Tech Note 805
Validating Custom Developed Client Controls for Use in InTouch®

All Tech Notes, Tech Alerts and KBCD documents and software are provided "as is" without warranty of any kind. See the **Terms of Use** for more information.

Topic#: 002591
Created: November 2011

Introduction

With a managed InTouch application, you can develop your own Client Control objects, import them through IDE and embed them in InTouch windows via ArchestrA Symbols. This gives you the ability to add highly-customized functionality to InTouch and Application Server applications.

In the meantime, it may also silently introduce unwanted harm to the InTouch application which may go unnoticed until Runtime. For example

- The Client Control can cause View to leak memory and handle, eventually causing View to crash.

- The Client Control can cause View to leak GDI Handle, eventually causing View window to fracture, or to crash with an assertion error.

This *Tech Note* provides one easy and handy way to bulletproof a custom-developed Client Control to keep the InTouch application safe from potential harm, thus saving time for additional troubleshooting or even downtime later in production.

Application Versions

- Application Server 3.1 and later

- InTouch 10.1 and later

Validation Procedure

1. Import the custom developed Client Control(s) into the ArchestrA IDE.

2. Embed the custom Client Control objects into an ArchestrA symbol using the ArchestrA Graphic Editor. Configure the Client Control objects just as you would normally apply in regular application.

3. Create a test InTouch application, embed the ArchestrA symbol (containing the custom Client Control objects) in InTouch windows, and configure the ArchestrA symbol just as you would normally do with your regular InTouch application.

4. Add scripts and make other configurations to InTouch application as of necessary. For example, I created scripts that automatically open and close the windows in a fixed interval.

5. Start WindowViewer™; or create and deploy an InTouchViewApp object instance from the template, then run WindowViewer from the deployed InTouchView node.

6. Simulate regular runtime environment, such as opening/closing windows, interacting with the Client Control objects etc.

Tools and Validation Method

Performance Monitor (Perfmon)

This is a commonly available utility on all OS, and can be started by running Perfmon command line in DOS Window, or directly from Start/Run.

- Add **Private Bytes** and **Handle Count** counters of the **Process** object from InTouch **View** instance.

  WindowViewer has to be running in order to have View showing in the instance list. The utility's UI looks slightly different on Windows XP, Windows 2003 to Windows 2008, Windows 7. The following graphics show Windows 2008 as the example.
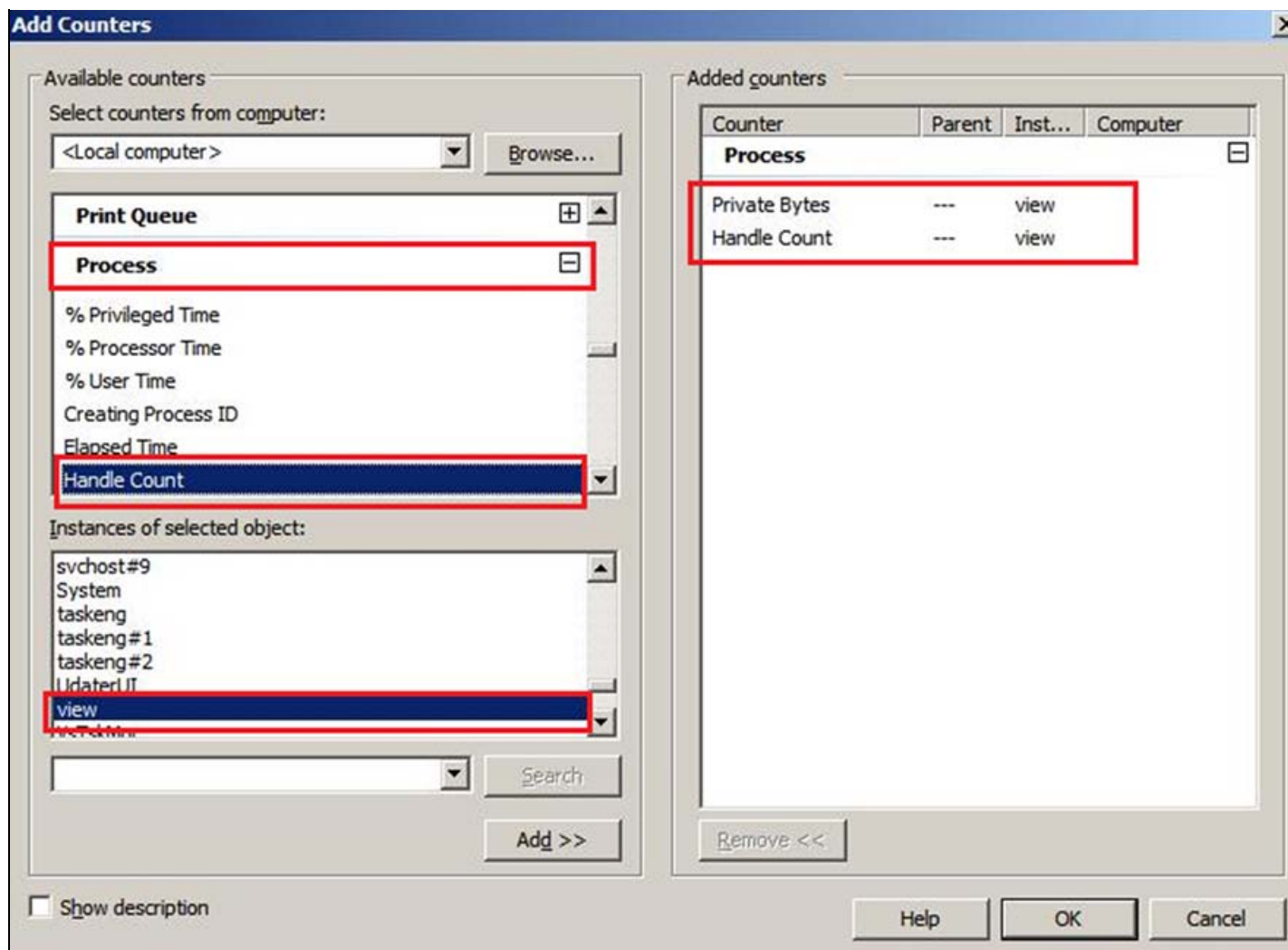
FIGURE 1: ADDING COUNTERS IN PERFORMANCE MONITOR

- Choose the proper values for the **Sample Interval**, **Graph Duration**, and **Vertical Scale** properties etc. to fit your testing needs (Figures 2 and 2a below).
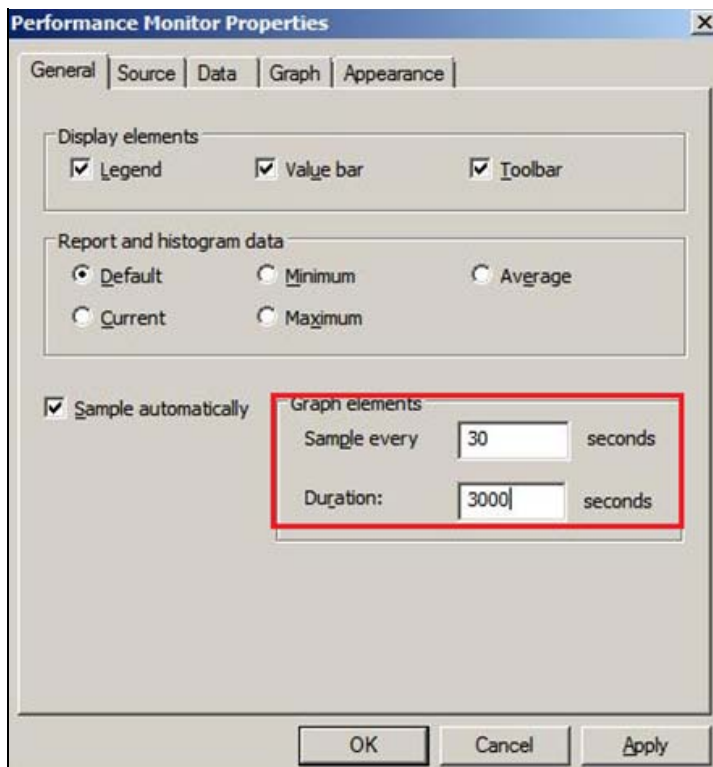
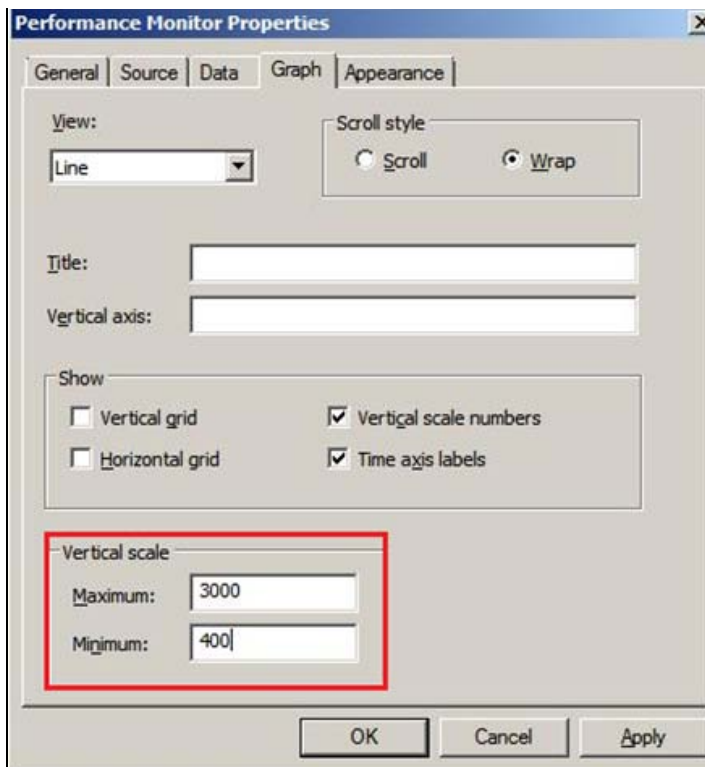**Figure 2: Chart Display Properties > General**

Figure 2a: Chart Display Properties > Graph

- You can generate a Counter Log (file) for later analysis, or for when the testing may take a long time to complete.

GDIView

This is a unique tool that displays the list of GDI handles (to brushes, pens, fonts, bitmaps and others) opened by every process. It displays the total count for each type of GDI handle, as well as detailed information about each handle. This tool is very useful for detecting and tracing GDI resources leak in a process runtime, such as Client Control object embedded in InTouch View (process) through ArchestrA graphics.

Two different versions are available for 32-bit and 64-bit OS.

- **Download this free utility**

Process Explorer

- This is another very useful handy tool for debugging runtime resource access conditions of a process.
- Process Explorer is bundled in the Microsoft Windows **Sysinternals** package.
- **Download this free utility**

Results and Analysis

- With Performance Monitor, Handle Count and Memory leaking conditions were detected while operating InTouch windows that included the custom Client Control embedded in the InTouch window via ArchestrA graphic.
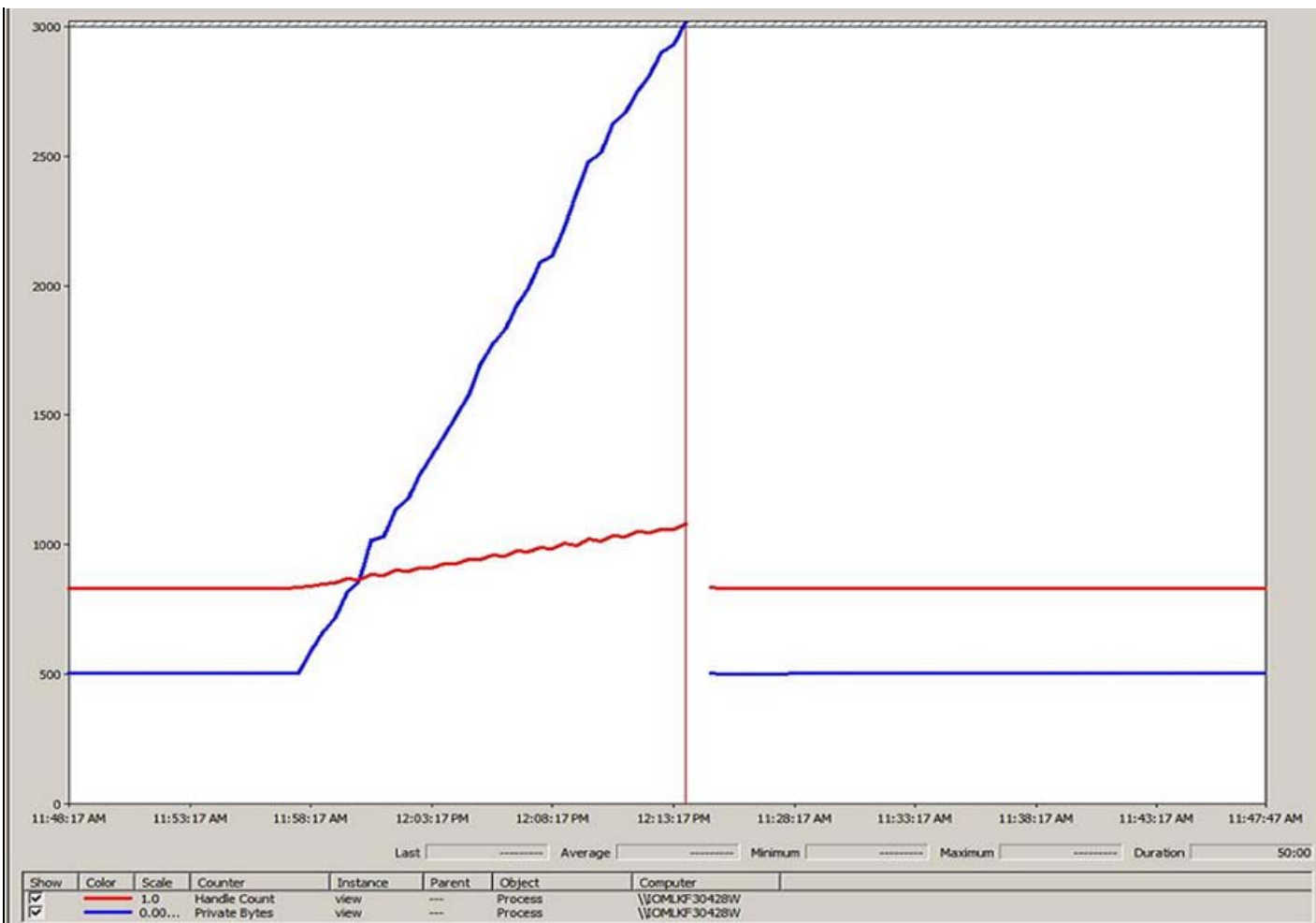
FIGURE 3: 'PRIVATE BYTES' AND 'HANDLE COUNT' COUNTERS VERSUS TIME. THE BLUE LINE IS FOR VIEW'S 'PRIVATE BYTES' COUNTER VERSUS TIME

- With GDIView, GDI Handles leaking condition was detected while operating InTouch windows with custom Client Controls (embedded in InTouch windows via ArchestrA graphics) (Figure 4 below).

FIGURE 4. VIEW'S GDI HANDLE COUNT KEEPS INCREASING WHILE VIEW IS RUNNING. THE CIRCLED NUMBERS SHOW A NET INCREASE

- With Process Explorer, Handle Count, Memory and GDI Handle leaking conditions were detected (Figure 5 below).

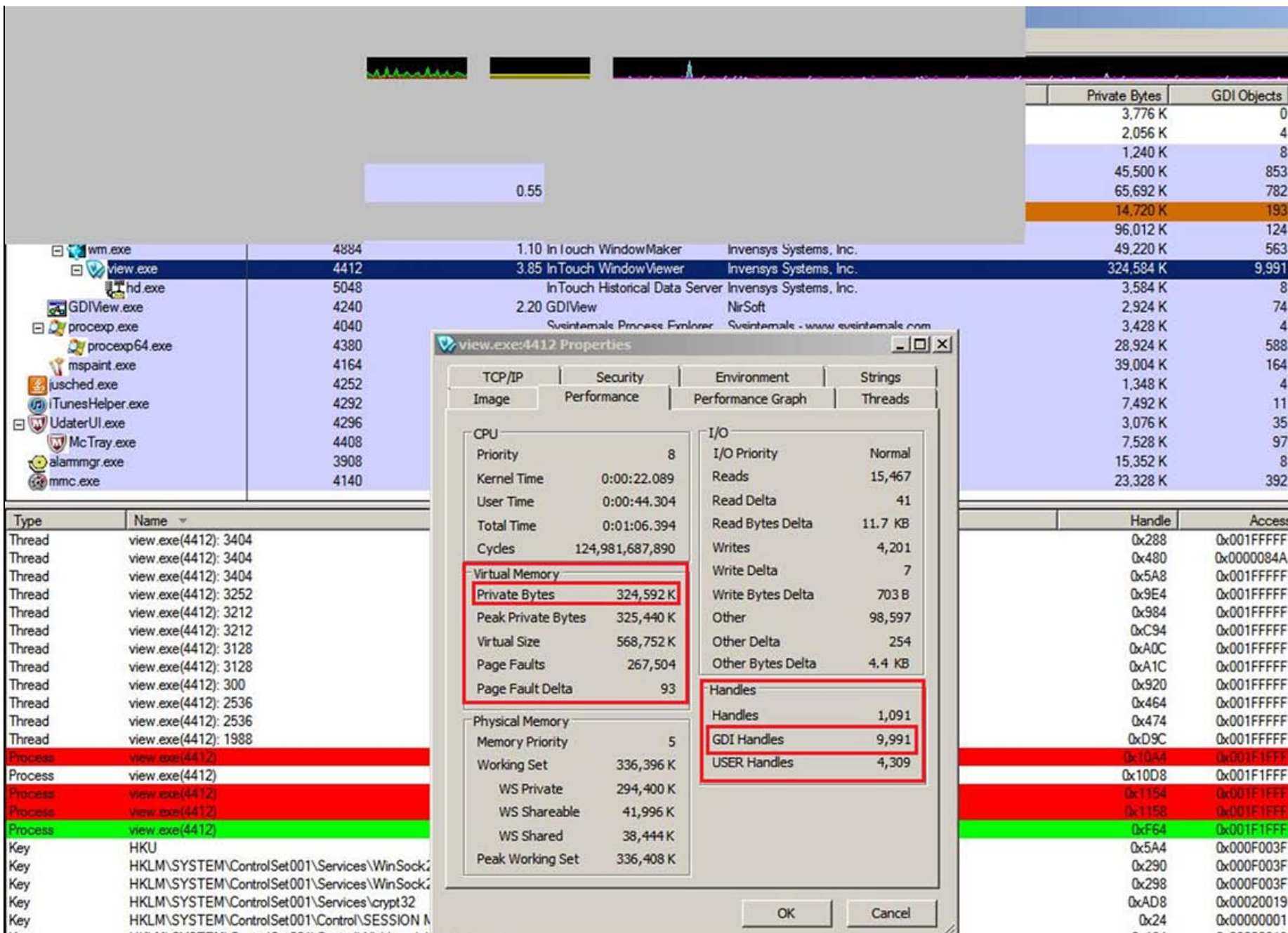**FIGURE 5: 'VIEW' PROCESS CONDITIONS AT START OF WINDOWVIEWER**

FIGURE 6: 'VIEW' PROCESS CONDITIONS AT CERTAIN POINT AFTER WINDOWVIEWER HAS BEEN RUNNING FOR A TIME

Summary

The cause of memory, GDI Handle and other resource leaks can come from many different aspects in programming practices. When developing a custom Client Control, it is always a good practice to remember to use implicit and explicit ways to dispose of (even override Standard Dispose methods) the resources allocated during the

initialization and run of the Client Control.

The information provided on this website (**Click HERE**) provides guidelines for implementing Finalize and Dispose methods to clean up unmanaged resources.

C. He, E. Xu

For technical support questions, send an e-mail to **wwsupport@invensys.com**.

 Back to top