using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Net.NetworkInformation;

using System.Threading;

using EQ2008\_DataStruct;

using log4net;

using Suspe.LED.Model;

using Suspe.LED.Service.Ext;

namespace Suspe.LED.Service

{

// Token: 0x0200001C RID: 28

public class LedData : IDisposable

{

// Token: 0x06000075 RID: 117 RVA: 0x00002860 File Offset: 0x00000A60

~LedData()

{

this.Dispose(false);

}

// Token: 0x06000076 RID: 118 RVA: 0x00002890 File Offset: 0x00000A90

public void Dispose()

{

this.Dispose(true);

GC.SuppressFinalize(this);

}

// Token: 0x06000077 RID: 119 RVA: 0x0000289F File Offset: 0x00000A9F

protected virtual void Dispose(bool disposing)

{

if (this.disposed)

{

return;

}

this.disposed = true;

}

// Token: 0x06000078 RID: 120 RVA: 0x000028B3 File Offset: 0x00000AB3

public bool Start()

{

if (this.m\_isStart)

{

log.InfoFormat("任务已开启");

return this.m\_isStart;

}

if (!this.m\_isStart)

{

log.InfoFormat("正在获取屏幕配置");

this.m\_screenList = LedScreenConfigService.Instance.GetLedScreenConfigList(); //new List<ScreenCfg>(); //LdBLLCommon.FillList<ScreenCfg>(tScreenCfg.SelectSql());

log.InfoFormat("屏幕获取成功!数目-->" + m\_screenList.Count);

//var cg = new ScreenCfg();

//cg.guid = new Guid();

//cg.ScreenNo = 2;

//cg.ScreenType = 1;

//cg.CommType = 1;

//cg.Width = 320;

//cg.Height = 64;

//cg.IpAdress = "192.168.1.236";

//cg.NetPort = 5005;

//cg.ColorType = 1;

//this.m\_screenList.Add(cg);

if (this.m\_screenList != null && m\_screenList.Count != 0)

{

log.InfoFormat("更新屏幕配置开始");

this.SaveIniFile();

this.m\_isStart = true;

log.InfoFormat("更新屏幕配置结束");

}

else

{

log.InfoFormat("未配置任务屏幕");

}

}

return this.m\_isStart;

}

// Token: 0x06000079 RID: 121 RVA: 0x000028E8 File Offset: 0x00000AE8

public void Stop()

{

this.m\_isStart = false;

}

// Token: 0x0600007A RID: 122 RVA: 0x000028F4 File Offset: 0x00000AF4

private void SaveIniFile()

{

try

{

string text = string.Format("{0}EQ2008\_Dll\_Set.ini", Thread.GetDomain().BaseDirectory);// string.Format("{0}EQ2008\_Dll\_Set.ini", Thread.GetDomain().BaseDirectory);

log.InfoFormat("EQ2008\_Dll\_Set.ini 路径--->{0}", text);

File.Delete(text);

SusIniFile ldIniFile = new SusIniFile(text);

foreach (ScreenCfg screenCfg in this.m\_screenList)

{

int num = screenCfg.ScreenNo - 1;

string section = string.Format("地址：{0}", num);

ldIniFile.WriteInt(section, "CardAddress", num);

ldIniFile.WriteInt(section, "CardType", screenCfg.ScreenType);

ldIniFile.WriteInt(section, "CommunicationMode", screenCfg.CommType);

ldIniFile.WriteInt(section, "ScreemHeight", screenCfg.Height);

ldIniFile.WriteInt(section, "ScreemWidth", screenCfg.Width);

ldIniFile.WriteInt(section, "SerialBaud", 115200);

ldIniFile.WriteInt(section, "SerialNum", 1);

ldIniFile.WriteInt(section, "NetPort", screenCfg.NetPort);

string[] array = screenCfg?.IpAdress?.Split(new char[]

{

'.'

});

if (null == array) {

log.InfoFormat("ip地址为空!");

}

if (null != array)

{

//continue;

for (int i = 0; i < array.Length; i++)

{

ldIniFile.WriteString(section, string.Format("IpAddress{0}", i), array[i]);

}

}

ldIniFile.WriteInt(section, "ColorStyle", screenCfg.ColorType);

}

}

catch (Exception ex)

{

log.Error(ex);

}

}

// Token: 0x0600007B RID: 123 RVA: 0x00002A5C File Offset: 0x00000C5C

private bool LEDIsOnline(ScreenCfg scrn)

{

bool result = false;

Ping ping = new Ping();

PingReply pingReply = ping.Send(scrn.IpAdress, 120);

if (pingReply.Status == IPStatus.Success)

{

result = true;

}

else

{

string format = string.Format("{0}号屏，IP：{1}不通", scrn.ScreenNo, scrn.IpAdress);

log.InfoFormat(format, new object[0]);

}

return result;

}

// Token: 0x0600007C RID: 124 RVA: 0x00002AB8 File Offset: 0x00000CB8

public void RefreshLed()

{

foreach (ScreenCfg screenCfg in this.m\_screenList)

{

log.InfoFormat("开始操作屏幕{0}, IP{1}", new object[]

{

screenCfg.ScreenNo,

screenCfg.IpAdress

});

this.RefreshData(screenCfg);

}

}

// Token: 0x0600007D RID: 125 RVA: 0x00002B30 File Offset: 0x00000D30

private void RefreshData(ScreenCfg scrn)

{

uint tickCount = LdAPI.GetTickCount();

log.InfoFormat("tickCount-->{0}", tickCount);

bool flag = false;

DataRow dataRow = null;

if (scrn.TblSchedule != null && scrn.SchIdx < scrn.TblSchedule.Rows.Count)

{

dataRow = scrn.TblSchedule.Rows[scrn.SchIdx];

int num = dataRow.LdToInt32("Duration");

uint num2 = (tickCount - scrn.TickDura) / 1000u;

if (scrn.IsFinish && (ulong)num2 >= (ulong)((long)num))

{

scrn.SchIdx++;

if (scrn.SchIdx >= scrn.TblSchedule.Rows.Count)

{

dataRow = null;

}

else

{

dataRow = scrn.TblSchedule.Rows[scrn.SchIdx];

scrn.TickDura = tickCount;

scrn.TickRefresh = tickCount;

scrn.IsFinish = false;

scrn.RowIdx = 0;

flag = true;

}

}

}

if (dataRow == null)

{

this.GetScreenSchedule(scrn);

if (scrn.TblSchedule.Rows.Count <= 0)

{

return;

}

dataRow = scrn.TblSchedule.Rows[0];

scrn.SchIdx = 0;

scrn.TickDura = tickCount;

scrn.TickRefresh = tickCount;

scrn.IsFinish = false;

scrn.RowIdx = 0;

flag = true;

}

bool bNext = false;

int num3 = dataRow.LdToInt32("Duration");

uint num4 = (tickCount - scrn.TickDura) / 1000u;

if ((ulong)num4 >= (ulong)((long)num3))

{

scrn.TickDura = tickCount;

bNext = true;

}

if (!flag)

{

int num5 = dataRow.LdToInt32("RefreshCycle");

uint num6 = (tickCount - scrn.TickRefresh) / 1000u;

if ((ulong)num6 >= (ulong)((long)num5))

{

scrn.TickRefresh = tickCount;

flag = true;

}

}

int num7 = dataRow.LdToInt32("InfoType");

string text = dataRow.LdToString("InfoDetail");

switch (num7)

{

case 130:

{

int fnt = 12;

if (string.IsNullOrEmpty(text))

{

text = string.Format("{0}\r\n{1}", "LdSysInfo.ProductName", DateTime.Now.ToString("yyyy年M月d日 hh:mm:ss"));//LdSysInfo.ProductName, DateTime.Now.ToString("yyyy年M月d日 hh:mm:ss"));

}

else

{

int num8 = text.IndexOf('\\');

if (num8 > 0)

{

string s = text.Substring(0, num8);

int num9 = 0;

if (int.TryParse(s, out num9))

{

fnt = num9;

}

text = text.Substring(num8 + 1);

}

text = text.Replace("date", DateTime.Now.ToString("yyyy年M月d日"));

text = text.Replace("time", DateTime.Now.ToString("hh:mm:ss"));

}

this.PrintInfo(scrn, fnt, text);

scrn.IsFinish = true;

return;

}

case 131:

this.PageDailyPlan(scrn, bNext, flag);

return;

case 136:

this.PageHourPlan(scrn, bNext, flag);

return;

//case 136:

// this.PageHourPlanV2(scrn, bNext, flag);

//return;

case 137:

this.PageHourPlanV3(scrn, bNext, flag);

return;

}

this.PrintInfo(scrn, 12, "功能未开放的信息类型");

}

// Token: 0x0600007E RID: 126 RVA: 0x00002E14 File Offset: 0x00001014

private void GetScreenSchedule(ScreenCfg scrn)

{

if (scrn.TblSchedule != null)

{

scrn.TblSchedule.Clear();

}

else

{

scrn.TblSchedule = new DataTable("tScreenSchedule");

}

//LdBLLCommon.FillDataTable(scrn.TblSchedule, string.Format("SELECT \* FROM tScreenSchedule WITH(NOLOCK) WHERE IsValid=1 AND ScreenCfg\_guid=N'{0}' ORDER BY Odr", scrn.guid));

//页面

var scPageList = LedScreenConfigService.Instance.GetLedScreenPageList(scrn.Id);

scrn.TblSchedule = new System.Data.DataTable();

scrn.TblSchedule.Columns.Add("Duration");

scrn.TblSchedule.Columns.Add("RefreshCycle");

scrn.TblSchedule.Columns.Add("InfoType");

scrn.TblSchedule.Columns.Add("InfoDetail");

if (scPageList.Count == 0)

{

log.ErrorFormat("GetScreenSchedule--->屏幕页面为空:{0}", scrn.Id);

return;

}

foreach (var page in scPageList)

{

DataRow dr = scrn.TblSchedule.NewRow();

dr["Duration"] = page.Times;

dr["RefreshCycle"] = page.RefreshCycle;

dr["InfoType"] = page.InfoType;

dr["InfoDetail"] = page.CusContent;

scrn.TblSchedule.Rows.Add(dr);

}

}

// Token: 0x0600007F RID: 127 RVA: 0x00002E68 File Offset: 0x00001068

private void PrintInfo(ScreenCfg scrn, int fnt, string strInfo)

{

if (this.LEDIsOnline(scrn) && EQ2008.User\_RealtimeConnect(scrn.ScreenNo))

{

string format = string.Format("打开{0}号屏", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

User\_FontSet user\_FontSet = default(User\_FontSet);

user\_FontSet.bFontBold = false;

user\_FontSet.bFontItaic = false;

user\_FontSet.bFontUnderline = false;

user\_FontSet.colorFont = 255;

user\_FontSet.strFontName = "宋体";

user\_FontSet.iAlignStyle = 0;

user\_FontSet.iVAlignerStyle = 0;

user\_FontSet.iRowSpace = 0;

user\_FontSet.iFontSize = fnt;

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 0, 320, 64, strInfo, ref user\_FontSet);

EQ2008.User\_RealtimeDisConnect(scrn.ScreenNo);

format = string.Format("关闭{0}号屏", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

return;

}

scrn.IsFinish = true;

}

// Token: 0x06000080 RID: 128 RVA: 0x00002F5C File Offset: 0x0000115C

/// <summary>

/// 制单日产量

/// </summary>

/// <param name="scrn"></param>

/// <param name="bNext"></param>

/// <param name="bRefresh"></param>

private void PageDailyPlan(ScreenCfg scrn, bool bNext, bool bRefresh)

{

if (bRefresh)

{

if (scrn.TblInfo != null)

{

scrn.TblInfo.Clear();

}

else

{

scrn.TblInfo = new DataTable("tInfoDetail");

}

//using (SqlCommand sqlCommand = new SqlCommand("LD\_LED\_DailyOutputV2"))

//{

// sqlCommand.CommandType = CommandType.StoredProcedure;

// sqlCommand.Parameters.AddWithValue("@lineguid", scrn.Line\_guid);

// //LdBLLCommon.FillDataTable(scrn.TblInfo, sqlCommand);

//}

scrn.TblInfo = LedScreenConfigService.Instance.GetTodayProcessOrderYield(scrn.GroupNo);

scrn.RowIdx = 0;

}

if (scrn.TblInfo == null)

{

scrn.IsFinish = true;

return;

}

if (bNext)

{

scrn.RowIdx += 3;

}

int num = scrn.TblInfo.Rows.Count - scrn.RowIdx;

if (num > 3)

{

num = 3;

}

if (num <= 0)

{

scrn.IsFinish = true;

log.InfoFormat("{0}号屏, 日目标达成率内容为空", new object[]

{

scrn.ScreenNo

});

return;

}

if (scrn.RowIdx + num >= scrn.TblInfo.Rows.Count)

{

scrn.IsFinish = true;

}

if (this.LEDIsOnline(scrn) && EQ2008.User\_RealtimeConnect(scrn.ScreenNo))

{

string format = string.Format("打开{0}号屏，日目标达成率", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

User\_FontSet user\_FontSet = default(User\_FontSet);

user\_FontSet.bFontBold = false;

user\_FontSet.bFontItaic = false;

user\_FontSet.bFontUnderline = false;

user\_FontSet.colorFont = 255;

user\_FontSet.strFontName = "宋体";

user\_FontSet.iAlignStyle = 0;

user\_FontSet.iVAlignerStyle = 0;

user\_FontSet.iRowSpace = 0;

user\_FontSet.iFontSize = 12;

int iHeight = 64;

//输出空白

log.InfoFormat("制单号 制单数 累计产出 当日 计划 达成率--->标题开始", new object[0]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 0, 320, 64, " ", ref user\_FontSet);

log.InfoFormat("制单号 制单数 累计产出 当日 计划 达成率--->标题开始，空白推送时间测试", new object[0]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 16, 0, 48, iHeight, "制单号", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 80, 0, 48, iHeight, "制单数", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 144, 0, 32, iHeight, "产出", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 192, 0, 32, iHeight, "当日", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 232, 0, 32, iHeight, "计划", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 272, 0, 48, iHeight, "达成率", ref user\_FontSet);

log.InfoFormat("制单号 制单数 累计产出 当日 计划 达成率--->标题结束", new object[0]);

for (int i = 0; i < 3; i++)

{

if (i < num)

{

DataRow dataRow = scrn.TblInfo.Rows[scrn.RowIdx + i];

user\_FontSet.iAlignStyle = 0;

string processOrderNo = string.Format("{0}", dataRow["ProcessOrderNo"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 16 + i \* 16, 72, iHeight, processOrderNo, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

string processOrderCount = string.Format("{0}", dataRow["ProcessOrderCount"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 80, 16 + i \* 16, 48, iHeight, processOrderCount, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

string totlOutCount = string.Format("{0}", dataRow["TotlOutCount"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 136, 16 + i \* 16, 48, iHeight, totlOutCount, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

string todayCount = string.Format("{0}", dataRow["TodayCount"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 192, 16 + i \* 16, 32, iHeight, todayCount, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

string plan = string.Format("{0}", dataRow["TPlan"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 232, 16 + i \* 16, 32, iHeight, plan, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

var eff = decimal.Parse("0");

if (int.Parse(plan) == 0)

{

eff = 0;

}

else

{

eff = (decimal.Parse(todayCount) / int.Parse(plan) \* 100);

}

string tEff = string.Format("{0:P2}", eff);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 272, 16 + i \* 16, 48, iHeight, tEff, ref user\_FontSet);

log.InfoFormat(string.Format("制单日产量 输出内容：{0} {1} {2} {3} {4} ", new object[]

{

processOrderNo,

processOrderCount,

totlOutCount,

todayCount,

tEff

}));

}

else

{

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 16 + i \* 16, 320, iHeight, " ", ref user\_FontSet);

}

}

EQ2008.User\_RealtimeDisConnect(scrn.ScreenNo);

format = string.Format("关闭{0}号屏", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

return;

}

scrn.IsFinish = true;

}

// Token: 0x06000081 RID: 129 RVA: 0x0000337C File Offset: 0x0000157C

private void PageHourPlan(ScreenCfg scrn, bool bNext, bool bRefresh)

{

if (bRefresh)

{

if (scrn.TblHourPlan != null)

{

scrn.TblHourPlan.Clear();

}

else

{

scrn.TblHourPlan = new DataTable("tLEDHourPlan");

}

//using (SqlCommand sqlCommand = new SqlCommand("LD\_LED\_HourOutput"))

//{

// sqlCommand.CommandType = CommandType.StoredProcedure;

// sqlCommand.Parameters.AddWithValue("@lineguid", scrn.Line\_guid);

// //LdBLLCommon.FillDataTable(scrn.TblHourPlan, sqlCommand);

//}

scrn.TblHourPlan = LedScreenConfigService.Instance.HoursPlanInfo(scrn.GroupNo);

scrn.RowIdx = 0;

}

if (scrn.TblHourPlan == null)

{

scrn.IsFinish = true;

return;

}

if (bNext)

{

scrn.RowIdx += 3;

}

int num = scrn.TblHourPlan.Rows.Count - scrn.RowIdx;

if (num > 3)

{

num = 3;

}

if (num <= 0)

{

scrn.IsFinish = true;

log.InfoFormat("{0}号屏, 小时计划达成率内容为空", new object[]

{

scrn.ScreenNo

});

return;

}

if (scrn.RowIdx + num >= scrn.TblHourPlan.Rows.Count)

{

scrn.IsFinish = true;

}

if (this.LEDIsOnline(scrn) && EQ2008.User\_RealtimeConnect(scrn.ScreenNo))

{

string format = string.Format("打开{0}号屏, 小时计划达成率", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

User\_FontSet user\_FontSet = default(User\_FontSet);

user\_FontSet.bFontBold = false;

user\_FontSet.bFontItaic = false;

user\_FontSet.bFontUnderline = false;

user\_FontSet.colorFont = 255;

user\_FontSet.strFontName = "宋体";

user\_FontSet.iAlignStyle = 0;

user\_FontSet.iVAlignerStyle = 0;

user\_FontSet.iRowSpace = 0;

user\_FontSet.iFontSize = 12;

int iHeight = 64;

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 0, 320, 64, " ", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 24, 0, 32, iHeight, "时间", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 80, 0, 32, iHeight, "计划", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 120, 0, 32, iHeight, "实际", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 160, 0, 48, iHeight, "达成率", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 216, 0, 48, iHeight, "不良数", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 272, 0, 48, iHeight, "合格率", ref user\_FontSet);

log.InfoFormat("时间 计划 实际 达成率 不良数 合格率", new object[0]);

for (int i = 0; i < 3; i++)

{

if (i < num)

{

DataRow dataRow = scrn.TblHourPlan.Rows[scrn.RowIdx + i];

user\_FontSet.iAlignStyle = 1;

string text = string.Format("{0}", dataRow["Times"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 16 + i \* 16, 72, iHeight, text, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

string text2 = string.Format("{0}", dataRow["PlanQty"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 80, 16 + i \* 16, 32, iHeight, text2, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

string text3 = string.Format("{0}", dataRow["TOutput"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 120, 16 + i \* 16, 32, iHeight, text3, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

string text4 = string.Format("{0:P2}", dataRow["Eff"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 160, 16 + i \* 16, 48, iHeight, text4, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

string text5 = string.Format("{0:D2}", dataRow["TFail"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 216, 16 + i \* 16, 48, iHeight, text5, ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

string text6 = string.Format("{0:P2}", dataRow["EffFail"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 272, 16 + i \* 16, 48, iHeight, text6, ref user\_FontSet);

log.InfoFormat("{0} {1} {2} {3} {4} {5}", new object[]

{

text,

text2,

text3,

text4,

text5,

text6

});

}

else

{

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 16 + i \* 16, 320, iHeight, " ", ref user\_FontSet);

}

}

EQ2008.User\_RealtimeDisConnect(scrn.ScreenNo);

format = string.Format("关闭{0}号屏", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

log.InfoFormat(" ", new object[0]);

return;

}

scrn.IsFinish = true;

}

// Token: 0x06000082 RID: 130 RVA: 0x00003850 File Offset: 0x00001A50

private void PageHourPlanV2(ScreenCfg scrn, bool bNext, bool bRefresh)

{

if (bRefresh)

{

if (scrn.TblHourPlan != null)

{

scrn.TblHourPlan.Clear();

}

else

{

scrn.TblHourPlan = new DataTable("tLEDHourPlan");

}

using (SqlCommand sqlCommand = new SqlCommand("LD\_LED\_HourOutputV2"))

{

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@lineguid", scrn.Line\_guid);

//LdBLLCommon.FillDataTable(scrn.TblHourPlan, sqlCommand);

}

scrn.RowIdx = 0;

}

if (scrn.TblHourPlan == null)

{

scrn.IsFinish = true;

return;

}

if (bNext)

{

scrn.RowIdx += 2;

}

int num = scrn.TblHourPlan.Rows.Count - scrn.RowIdx;

if (num > 2)

{

num = 2;

}

if (num <= 0)

{

scrn.IsFinish = true;

log.InfoFormat("{0}号屏, 小时计划达成率V2内容为空", new object[]

{

scrn.ScreenNo

});

return;

}

if (scrn.RowIdx + num >= scrn.TblHourPlan.Rows.Count)

{

scrn.IsFinish = true;

}

if (this.LEDIsOnline(scrn) && EQ2008.User\_RealtimeConnect(scrn.ScreenNo))

{

string format = string.Format("打开{0}号屏, 小时计划达成率V2", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

User\_FontSet user\_FontSet = default(User\_FontSet);

user\_FontSet.bFontBold = false;

user\_FontSet.bFontItaic = false;

user\_FontSet.bFontUnderline = false;

user\_FontSet.colorFont = 255;

user\_FontSet.strFontName = "宋体";

user\_FontSet.iAlignStyle = 0;

user\_FontSet.iVAlignerStyle = 0;

user\_FontSet.iRowSpace = 0;

user\_FontSet.iFontSize = 12;

int iHeight = 64;

user\_FontSet.iAlignStyle = 0;

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 0, 320, 64, " ", ref user\_FontSet);

user\_FontSet.iAlignStyle = 1;

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 0, 72, iHeight, "Time", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 80, 0, 32, iHeight, "Plan", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 120, 0, 48, iHeight, "Actual", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 176, 0, 56, iHeight, "Fulfill", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 240, 0, 80, iHeight, "FPY", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 16, 72, iHeight, "时间", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 80, 16, 32, iHeight, "计划", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 120, 16, 48, iHeight, "实际", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 176, 16, 56, iHeight, "达成率", ref user\_FontSet);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 240, 16, 80, iHeight, "一次通过率", ref user\_FontSet);

log.InfoFormat("Time Plan Actual Fulfill FPY", new object[0]);

log.InfoFormat("时间 计划 实际 达成率 一次通过率", new object[0]);

for (int i = 0; i < 2; i++)

{

if (i < num)

{

DataRow dataRow = scrn.TblHourPlan.Rows[scrn.RowIdx + i];

user\_FontSet.iAlignStyle = 1;

string text = string.Format("{0}", dataRow["vRow"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 32 + i \* 16, 72, iHeight, text, ref user\_FontSet);

string text2 = string.Format("{0}", dataRow["PlanQty"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 80, 32 + i \* 16, 32, iHeight, text2, ref user\_FontSet);

string text3 = string.Format("{0}", dataRow["TOutput"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 120, 32 + i \* 16, 48, iHeight, text3, ref user\_FontSet);

string text4 = string.Format("{0:P1}", dataRow["Eff"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 176, 32 + i \* 16, 56, iHeight, text4, ref user\_FontSet);

string text5 = string.Format("{0:P1}", dataRow["EffFail"]);

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 240, 32 + i \* 16, 80, iHeight, text5, ref user\_FontSet);

log.InfoFormat(string.Format("{0} {1} {2} {3} {4}", new object[]

{

text,

text2,

text3,

text4,

text5

}), new object[0]);

}

else

{

user\_FontSet.iAlignStyle = 0;

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 32 + i \* 16, 320, iHeight, " ", ref user\_FontSet);

}

}

EQ2008.User\_RealtimeDisConnect(scrn.ScreenNo);

format = string.Format("关闭{0}号屏", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

log.InfoFormat(" ", new object[0]);

return;

}

scrn.IsFinish = true;

}

// Token: 0x06000083 RID: 131 RVA: 0x00003D74 File Offset: 0x00001F74

private void PageHourPlanV3(ScreenCfg scrn, bool bNext, bool bRefresh)

{

if (bRefresh)

{

if (scrn.TblHourPlanV3 != null)

{

scrn.TblHourPlanV3.Clear();

}

else

{

scrn.TblHourPlanV3 = new DataTable("tLEDHourPlan");

}

using (SqlCommand sqlCommand = new SqlCommand("LD\_LED\_HourOutputV3"))

{

sqlCommand.CommandType = CommandType.StoredProcedure;

sqlCommand.Parameters.AddWithValue("@lineguid", scrn.Line\_guid);

//LdBLLCommon.FillDataTable(scrn.TblHourPlanV3, sqlCommand);

}

scrn.RowIdx = 0;

}

if (scrn.TblHourPlanV3 == null)

{

scrn.IsFinish = true;

return;

}

if (bNext)

{

scrn.RowIdx += 4;

}

int num = scrn.TblHourPlanV3.Rows.Count - scrn.RowIdx;

if (num > 4)

{

num = 4;

}

if (num <= 0)

{

scrn.IsFinish = true;

log.InfoFormat("{0}号屏, 小时计划达成率V3内容为空", new object[]

{

scrn.ScreenNo

});

return;

}

if (scrn.RowIdx + num >= scrn.TblHourPlanV3.Rows.Count)

{

scrn.IsFinish = true;

}

if (this.LEDIsOnline(scrn) && EQ2008.User\_RealtimeConnect(scrn.ScreenNo))

{

string format = string.Format("打开{0}号屏, 小时计划达成率V3", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

User\_FontSet user\_FontSet = default(User\_FontSet);

user\_FontSet.bFontBold = false;

user\_FontSet.bFontItaic = false;

user\_FontSet.bFontUnderline = false;

user\_FontSet.colorFont = 255;

user\_FontSet.strFontName = "宋体";

user\_FontSet.iAlignStyle = 0;

user\_FontSet.iVAlignerStyle = 0;

user\_FontSet.iRowSpace = 0;

user\_FontSet.iFontSize = 12;

int iHeight = 64;

user\_FontSet.iAlignStyle = 0;

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, 0, 320, 64, " ", ref user\_FontSet);

for (int i = 0; i < 4; i++)

{

if (i < num)

{

DataRow dr = scrn.TblHourPlanV3.Rows[scrn.RowIdx + i];

user\_FontSet.iAlignStyle = 1;

string text = dr.LdToString("ColA");

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, i \* 16, 72, iHeight, text, ref user\_FontSet);

string text2 = dr.LdToString("ColB");

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 80, i \* 16, 32, iHeight, text2, ref user\_FontSet);

string text3 = dr.LdToString("ColC");

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 120, i \* 16, 48, iHeight, text3, ref user\_FontSet);

string text4 = dr.LdToString("ColD");

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 176, i \* 16, 56, iHeight, text4, ref user\_FontSet);

string text5 = dr.LdToString("ColE");

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 240, i \* 16, 80, iHeight, text5, ref user\_FontSet);

log.InfoFormat(string.Format("{0} {1} {2} {3} {4}", new object[]

{

text,

text2,

text3,

text4,

text5

}), new object[0]);

}

else

{

user\_FontSet.iAlignStyle = 0;

EQ2008.User\_RealtimeSendText(scrn.ScreenNo, 0, i \* 16, 320, iHeight, " ", ref user\_FontSet);

}

}

EQ2008.User\_RealtimeDisConnect(scrn.ScreenNo);

format = string.Format("关闭{0}号屏", scrn.ScreenNo);

log.InfoFormat(format, new object[0]);

log.InfoFormat(" ", new object[0]);

return;

}

scrn.IsFinish = true;

}

// Token: 0x06000084 RID: 132 RVA: 0x00004118 File Offset: 0x00002318

private void PageEmpOutput(short lineNo)

{

}

// Token: 0x06000085 RID: 133 RVA: 0x0000411A File Offset: 0x0000231A

private void PageEffRanking(short lineNo)

{

}

// Token: 0x040000B9 RID: 185

public IList<ScreenCfg> m\_screenList;

// Token: 0x040000BA RID: 186

private bool disposed;

// Token: 0x040000BB RID: 187

private bool m\_isStart;

private ILog log;

public LedData(ILog log)

{

this.log = log;

}

}

}using log4net;

using Suspe.LED.Model;

using SuspeSys.Dao;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Suspe.LED.Service

{

public class LedScreenConfigService

{

readonly static ILog log = LogManager.GetLogger(typeof(LedScreenConfigService));

private LedScreenConfigService() { }

public static readonly LedScreenConfigService Instance = new LedScreenConfigService();

public IList<ScreenCfg> GetLedScreenConfigList()

{

IList<ScreenCfg> scList = new List<ScreenCfg>();

try

{

var sql = "select \* from LEDScreenConfig where Enable=@Enable";

var ledsConfigList = DapperHelp.Query<LedScreenConfig>(sql, new { Enable = 1 }).ToList<LedScreenConfig>();

if (ledsConfigList.Count == 0)

{

log.InfoFormat("未找到屏幕配置");

}

foreach (var lc in ledsConfigList)

{

var sc = new ScreenCfg();

sc.Id = lc.Id;

sc.ColorType = null != lc.ColorType ? lc.ColorType.Value : 0;

sc.CommType = null != lc.CommunicationWay ? lc.CommunicationWay.Value : 0;

sc.Height = null != lc.SHeight ? lc.SHeight.Value : 0;

sc.Width = null != lc.SWidth ? lc.SWidth.Value : 0;

sc.ScreenNo = !string.IsNullOrEmpty(lc.ScreenNo) ? int.Parse(lc.ScreenNo) : 0;

sc.ScreenType = !string.IsNullOrEmpty(lc.ControllerKey) ? int.Parse(lc.ControllerKey) : 0;

sc.NetPort = null != lc.Port ? lc.Port.Value : 0;

sc.IpAdress = lc.IpAddress;

//页面

var scPageList = GetLedScreenPageList(lc.Id);

sc.TblSchedule = new System.Data.DataTable();

sc.TblSchedule.Columns.Add("Duration");

sc.TblSchedule.Columns.Add("RefreshCycle");

sc.TblSchedule.Columns.Add("InfoType");

sc.TblSchedule.Columns.Add("InfoDetail");

sc.TblSchedule.Columns.Add("GroupNo");

sc.GroupNo = lc.GroupNo?.Trim();

if (scPageList.Count == 0)

{

log.ErrorFormat("GetLedScreenConfigList--->屏幕页面为空:{0}", lc.Id);

continue;

}

foreach (var page in scPageList)

{

DataRow dr = sc.TblSchedule.NewRow();

dr["Duration"] = page.Times;

dr["RefreshCycle"] = page.RefreshCycle;

dr["InfoType"] = page.InfoType;

dr["InfoDetail"] = page.CusContent;

dr["GroupNo"] = page.GroupNo?.Trim();

sc.TblSchedule.Rows.Add(dr);

}

scList.Add(sc);

}

}

catch (Exception ex)

{

log.ErrorFormat("GetLedScreenConfigList--->异常:{0}", ex);

}

return scList;

}

public IList<LedScreenPage> GetLedScreenPageList(string screenConfigId)

{

var sql = "select Sp.\*,SC.GroupNo from LEDScreenConfig SC INNER JOIN LEDScreenPage SP ON SP.LEDSCREENCONFIG\_Id=SC.Id where SP.LEDSCREENCONFIG\_Id=@screenConfigId AND SP.Enabled=1";

var ledScreenPageList = DapperHelp.Query<LedScreenPage>(sql, new { screenConfigId = screenConfigId }).ToList<LedScreenPage>();

return ledScreenPageList;

}

public DataTable GetTodayProcessOrderYield(string groupNo)

{

string sql = string.Format(@"SELECT ProcessOrderNo, ProcessOrderCount

,ISNULL(TotalProdOutNum,0) TotlOutCount,ISNULL(TodayCount,0)TodayCount,ISNULL(TPlan,0) TPlan

FROM (

SELECT ProcessOrderNo,T\_TotalProdOutNum.TotalProdOutNum,T\_TodayCount.TodayCount,T\_Main.TPlan,

(

SELECT

SUM(CONVERT(INT ,ISNULL(t3.Total,0))) ProcessOrderCount

FROM dbo.ProcessOrder t1

LEFT JOIN ProcessOrderColorItem t2 ON t1.Id=t2.PROCESSORDER\_Id

LEFT JOIN dbo.ProcessOrderColorSizeItem t3 ON t3.PROCESSORDERCOLORITEM\_Id = t2.Id

WHERE t1.Id=T\_Main.ProcessOrder\_Id

)ProcessOrderCount

FROM

( SELECT ProcessOrder\_Id,ProcessOrderNo,SUM(TargetNum) TPlan FROM(

SELECT ProcessOrder\_Id,ProcessOrderNo,PROCESSFLOWCHART\_Id,T2.TargetNum FROM Products TP

LEFT JOIN ProcessFlowChart T2 ON TP.PROCESSFLOWCHART\_Id=T2.ID)T\_Main

GROUP BY ProcessOrder\_Id,ProcessOrderNo

) T\_Main

LEFT JOIN(

SELECT ProcessOrderId,SUM(ISNULL(SizeNum, 0)) AS TotalProdOutNum

FROM dbo.SucessProcessOrderHanger where GroupNo='{0}'

Group BY ProcessOrderId

) AS T\_TotalProdOutNum ON T\_TotalProdOutNum.ProcessOrderId = T\_Main.ProcessOrder\_Id

LEFT JOIN

(

SELECT ProcessOrderId,COUNT(ProcessFlowCode) TodayCount FROM SucessProcessOrderHanger

WHERE InsertDateTime BETWEEN CONVERT(varchar(10), GETDATE(), 120) AND

CONVERT(varchar(10), DATEADD(day, 1, GETDATE()), 120) AND GroupNo='{0}'

GROUP BY ProcessOrderId

) T\_TodayCount ON T\_TodayCount.ProcessOrderId=T\_Main.ProcessOrder\_Id

)Res

", groupNo);

return DapperHelp.Query(sql);

}

/// <summary>

/// 小时计划

/// </summary>

public DataTable HoursPlanInfo(string groupNo)

{

var sqlWhereData = string.Empty;

var sqlWhereSet = string.Empty;

var sqlCheckData = string.Format($@" SELECT FlowCode,InsertDateTime FROM dbo.HangerProductFlowChart WHERE InsertDateTime BETWEEN CONVERT(varchar(100), GETDATE(), 111) AND GETDATE() AND CAST(HangerNo AS INT) > 0 AND GroupNo='{groupNo}'

UNION ALL

SELECT FlowCode,InsertDateTime FROM dbo.SuccessHangerProductFlowChart WHERE InsertDateTime BETWEEN CONVERT(varchar(100), GETDATE(), 111) AND GETDATE() AND FlowType=1 AND CAST(HangerNo AS INT) > 0 AND GroupNo='{groupNo}' ");

var dtCheck = DapperHelp.Query(sqlCheckData);

if (dtCheck.Rows.Count != 0)

{

sqlWhereData += $" InsertDateTime BETWEEN CONVERT(varchar(100), GETDATE(), 111) AND GETDATE()";

}

else

{

var sqlMax = $"SELECT MAX(InsertDateTime) InsertDateTime FROM dbo.HangerProductFlowChart";

var maxDate = DapperHelp.QueryForObject<DateTime?>(sqlMax);

if (maxDate == null) return null;

sqlWhereData += $" InsertDateTime BETWEEN '{maxDate.Value.ToString("yyyy-MM-dd")}' AND '{maxDate.Value.AddDays(1).ToString("yyyy-MM-dd")}'";

}

var sqlCheckSet = string.Format(@"SELECT TOP 2 \* FROM [dbo].[LEDHoursPlanTableItem] WHERE

BeginDate>=CONVERT(varchar(100), GETDATE(), 111) AND EndDate<GETDATE()

OR (GETDATE() BETWEEN BeginDate AND EndDate)");

var dtCheckSet = DapperHelp.Query(sqlCheckSet);

if (dtCheckSet.Rows.Count != 0)

{

sqlWhereSet = "BeginDate>=CONVERT(varchar(100), GETDATE(), 111) AND EndDate<GETDATE() OR (GETDATE() BETWEEN BeginDate AND EndDate)";

}

else

{

var sqlCheckSet2 = string.Format(@"SELECT max(BeginDate) BeginDate FROM [dbo].[LEDHoursPlanTableItem]");

var maxDate = DapperHelp.QueryForObject<DateTime?>(sqlCheckSet2);

if (maxDate == null) return null;

sqlWhereSet += $" BeginDate BETWEEN '{maxDate.Value.ToString("yyyy-MM-dd")}' AND '{maxDate.Value.AddDays(1).ToString("yyyy-MM-dd")}'";

}

var sqlInfo = string.Format($@"SELECT ISNULL(Res.PlanNum,0)PlanNum,FORMAT(Res.BeginDate,'tt-HH') BTime,FORMAT(Res.EndDate,'tt-HH') ENDTimes,RealiyCount,Res.DefectCount FROM(

SELECT T\_Times.BeginDate,T\_Times.EndDate,T\_Times.PlanNum,

(

SELECT COUNT(FlowCode) TodayCount FROM HangerProductFlowChart WHERE {sqlWhereData} AND CAST(HangerNo AS INT) > 0 AND GroupNo='{groupNo}'

) RealiyCount

,(

SELECT COUNT(FlowCode) DefectCount FROM(

SELECT FlowCode,InsertDateTime FROM dbo.HangerProductFlowChart WHERE {sqlWhereData} AND FlowType=1 AND CAST(HangerNo AS INT) > 0 AND GroupNo='{groupNo}'

UNION ALL

SELECT FlowCode,InsertDateTime FROM dbo.SuccessHangerProductFlowChart WHERE {sqlWhereData} AND FlowType=1 AND CAST(HangerNo AS INT) > 0 AND GroupNo='{groupNo}'

)T\_FlowCode

) DefectCount

From(

SELECT TOP 2 \* FROM [dbo].[LEDHoursPlanTableItem] WHERE

{sqlWhereSet}

ORDER BY BeginDate DESC,EndDate DESC

)T\_Times

)Res");

var sqlInfoTotal = string.Format($@"SELECT SUM(ISNULL(Res.PlanNum,0))PlanNum,SUM(ISNULL(RealiyCount,0)) RealiyCount,SUM(ISNULL(Res.DefectCount,0)) DefectCount FROM(

SELECT T\_Times.BeginDate,T\_Times.EndDate,T\_Times.PlanNum,

(

SELECT COUNT(FlowCode) TodayCount FROM HangerProductFlowChart WHERE {sqlWhereData} AND CAST(HangerNo AS INT) > 0 AND GroupNo='{groupNo}'

) RealiyCount

,(

SELECT COUNT(FlowCode) DefectCount FROM(

SELECT FlowCode,InsertDateTime FROM dbo.HangerProductFlowChart WHERE {sqlWhereData} AND FlowType=1 AND CAST(HangerNo AS INT) > 0 AND GroupNo='{groupNo}'

UNION ALL

SELECT FlowCode,InsertDateTime FROM dbo.SuccessHangerProductFlowChart WHERE {sqlWhereData} AND FlowType=1 AND CAST(HangerNo AS INT) > 0 AND GroupNo='{groupNo}'

)T\_FlowCode

) DefectCount

From(

SELECT TOP 2 \* FROM [dbo].[LEDHoursPlanTableItem] WHERE

{sqlWhereSet}

ORDER BY BeginDate DESC,EndDate DESC

)T\_Times

)Res

");

var dtResult = new DataTable();

dtResult.Columns.Add("Times");

dtResult.Columns.Add("PlanQty");

dtResult.Columns.Add("TOutput");

dtResult.Columns.Add("Eff", typeof(decimal));

dtResult.Columns.Add("TFail");

dtResult.Columns.Add("EffFail", typeof(decimal));

var dtInfo = DapperHelp.Query(sqlInfo);

var dtInfoTotal = DapperHelp.Query(sqlInfoTotal);

if (dtInfo.Rows.Count == 0)

{

log.Info($"组【{groupNo}】 无数据!");

return null;

}

if (dtInfo.Rows.Count == 2)

{

DataRow drFisrt = dtResult.NewRow();

var drrrInfo1 = dtInfo.Rows[1];

var beginTimeFirst = Convert.ToString(drrrInfo1["BTime"]);

var endTimeFirst = Convert.ToString(drrrInfo1["ENDTimes"]);

var begTimesArrsFirst = beginTimeFirst.Split('-');

var endTimesArrsFirst = endTimeFirst.Split('-');

drFisrt["Times"] = string.Format("{0}-{1}", begTimesArrsFirst[0] + begTimesArrsFirst[1], endTimesArrsFirst[1]);

drFisrt["PlanQty"] = drrrInfo1["PlanNum"];

drFisrt["TOutput"] = drrrInfo1["RealiyCount"];

var tEff1 = decimal.Parse("0");

if (int.Parse(string.IsNullOrEmpty(Convert.ToString(drrrInfo1["PlanNum"])) ? "0" : Convert.ToString(drrrInfo1["PlanNum"])) != 0)

{

tEff1 = decimal.Parse(Convert.ToString(drrrInfo1["RealiyCount"])) / Int64.Parse(string.IsNullOrEmpty(Convert.ToString(drrrInfo1["PlanNum"])) ? "0" : Convert.ToString(drrrInfo1["PlanNum"]));

}

drFisrt["Eff"] = tEff1;//.ToString("{0:P2}");

drFisrt["TFail"] = drrrInfo1["DefectCount"];

drFisrt["EffFail"] = (1 - tEff1 < 0 ? 0 : (1 - tEff1));

dtResult.Rows.Add(drFisrt);

DataRow drSecond = dtResult.NewRow();

var drrrInfo2 = dtInfo.Rows[0];

var beginTimeSecond = Convert.ToString(drrrInfo2["BTime"]);

var endTimeSecond = Convert.ToString(drrrInfo2["ENDTimes"]);

var begTimesArrsSecond = beginTimeSecond.Split('-');

var endTimesArrsSecond = endTimeSecond.Split('-');

drSecond["Times"] = string.Format("{0}-{1}", begTimesArrsSecond[0] + begTimesArrsSecond[1], endTimesArrsSecond[1]);//string.Format("{0}-{1}", Convert.ToString(drrrInfo2["BTime"]), Convert.ToString(drrrInfo2["ENDTimes"]));

drSecond["PlanQty"] = drrrInfo2["PlanNum"];

drSecond["TOutput"] = drrrInfo2["RealiyCount"];

var tEff2 = decimal.Parse("0");

if (int.Parse(string.IsNullOrEmpty(Convert.ToString(drrrInfo2["PlanNum"])) ? "0" : Convert.ToString(drrrInfo2["PlanNum"])) != 0)

{

tEff2 = decimal.Parse(Convert.ToString(drrrInfo2["RealiyCount"])) / Int64.Parse(string.IsNullOrEmpty(Convert.ToString(drrrInfo2["PlanNum"])) ? "0" : Convert.ToString(drrrInfo2["PlanNum"]));

}

drSecond["Eff"] = tEff2;//.ToString("{0:P2}");

drSecond["TFail"] = drrrInfo2["DefectCount"];

drSecond["EffFail"] = (1 - tEff2 < 0 ? 0 : (1 - tEff2));

dtResult.Rows.Add(drSecond);

}

else

{

DataRow drSecond = dtResult.NewRow();

var drrrInfo2 = dtInfo.Rows[0];

var beginTimeSecond = Convert.ToString(drrrInfo2["BTime"]);

var endTimeSecond = Convert.ToString(drrrInfo2["ENDTimes"]);

var begTimesArrsSecond = beginTimeSecond.Split('-');

var endTimesArrsSecond = endTimeSecond.Split('-');

drSecond["Times"] = string.Format("{0}-{1}", begTimesArrsSecond[0] + begTimesArrsSecond[1], endTimesArrsSecond[1]);//string.Format("{0}-{1}", Convert.ToString(drrrInfo2["BTime"]), Convert.ToString(drrrInfo2["ENDTimes"]));

drSecond["PlanQty"] = drrrInfo2["PlanNum"];

drSecond["TOutput"] = drrrInfo2["RealiyCount"];

var tEff2 = decimal.Parse("0");

if (int.Parse(string.IsNullOrEmpty(Convert.ToString(drrrInfo2["PlanNum"])) ? "0" : Convert.ToString(drrrInfo2["PlanNum"])) != 0)

{

tEff2 = decimal.Parse(Convert.ToString(drrrInfo2["RealiyCount"])) / Int64.Parse(string.IsNullOrEmpty(Convert.ToString(drrrInfo2["PlanNum"])) ? "0" : Convert.ToString(drrrInfo2["PlanNum"]));

}

drSecond["Eff"] = tEff2;//.ToString("{0:P2}");

drSecond["TFail"] = drrrInfo2["DefectCount"];

drSecond["EffFail"] = (1 - tEff2 < 0 ? 0 : (1 - tEff2));

dtResult.Rows.Add(drSecond);

}

var drrrTotal = dtInfoTotal.Rows[0];

DataRow totalRows = dtResult.NewRow();

totalRows["Times"] = "Total累计";

totalRows["PlanQty"] = drrrTotal["PlanNum"];

totalRows["TOutput"] = drrrTotal["RealiyCount"];

var tEff = decimal.Parse("0");

if (int.Parse(string.IsNullOrEmpty(Convert.ToString(drrrTotal["PlanNum"])) ? "0" : Convert.ToString(drrrTotal["PlanNum"])) != 0)

{

tEff = decimal.Parse(Convert.ToString(drrrTotal["RealiyCount"])) / Int64.Parse(string.IsNullOrEmpty(Convert.ToString(drrrTotal["PlanNum"])) ? "0" : Convert.ToString(drrrTotal["PlanNum"]));

}

totalRows["Eff"] = tEff;

totalRows["TFail"] = drrrTotal["DefectCount"];

totalRows["EffFail"] = (1 - tEff < 0 ? 0 : (1 - tEff));

dtResult.Rows.Add(totalRows);

return dtResult;

}

}

}