Contents

[Requirements: 2](#_Toc318528665)

[ClassSCD 2](#_Toc318528666)

[InterServ 10](#_Toc318528667)

[WindowsService 24](#_Toc318528668)

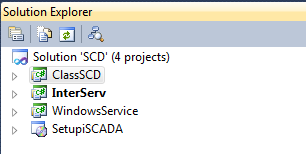
[SetupiSCADA 28](#_Toc318528669)

# Requirements:

Build on : MS Visual Studio 2010 C#

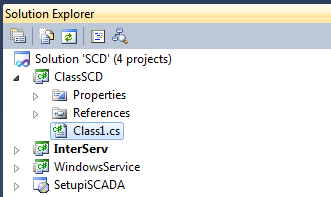
Microsoft .Net Framework 4

This source code divided into 4 projects:



# ClassSCD

The ClassSCD, this class as the global parameter class for security, configuration, connection and log history



Class1.cs

using System;

using System.Collections.Generic;

using System.Configuration;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Data.ProviderBase;

using System.IO;

using System.Runtime.InteropServices;

using System.Security.Cryptography;

using System.Text;

using System.Windows.Forms;

namespace ClassSCD

{

public class Class1

{

public string fn = "\\SetTimer.txt", fE = "\\EAMSet.csi", fS = "\\SCDSet.csi", fL = "\\SCADA\_EAM\_LOG.txt", aS = Application.StartupPath.ToString();

public string Encrypt(string toEncrypt, bool useHashing)

{

byte[] keyArray;

byte[] toEncryptArray = UTF8Encoding.UTF8.GetBytes(toEncrypt);

System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();

// Get the key from config file

string key = "GG";

//string key = (string)settingsReader.GetValue("SecurityKey", typeof(String));

//System.Windows.Forms.MessageBox.Show(key);

//If hashing use get hashcode regards to your key

if (useHashing)

{

MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();

keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));

//Always release the resources and flush data of the Cryptographic service provide. Best Practice

hashmd5.Clear();

}

else

keyArray = UTF8Encoding.UTF8.GetBytes(key);

TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();

//set the secret key for the tripleDES algorithm

tdes.Key = keyArray;

//mode of operation. there are other 4 modes. We choose ECB(Electronic code Book)

tdes.Mode = CipherMode.ECB;

//padding mode(if any extra byte added)

tdes.Padding = PaddingMode.PKCS7;

ICryptoTransform cTransform = tdes.CreateEncryptor();

//transform the specified region of bytes array to resultArray

byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);

//Release resources held by TripleDes Encryptor

tdes.Clear();

//Return the encrypted data into unreadable string format

return Convert.ToBase64String(resultArray, 0, resultArray.Length);

}

public string Decrypt(string cipherString, bool useHashing)

{

byte[] keyArray;

//get the byte code of the string

byte[] toEncryptArray = Convert.FromBase64String(cipherString);

System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();

//Get your key from config file to open the lock!

// string key = (string)settingsReader.GetValue("SecurityKey", typeof(String));

string key = "GG";

if (useHashing)

{

//if hashing was used get the hash code with regards to your key

MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();

keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));

//release any resource held by the MD5CryptoServiceProvider

hashmd5.Clear();

}

else

{

//if hashing was not implemented get the byte code of the key

keyArray = UTF8Encoding.UTF8.GetBytes(key);

}

TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();

//set the secret key for the tripleDES algorithm

tdes.Key = keyArray;

//mode of operation. there are other 4 modes. We choose ECB(Electronic code Book)

tdes.Mode = CipherMode.ECB;

//padding mode(if any extra byte added)

tdes.Padding = PaddingMode.PKCS7;

ICryptoTransform cTransform = tdes.CreateDecryptor();

byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);

//Release resources held by TripleDes Encryptor

tdes.Clear();

//return the Clear decrypted TEXT

return UTF8Encoding.UTF8.GetString(resultArray);

}

public string StrVar01(string iEQ, string iEQOrg, string iUOM)

{

string stri = @"SELECT OBJ\_CODE,OBJ\_ORG,OUD\_TOTALUSAGE,OUD\_SINCEINSTALL,OUD\_UOM,OUD\_METER,MET\_ORG FROM R5OBJECTS " +

"LEFT OUTER JOIN R5OBJUSAGEDEFS ON OBJ\_CODE=OUD\_OBJECT " +

"LEFT OUTER JOIN R5METERS ON OUD\_METER = MET\_CODE " +

"WHERE OBJ\_CODE='" + iEQ + "' AND OBJ\_ORG='" + iEQOrg + "'" +

"AND OUD\_UOM='" + iUOM + "'";

return stri;

}

public string StrVar02(string jEQ, string jEQOrg, string jUOM, int jRead, string jDate)

{

string stri = "INSERT INTO R5TRACKINGDATA " +

"(TKD\_TRANS,TKD\_SOURCESYSTEM,TKD\_TRACKDATE,TKD\_CREATED," +

"TKD\_PROMPTDATA2,TKD\_PROMPTDATA3,TKD\_PROMPTDATA1," +

"TKD\_PROMPTDATA4,TKD\_PROMPTDATA8,TKD\_PROMPTDATA5)" +

"VALUES ('M210','METER READ','" + jDate + "','" + DateTime.Now.ToString() + "'," +

"'" + jEQ + "','" + jEQOrg + "','R5'," +

"'Y','" + jRead + "','" + jUOM + "')";

return stri;

}

public string ConnStringEAM()

{

string server, username, password, database, constr;

StreamReader sr = new StreamReader(aS + fE);

server = sr.ReadLine();

username = sr.ReadLine();

password = Decrypt(sr.ReadLine(), true);

database = sr.ReadLine();

sr.Close();

constr = @"Data Source=" + server + ";" +

"Initial Catalog=" + database + ";" +

"User ID=" + username + ";" +

"Password=" + password + ";";

return constr;

}

public string ConnStringSCD()

{

string server, username, password, database, constr;

StreamReader sr = new StreamReader(aS + fS);

server = sr.ReadLine();

username = sr.ReadLine();

password = Decrypt(sr.ReadLine(), true);

database = sr.ReadLine();

sr.Close();

constr = @"Data Source=" + server + ";" +

"Initial Catalog=" + database + ";" +

"User ID=" + username + ";" +

"Password=" + password + ";";

return constr;

}

public int StringToInt(string iSTI)

{

int iI;

if ((iSTI == null) || (iSTI == ""))

{

iI = 0;

}

else

{

if (iSTI.Contains(","))

{

iSTI = iSTI.Replace(",", "");

}

if (iSTI.Contains("."))

{

iSTI = iSTI.Substring(0, iSTI.IndexOf(".", 0));

}

iI = Convert.ToInt32(iSTI);

}

return iI;

}

public Boolean CekValidServer(string xConnSvr)

{

SqlConnection sqlconn;

string strConn;

strConn = xConnSvr;

try

{

sqlconn = new SqlConnection();

sqlconn.ConnectionString = strConn;

sqlconn.Open();

sqlconn = null;

return true;

}

catch (Exception ex)

{

WriteLogHis(DateTime.Now.ToString(), ex.Message.ToString()+" CONNECTION FAILED");

return false;

}

}

public Boolean FileCheckExist(string iPath)

{

if (File.Exists(iPath))

{

return true;

}

else

{

return false;

}

}

public DataSet GetData(string iStrSQL, string iConn)

{

DataSet DS = new DataSet();

SqlConnection SqlConn = new SqlConnection(iConn);

try

{

SqlConn.Open();

SqlDataAdapter DA = new SqlDataAdapter(iStrSQL, SqlConn);

DA.Fill(DS);

SqlConn.Close();

return DS;

}

catch (Exception Ex)

{

throw Ex;

}

finally

{

if (SqlConn.State == ConnectionState.Open)

{

SqlConn.Close();

}

}

}

public void WriteLogToDB(int wID, string wEQOrg, string wEQ, string wTime, int wSeq, int wRead, string wUOM, string wMessage)

{

string wStr = "INSERT INTO TbLogHis (LogDate,SCD\_ID,SCD\_ORG,SCD\_UNIT,SCD\_TIME,SCD\_SEQ,SCD\_READ,SCD\_UOM,SCD\_MESSAGE) VALUES "+

"('" +DateTime.Now.ToString() + "'," + wID + ",'" + wEQOrg + "','" + wEQ + "','" + wTime + "'," + wSeq + "," + wRead + ","+

"'" + wUOM + "','" + wMessage + "')";

try

{

ExecuteQuery(wStr, ConnStringSCD());

}

catch (Exception ex)

{

//MessageBox.Show(ex.Message);

WriteLogHis(DateTime.Now.ToString(), ex.Message.ToString());

}

}

public void WriteLogHis(string sMessage, string sWriteon)

{

string sPath;

sPath = Application.StartupPath.ToString() + fL;

sMessage = sMessage + ";" + sWriteon;

if (FileCheckExist(sPath))

{

FileStream addFile = new FileStream(sPath, FileMode.Append, FileAccess.Write);

StreamWriter sw = new StreamWriter(addFile);

sw.WriteLine(sMessage);

sw.Close();

addFile.Close();

}

else

{

StreamWriter sw2 = new StreamWriter(sPath);

sw2.WriteLine(sMessage);

sw2.Close();

}

}

public void ExecuteQuery(string eStr, string eConn)

{

SqlConnection SqlConn = new SqlConnection(eConn);

try

{

SqlConn.Open();

SqlCommand sqlCmd = new SqlCommand();

sqlCmd.CommandText = eStr;

sqlCmd.Connection = SqlConn;

sqlCmd.ExecuteNonQuery();

SqlConn.Close();

}

catch (Exception Ex)

{

throw Ex;

}

finally

{

if (SqlConn.State == ConnectionState.Open)

{

SqlConn.Close();

}

}

}

public void ResponToSCD(int rStr, Boolean rT)

{

string sStr;

if (rT == true)

{

sStr = "UPDATE Tbscd SET scd\_flag=1 WHERE scd\_id=" + rStr;

}

else

{

sStr = "UPDATE Tbscd SET scd\_flag=2 WHERE scd\_id=" + rStr;

}

ExecuteQuery(sStr, ConnStringSCD());

}

public void DataCollection()

{

if (FileCheckExist(aS + fn))

{

if (FileCheckExist(aS + fS))

{

if (FileCheckExist(aS + fE))

{

int nou,xs,sUsage=0,sDiff=0,sRea=0;

string zsql, ysql, isql, sEQOrg, sEQ, sTime, sUom;

DataSet dsSCD, dsEAM;

DataTable dtSCD;

try

{

zsql = "SELECT scd\_id,scd\_unit,scd\_org,scd\_time,"+

"scd\_prod1,scd\_prod2,scd\_prod3,scd\_prod4,"+

"scd\_uom1,scd\_uom2,scd\_uom3,scd\_uom4,"+

"scd\_flag FROM Tbscd WHERE scd\_flag=0";

dsSCD = GetData(zsql, ConnStringSCD()); //GET SCADA DATASET

dtSCD = dsSCD.Tables[0];

xs = dtSCD.Rows.Count;

if (xs > 0)

{

for (int i = 0; i < xs ; i++)

{

nou = StringToInt(dtSCD.Rows[i][0].ToString()); //scd\_id

sEQ = dtSCD.Rows[i][1].ToString();

sEQOrg = dtSCD.Rows[i][2].ToString();

sTime = dtSCD.Rows[i][3].ToString();

if ((sEQ.Length > 0) && (sEQOrg.Length > 0) && (sTime.Length > 0))

{

for (int kolR = 4, kolM = 8; kolR < 8; kolR++, kolM++)

{

string pRea = dtSCD.Rows[i][kolR].ToString();

sUom = dtSCD.Rows[i][kolM].ToString();

if ((pRea != "") && (sUom != ""))

{

sRea = StringToInt(pRea);

ysql = StrVar01(sEQ, sEQOrg, sUom);

dsEAM = GetData(ysql, ConnStringEAM());

if (dsEAM.Tables[0].Rows.Count > 0)

{

sUsage = StringToInt(dsEAM.Tables[0].Rows[0]["OUD\_TOTALUSAGE"].ToString());

sDiff = sRea - sUsage;

if (sDiff > 0)

{

isql = StrVar02(sEQ, sEQOrg, sUom, sRea, sTime);

ExecuteQuery(isql, ConnStringEAM());

ResponToSCD(nou, true);

}

else

{

WriteLogToDB(nou, sEQOrg, sEQ, sTime, (kolR - 3), sRea, sUom, "METER READING ROLLBACK");

ResponToSCD(nou, false);

}

}

else

{

WriteLogToDB(nou, sEQOrg, sEQ, sTime, (kolR - 3), sRea, sUom, "EQUIPMENT/UOM NOT REGISTERED");

ResponToSCD(nou, false);

}

}

}

}

else

{

WriteLogToDB(nou, sEQOrg, sEQ, sTime,0, 0, "", "MANDATORY DATA INVALID");

ResponToSCD(nou, false);

}

}

}

}

catch (Exception Ex) { WriteLogToDB(0, "", "", "", 0, 0, "", Ex.ToString()); }

}

else { WriteLogToDB(0, "", "", "", 0, 0, "", "EAM FILE SETTING DOESN'T EXIST"); }

}

else { WriteLogToDB(0, "", "", "", 0, 0, "", "SCADA FILE SETTING DOESN'T EXIST"); }

}

else { WriteLogToDB(0, "", "", "", 0, 0, "", "TIMER FILE SETTING DOESN'T EXIST"); }

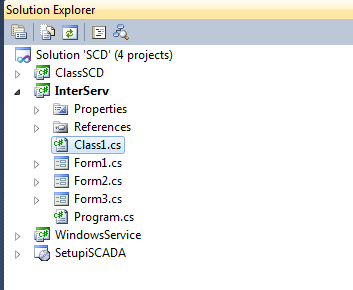
}

}

}

# InterServ

The InterServ is all interface forms for user interaction into SCADA Interface



Class1.cs

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Data.ProviderBase;

using System.IO;

using InterServ;

using System.Runtime.InteropServices;

using System.Security.Cryptography;

using System.Text;

using System.Windows.Forms;

namespace InterServ

{

public class Class1

{

public string Encrypt(string toEncrypt, bool useHashing)

{

byte[] keyArray;

byte[] toEncryptArray = UTF8Encoding.UTF8.GetBytes(toEncrypt);

System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();

// Get the key from config file

string key = "GG";

//string key = (string)settingsReader.GetValue("SecurityKey", typeof(String));

//System.Windows.Forms.MessageBox.Show(key);

//If hashing use get hashcode regards to your key

if (useHashing)

{

MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();

keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));

//Always release the resources and flush data of the Cryptographic service provide. Best Practice

hashmd5.Clear();

}

else

{

keyArray = UTF8Encoding.UTF8.GetBytes(key);

}

TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();

//set the secret key for the tripleDES algorithm

tdes.Key = keyArray;

//mode of operation. there are other 4 modes. We choose ECB(Electronic code Book)

tdes.Mode = CipherMode.ECB;

//padding mode(if any extra byte added)

tdes.Padding = PaddingMode.PKCS7;

ICryptoTransform cTransform = tdes.CreateEncryptor();

//transform the specified region of bytes array to resultArray

byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);

//Release resources held by TripleDes Encryptor

tdes.Clear();

//Return the encrypted data into unreadable string format

return Convert.ToBase64String(resultArray, 0, resultArray.Length);

}

public string Decrypt(string cipherString, bool useHashing)

{

byte[] keyArray;

//get the byte code of the string

byte[] toEncryptArray = Convert.FromBase64String(cipherString);

System.Configuration.AppSettingsReader settingsReader = new AppSettingsReader();

//Get your key from config file to open the lock!

// string key = (string)settingsReader.GetValue("SecurityKey", typeof(String));

string key = "GG";

if (useHashing)

{

//if hashing was used get the hash code with regards to your key

MD5CryptoServiceProvider hashmd5 = new MD5CryptoServiceProvider();

keyArray = hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));

//release any resource held by the MD5CryptoServiceProvider

hashmd5.Clear();

}

else

{

//if hashing was not implemented get the byte code of the key

keyArray = UTF8Encoding.UTF8.GetBytes(key);

}

TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider();

//set the secret key for the tripleDES algorithm

tdes.Key = keyArray;

//mode of operation. there are other 4 modes. We choose ECB(Electronic code Book)

tdes.Mode = CipherMode.ECB;

//padding mode(if any extra byte added)

tdes.Padding = PaddingMode.PKCS7;

ICryptoTransform cTransform = tdes.CreateDecryptor();

byte[] resultArray = cTransform.TransformFinalBlock(toEncryptArray, 0, toEncryptArray.Length);

//Release resources held by TripleDes Encryptor

tdes.Clear();

//return the Clear decrypted TEXT

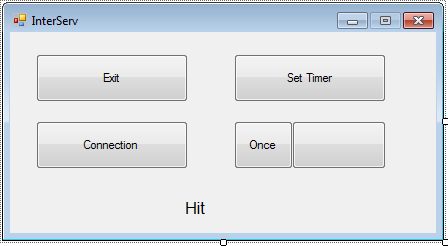
return UTF8Encoding.UTF8.GetString(resultArray);

}

}

}

Form1.cs



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.IO;

using System.ServiceProcess;

using System.Text;

using System.Windows.Forms;

using Microsoft.CSharp;

using Microsoft.SqlServer.Server;

using ClassSCD;

namespace InterServ

{

public partial class Form1 : Form

{

public int iwaktu;

//public string fn = "\\SetTimer.txt";

ClassSCD.Class1 tClass = new ClassSCD.Class1();

public Form1()

{

InitializeComponent();

label1.Width = 150;

label1.Height = 25;

RefTimer();

timer1.Tick += new EventHandler(timer1\_tick); // Everytime timer elapse, timer\_tick will be called

timer1.Interval = (iwaktu \* 1000); // Timer will tick evert second

timer1.Enabled = false; // Enable the timer

label1.AutoSize = true;

this.Controls.Add(label1);

button4.Text = "";

}

private void RefTimer()

{

try

{

string waktu;

StreamReader sr = new StreamReader(Application.StartupPath.ToString() + tClass.fn);

waktu = sr.ReadLine();

sr.Close();

iwaktu = int.Parse(waktu);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void timer1\_tick(object sender, EventArgs e)

{

label1.Text = DateTime.Now.ToLongTimeString();

try

{

RunTask();

//srv = new ServiceController("Interface Service");

//AppLog.Log(string.Format("Interface Service Status {0}", srv.Status));

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void button4\_Click(object sender, EventArgs e)

{

ServiceController byServiceName = new ServiceController();

byServiceName.ServiceName = "SCADAiService";

if (button4.Text == "Start Int-->EAM")

{

button4.Text="Stop Int-->EAM";

timer1.Enabled = true;

byServiceName.Start();

}

else

{

button4.Text="Start Int-->EAM";

timer1.Enabled = false;

byServiceName.Stop();

}

}

private void button3\_Click(object sender, EventArgs e)

{

Form3 myform = new Form3();

myform.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

Form2 myform = new Form2();

myform.Show();

}

private void button1\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void Form1\_Load(object sender, EventArgs e)

{

CheckWinServ();

}

public void CheckWinServ()

{

ClassSCD.Class1 cClass= new ClassSCD.Class1();

try

{

// Link by service name

//ServiceController byServiceName = new ServiceController("SCADAiService","Remote Server");

ServiceController byServiceName = new ServiceController();

byServiceName.ServiceName = "SCADAiService";

if (byServiceName.Status == ServiceControllerStatus.Running)

{

button4.Text = "Stop Int-->EAM";

}

else

{

button4.Text = "Start Int-->EAM";

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

button4.Text = "SERVICE NOT FOUND";

}

}

private void RunTask() {

ClassSCD.Class1 rClass = new ClassSCD.Class1();

rClass.DataCollection();

}

private void button5\_Click(object sender, EventArgs e)

{

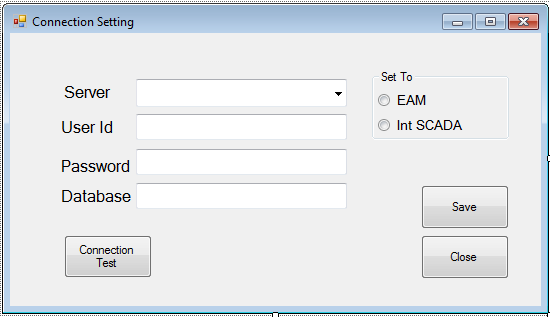
RunTask();

}

}

}

Form2.cs



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Runtime.InteropServices;

using System.IO;

using ClassSCD;

namespace InterServ

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

private enum SERVER\_TYPE : uint

{

// list of all available server types

SV\_TYPE\_WORKSTATION = 1,

SV\_TYPE\_SERVER = 2, // All Servers

SV\_TYPE\_SQLSERVER = 4, // SQL Server

SV\_TYPE\_DOMAIN\_CTRL = 8,

SV\_TYPE\_DOMAIN\_BAKCTRL = 16,

SV\_TYPE\_TIME\_SOURCE = 32,

SV\_TYPE\_AFP = 64,

SV\_TYPE\_NOVELL = 128,

SV\_TYPE\_DOMAIN\_MEMBER = 256,

SV\_TYPE\_PRINTQ\_SERVER = 512,

SV\_TYPE\_DIALIN\_SERVER = 1024,

SV\_TYPE\_XENIX\_SERVER = 2048,

SV\_TYPE\_SERVER\_UNIX = 2048,

SV\_TYPE\_NT = 4096,

SV\_TYPE\_WFW = 8192,

SV\_TYPE\_SERVER\_MFPN = 16384,

SV\_TYPE\_SERVER\_NT = 32768,

SV\_TYPE\_POTENTIAL\_BROWSER = 65536,

SV\_TYPE\_BACKUP\_BROWSER = 0X20000,

SV\_TYPE\_MASTER\_BROWSER = 0X40000,

SV\_TYPE\_DOMAIN\_MASTER = 0X80000,

SV\_TYPE\_SERVER\_OSF = 0X100000,

SV\_TYPE\_SERVER\_VMS = 0X200000,

SV\_TYPE\_WINDOWS = 0X400000, ///\* Windows95 and above \*/

SV\_TYPE\_DFS = 0X800000, ///\* Root of a DFS tree \*/

SV\_TYPE\_CLUSTER\_NT = 0X1000000, ///\* NT Cluster \*/

SV\_TYPE\_DCE = 0X10000000, ///\* IBM DSS (Directory and Security Services) or equivalent \*/

SV\_TYPE\_ALTERNATE\_XPORT = 0X20000000, ///\* return list for alternate transport \*/

SV\_TYPE\_LOCAL\_LIST\_ONLY = 0X40000000, ///\* Return local list only \*/

SV\_TYPE\_DOMAIN\_ENUM = 0X80000000,

SV\_TYPE\_ALL = 0XFFFFFFFF ///\* handy for NetServerEnum2 \*/

}

public struct SERVER\_INFO

{

public Int32 iPlatformId;

public IntPtr ipServerName;

public Int32 iVersionMajor;

public Int32 iVersionMinor;

public Int32 iType;

public IntPtr ipComment;

}

[System.Runtime.InteropServices.DllImport("netapi32.dll", EntryPoint = "NetServerEnum", ExactSpelling = false, CharSet = System.Runtime.InteropServices.CharSet.Auto, SetLastError = true)]

private static extern Int32 NetServerEnum(Int32 nullptr, Int32 Level, ref IntPtr BufPtr, Int32 BufMaxLen, ref Int32 EntriesRead, ref Int32 TotalEntries, Int32 ServerType, IntPtr Domain, Int32 ResumeHandle);

private void Form2\_Load(object sender, EventArgs e)

{

try

{

string[] a;

a = GetSQLServerList("");

for (Int32 i = 0; i <= a.Length - 1; i++)

{

if (a[i] != null) comboBox1.Items.Add(a[i]);

}

CekRadioButton();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private string[] GetSQLServerList(string sDomain)

{

Int32 iRes = 0;

Int32 iLevel = 101;

IntPtr ipBuffer = new IntPtr();

Int32 iEntriesRead = 0;

Int32 iTotalEntries = 0;

IntPtr ipDomain;

SERVER\_INFO SIF = new SERVER\_INFO();

string[] asServers = null;

Int32 lCount = 0;

IntPtr ipNext = new IntPtr();

//Get the value for the domain pointer

//ipDomain = Marshal.StringToBSTR(v\_sDomain)

ipDomain = new IntPtr(0);

//Fill the memory

iRes = NetServerEnum(0, iLevel, ref ipBuffer, -1, ref iEntriesRead, ref iTotalEntries, Convert.ToInt32(SERVER\_TYPE.SV\_TYPE\_SQLSERVER), ipDomain, 0);

if (iRes == 0)

{

//Get the size of the structure, this will be used to advance the pointer when

//reading the result of this action.

Int32 iStructureSize = Marshal.SizeOf(SIF);

//Set the output array to be the correct size

asServers = new string[iEntriesRead + 1];

//Now loop through the memory, reading the names of the servers into the array

//INSTANT C# NOTE: The ending condition of VB 'For' loops is tested only on entry to the loop. Instant C# has created a temporary variable in order to use the initial value of iEntriesRead for every iteration:

Int32 tempFor1 = iEntriesRead;

for (lCount = 0; lCount < tempFor1; lCount++)

{

//Get the memory location of the next structure

ipNext = new IntPtr(ipBuffer.ToInt32() + (iStructureSize \* lCount));

//Gather the memory into a structure

SIF = (SERVER\_INFO)Marshal.PtrToStructure(ipNext, typeof(SERVER\_INFO));

//Get the string from the referenced memory

asServers[lCount] = Marshal.PtrToStringUni(SIF.ipServerName);

}

}

return asServers;

}

private Boolean ValidationFill()

{

try

{

if (comboBox1.Text.Trim() == "")

{

MessageBox.Show("Select Server");

comboBox1.Focus();

return false;

}

if (textBox1.Text.Trim() == "")

{

MessageBox.Show("Fill User Id");

textBox1.Text = "";

textBox1.Focus();

return false;

}

if (textBox2.Text.Trim() == "")

{

MessageBox.Show("Fill Password");

textBox2.Text = "";

textBox2.Focus();

return false;

}

if (textBox3.Text.Trim() == "")

{

MessageBox.Show("Fill Database");

textBox3.Text = "";

textBox3.Focus();

return false;

}

return true;

}

catch

{

return false;

}

}

public void WriteSet(string ns,string t1,string t2,string t3,string fn)

{

try

{

if (ValidationFill())

{

Class1 myC1 = new Class1();

StreamReader sr = new StreamReader(Application.StartupPath.ToString() + fn);

sr.Close();

StreamWriter sw = new StreamWriter(Application.StartupPath.ToString() + fn);

sw.WriteLine(ns.Trim());

sw.WriteLine(t1.Trim());

sw.WriteLine(myC1.Encrypt(t2.Trim(),true));

sw.WriteLine(t3.Trim());

sw.Close();

MessageBox.Show("File Saved");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

MessageBox.Show("System Will Create New File");

Class1 myC1 = new Class1();

StreamWriter sw1 = new StreamWriter(Application.StartupPath.ToString() + fn);

sw1.WriteLine(ns.Trim());

sw1.WriteLine(t1.Trim());

sw1.WriteLine(myC1.Encrypt(t2.Trim(),true));

sw1.WriteLine(t3.Trim());

sw1.Close();

MessageBox.Show("File Successfully Created");

}

}

public void Frozen(Boolean sBol)

{

comboBox1.Enabled = sBol;

textBox1.Enabled = sBol;

textBox2.Enabled = sBol;

textBox3.Enabled = sBol;

}

public void CekRadioButton()

{

if ((radioButton1.Checked == true) || (radioButton2.Checked == true))

{

Frozen(true);

}

else

{

Frozen(false);

}

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

string ns,t1,t2,t3,fn;

ns = comboBox1.Text.ToString();

t1 = textBox1.Text.ToString();

t2 = textBox2.Text.ToString();

t3 = textBox3.Text.ToString();

if (radioButton1.Checked){

fn="\\EAMSet.csi";

WriteSet(ns, t1, t2, t3, fn);

}

else if (radioButton2.Checked){

fn="\\SCDSet.csi";

WriteSet(ns,t1,t2,t3,fn);

}

else

MessageBox.Show("Select Set To");

}

private void button3\_Click(object sender, EventArgs e)

{

ClassSCD.Class1 tCL = new ClassSCD.Class1();

if (radioButton1.Checked)

{

if (tCL.CekValidServer(tCL.ConnStringEAM()))

{

MessageBox.Show("CONNECTION TO EAM CONNECTED");

}

else

{

MessageBox.Show("CONNECTION TO EAM FAILED");

}

}

else if (radioButton2.Checked)

{

if (tCL.CekValidServer(tCL.ConnStringSCD()))

{

MessageBox.Show("CONNECTION TO SCADA INTERFACE CONNECTED");

}

else

{

MessageBox.Show("CONNECTION TO SCADA INTERFACE FAILED");

}

}

else

MessageBox.Show("Select Set To");

}

private void radioButton1\_Click(object sender, EventArgs e)

{

CekRadioButton();

}

private void radioButton2\_Click(object sender, EventArgs e)

{

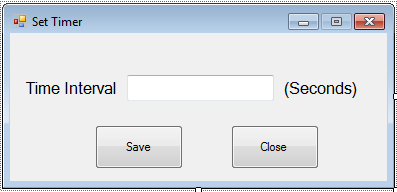
CekRadioButton();

}

}

}

Form3.cs



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.IO;

using System.Runtime.InteropServices;

using System.Collections.ObjectModel;

namespace InterServ

{

public partial class Form3 : Form

{

string fn = "\\SetTimer.txt";

string waktu;

public Form3()

{

InitializeComponent();

}

private void SaveTimer() {

try

{

if (textBox1.Text != "")

{

StreamReader sr = new StreamReader(Application.StartupPath.ToString() + fn);

sr.Close();

StreamWriter sw = new StreamWriter(Application.StartupPath.ToString() + fn);

sw.WriteLine(textBox1.Text.Trim());

sw.WriteLine("==========================================");

sw.WriteLine("Setting Timer Hanya pada baris pertama");

sw.WriteLine("Gunakan angka dengan satuan:");

sw.WriteLine("10 = interval 10 detik");

sw.WriteLine("60 = interval 1 menit (1 menit x 60 detik)");

sw.WriteLine("300 = interval 5 menit (5 menit x 60 detik)");

sw.WriteLine("1800 = interval 30 menit");

sw.WriteLine("3600 = interval 1 jam");

sw.WriteLine("86400 = interval 1 hari (24 jam)");

sw.Close();

MessageBox.Show("Timer Set Saved");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

MessageBox.Show("System Will Create New File");

StreamWriter sw1 = new StreamWriter(Application.StartupPath.ToString() + fn);

sw1.WriteLine(textBox1.Text.Trim());

sw1.WriteLine("==========================================");

sw1.WriteLine("Setting Timer Hanya pada baris pertama");

sw1.WriteLine("Gunakan angka dengan satuan:");

sw1.WriteLine("10 = interval 10 detik");

sw1.WriteLine("60 = interval 1 menit (1 menit x 60 detik)");

sw1.WriteLine("300 = interval 5 menit (5 menit x 60 detik)");

sw1.WriteLine("1800 = interval 30 menit");

sw1.WriteLine("3600 = interval 1 jam");

sw1.WriteLine("86400 = interval 1 hari (24 jam)");

sw1.Close();

MessageBox.Show("Timer Set Successfully Created");

}

finally

{ }

}

private void ReadTimer() {

StreamReader sr = new StreamReader(Application.StartupPath.ToString() + fn);

waktu = sr.ReadLine();

sr.Close();

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

SaveTimer();

}

private void Form3\_Load(object sender, EventArgs e)

{

try

{

ReadTimer();

textBox1.Text = waktu;

}

catch{

StreamWriter sw1 = new StreamWriter(Application.StartupPath.ToString() + fn);

sw1.WriteLine("5");

sw1.WriteLine("==========================================");

sw1.WriteLine("Setting Timer Hanya pada baris pertama");

sw1.WriteLine("Gunakan angka dengan satuan:");

sw1.WriteLine("10 = interval 10 detik");

sw1.WriteLine("60 = interval 1 menit (1 menit x 60 detik)");

sw1.WriteLine("300 = interval 5 menit (5 menit x 60 detik)");

sw1.WriteLine("1800 = interval 30 menit");

sw1.WriteLine("3600 = interval 1 jam");

sw1.WriteLine("86400 = interval 1 hari (24 jam)");

sw1.Close();

}

}

}

}

Program.cs

using System;

using System.Collections.Generic;

using System.Windows.Forms;

namespace InterServ

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

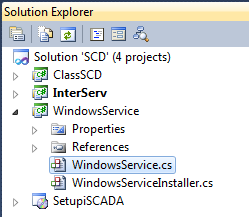
}

}

}

# WindowsService

The Windows service will be the running program as the service



WindowsService.cs

using System;

using System.Diagnostics;

using System.ServiceProcess;

using System.Timers;

using System.IO;

using System.Collections.Generic;

using System.ComponentModel;

using System.Runtime.InteropServices;

using System.Text;

using System.Windows.Forms;

using ClassSCD;

namespace WindowsService

{

class WindowsService : ServiceBase

{

/// <summary>

/// Public Constructor for WindowsService.

/// - Put all of your Initialization code here.

/// </summary>

System.Timers.Timer mytimer;

ServiceController srv;

Double iwaktu = 0;

string waktu;

ClassSCD.Class1 wClass = new ClassSCD.Class1();

public WindowsService()

{

this.ServiceName = "SCADA i Service";

this.EventLog.Source = "SCADA i Service";

this.EventLog.Log = "Application";

// These Flags set whether or not to handle that specific

// type of event. Set to true if you need it, false otherwise.

this.CanHandlePowerEvent = true;

this.CanHandleSessionChangeEvent = true;

this.CanPauseAndContinue = true;

this.CanShutdown = true;

this.CanStop = true;

if (!EventLog.SourceExists("SCADA i Service"))

EventLog.CreateEventSource("SCADA i Service", "Application");

}

public string TimerSetting()

{

StreamReader sr = new StreamReader(Application.StartupPath.ToString() + wClass.fn);

waktu = sr.ReadLine();

sr.Close();

return waktu;

}

/// <summary>

/// The Main Thread: This is where your Service is Run.

/// </summary>

static void Main()

{

ServiceBase.Run(new WindowsService());

}

/// <summary>

/// Dispose of objects that need it here.

/// </summary>

/// <param name="disposing">Whether or not disposing is going on.</param>

protected override void Dispose(bool disposing)

{

base.Dispose(disposing);

}

protected override void OnStart(string[] args)

{

TimerSetting();

iwaktu = Double.Parse(waktu);

if (mytimer == null)

mytimer = new System.Timers.Timer(iwaktu \* 1000.0);//1 Sec \* 1000Mil

mytimer.Elapsed += new ElapsedEventHandler(mytimer\_Elapsed);

mytimer.Start();

}

protected override void OnStop()

{

mytimer.Stop();

}

protected override void OnPause()

{

base.OnPause();

}

protected override void OnContinue()

{

base.OnContinue();

}

protected override void OnShutdown()

{

base.OnShutdown();

}

protected override void OnCustomCommand(int command)

{

// A custom command can be sent to a service by using this method:

//# int command = 128; //Some Arbitrary number between 128 & 256

//# ServiceController sc = new ServiceController("NameOfService");

//# sc.ExecuteCommand(command);

base.OnCustomCommand(command);

}

protected override bool OnPowerEvent(PowerBroadcastStatus powerStatus)

{

return base.OnPowerEvent(powerStatus);

}

protected override void OnSessionChange(SessionChangeDescription changeDescription)

{

base.OnSessionChange(changeDescription);

}

void mytimer\_Elapsed(object sender, ElapsedEventArgs e)

{

try

{

RunAutomation();

srv = new ServiceController("Interface Service");

AppLog.Log(string.Format("Interface Service Status {0}", srv.Status));

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void RunAutomation()

{

ClassSCD.Class1 CA = new ClassSCD.Class1();

CA.DataCollection();

/\*System.Diagnostics.Process proc = new System.Diagnostics.Process();

proc.EnableRaisingEvents = false;

string sVBSpath = System.Windows.Forms.Application.StartupPath.ToString();

string sVBSname = "RUN\_INTERFACE.bat";

string sCmdName = System.Windows.Forms.Application.StartupPath.ToString() + "\\RUN\_INTERFACE.bat";

if (!File.Exists(sVBSpath + "\\" + sVBSname))

{

StreamWriter sw2 = new StreamWriter(sVBSpath + "\\" + sVBSname);

sw2.WriteLine("Echo Off");

sw2.WriteLine("REM Isi dengan path aplikasi yang akan dirunning");

sw2.WriteLine("REM ============================================");

sw2.WriteLine("Start");

sw2.Close();

}

proc.StartInfo.FileName = sVBSpath + "\\" + sVBSname;

proc.Start();

//proc.WaitForExit();

//proc.Close();

{//write a vb script to make the process invisible

StreamWriter sw2 = new StreamWriter(sVBSpath + "\\" + sVBSname); //open text file

sw2.WriteLine("Set WshShell = CreateObject(\"WScript.Shell\")");

sw2.WriteLine("WshShell.Run chr(34) & \"" + sCmdName + "\" & Chr(34), 0");

sw2.WriteLine("Set WshShell = Nothing");

sw2.Close();

}

if (!CheckProcessExistanceByName("Interface"))

{//check if transfer desk is still running then don't do anything, wait for the next loop

proc.StartInfo.FileName = sVBSpath + "\\" + sVBSname;

proc.Start();

}\*/

}

public static class AppLog

{

public static string z = "Interface Service Control";

static EventLog Logger = null;

public static void Log(string message)

{

if (Logger == null)

{

if (!(EventLog.SourceExists(z)))

EventLog.CreateEventSource(z, "Application");

Logger = new EventLog("Application");

Logger.Source = z;

}

Logger.WriteEntry(message, EventLogEntryType.Information);

}

}

}

}

WindowsServiceInstaller.cs

using System;

using System.ComponentModel;

using System.Configuration.Install;

using System.ServiceProcess;

namespace WindowsService

{

[RunInstaller(true)]

public class WindowsServiceInstaller : Installer

{

/// <summary>

/// Public Constructor for WindowsServiceInstaller.

/// - Put all of your Initialization code here.

/// </summary>

public WindowsServiceInstaller()

{

ServiceProcessInstaller serviceProcessInstaller = new ServiceProcessInstaller();

ServiceInstaller serviceInstaller = new ServiceInstaller();

//# Service Account Information

serviceProcessInstaller.Account = ServiceAccount.LocalSystem;

serviceProcessInstaller.Username = null;

serviceProcessInstaller.Password = null;

//# Service Information

serviceInstaller.DisplayName = "SCADA i Service";

serviceInstaller.StartType = ServiceStartMode.Automatic;

// This must be identical to the WindowsService.ServiceBase name

// set in the constructor of WindowsService.cs

serviceInstaller.ServiceName = "SCADAiService";

this.Installers.Add(serviceProcessInstaller);

this.Installers.Add(serviceInstaller);

}

}

}

# SetupiSCADA

SetupiSCADA as package development for this application

