers.ElapsedEventArgs) Handles tmrProcess.Elapsed

Dim DefaultSleepTime As Integer = Process\_DefaultSleepTimer

Dim SleepTime As Integer = DefaultSleepTime

Sleeping = False

Try

tmrProcess.Enabled = False

RunProcessLoop()

If Not StopProcessing Then

If DateTime.Now.Hour >= ProcessStartHour And DateTime.Now.Hour < ProcessEndHour Then

SleepTime = DefaultSleepTime

Else

Dim NextRunDate As DateTime = If(DateTime.Now.Hour <= 4, Today(), DateAdd(DateInterval.Day, 1, Today())).ToShortDateString + " " & ProcessStartHour.ToString & ":05 AM"

SleepTime = DateDiff(DateInterval.Minute, Now(), NextRunDate) \* 60000

End If

End If

Catch ex As Exception

SleepTime = Process\_DefaultSleepTimer

Finally

If Not StopProcessing Then

Sleeping = True

AddLog("", "", Events.ServiceSleep, "DPS Service Sleep: " & (SleepTime / 1000).ToString & " seconds", UpdateStatus:=True)

tmrProcess.Interval = SleepTime

tmrProcess.Enabled = True

Else

EndService("Requested Stop (Status: Actively Processing)")

Sleeping = True

End If

End Try

End Sub

'Procedure to display in console. Instead of Console.WriteLine each time, just call this procedure and it will handle the same things but not put anything in console if running as service.

Public Sub DisplayStatus(StatusMsg As String)

If DisplayInConsole Then Console.WriteLine(Now().ToString & " " & StatusMsg)

End Sub

'Procedure to add log using the ApplicaitonStatus Logging in SysproDocument Tables

Public Sub AddLog(GroupingID As String, ID As String, EventDesc As String, LogValue As String, Optional LogValue2 As String = "", Optional XMLValue As String = "", Optional UpdateStatus As Boolean = False, Optional EndApplication As Boolean = False)

Try

DisplayStatus(IIf(LogValue = "" AndAlso XMLValue <> "", XMLValue, LogValue))

If XMLValue <> "" Then

Try

Dim Xdoc As XDocument = XDocument.Parse(XMLValue)

XMLValue = Xdoc.ToString

Catch ex As Exception

If LogValue = "" Then

LogValue = XMLValue

Else

LogValue2 = XMLValue

End If

End Try

End If

dataAccess.AddApplicationStatusLog(GroupingID, ID, EventDesc, LogValue, LogValue2, XMLValue, UpdateStatus)

Catch ex As Exception

If EndApplication Then

StopProcessing = True

StopService()

End If

End Try

End Sub

End Module

This code will need to be modified as you see fit in regards to the highlighted areas and any other areas you see fit. Here are comments on

This will get you started on being able to use the console application as an application as well as a service.

* When a user double clicks on the exe, it will run as console application.
* When it is installed through a service installation, it will run as a service (without the console screen displayed).
* This will also hand the start, stop, shutdown of server as needed.

1. **Update Application Startup Object:**

* Right-Click on the project in Solution Explorer and choose properties
* Click on Application on left hand side
* In the Startup object: choose the one that matches the Namespace.Program that is listed in the Module1. In the version copied above, it would be “PIM\_INTEGRATION\_OUT\_SERVICE.Program

Graphical user interface, text, application

Description automatically generated

1. **Optional Class Objects:**

**DataContext.vb**

This will be where all the different calls to any stored procedures or anything in the database will come from.

**XXX\_Objects.vb**

This could be where different class object definitions / structures are placed to be referenced.