

The CodeSite logging system gives developers deeper insight into how their code is executing. From development to production, CodeSite helps developers locate problems in their code, monitor application execution, and even analyze performance. CodeSite's logging classes make it very easy to capture all kinds of information from simple text to complex data structures, and then send that information to a live display or a log file; locally or remotely.

### COMPARE OBJECTS INSPECTOR

Property	Value 1	Value 2
BackColor	Control	Navy
ControlBox	True	False
FormBorderStyle	Sizable	FixedDialog
KeyPreview	False	True
Padding	8, 8, 8, 8	4, 4, 4, 4
Size	748, 376	738, 366

Select two object messages in the message log and this inspector automatically displays their differences.

### XML INSPECTOR

XML Structure

- Document
  - Page
    - StackPanel
      - Label
        - CodeSite 4.0
      - TextBlock
        - StackPanel
          - Image
          - Label

Analyze the structure of your XML data. Great for XAML.

Tag Parameter	Value
FontWeight	Bold
Background	Blue
Foreground	White

### COLLECTION INSPECTOR

StatusBarPanelCollection Collection Items

- 0 - StatusBarPanel: {#11}
- 1 - StatusBarPanel: {#29}
- 2 - StatusBarPanel: {Let's Get Rocked}

Capture and inspect ALL items in your collections.

Property	Value
Alignment	Left
AutoSize	Spring
BorderStyle	Sunken
Icon	null
MinWidth	10
Name	stsItemText
Style	Text
Tag	null
Text	Let's Get Rocked
ToolTipText	Item Text
Width	143

**Navigational Tools**  
Browse by message type, search for text, or jump to a method or bookmark.

**Categories**  
Use separate loggers to color code and group messages by category.

**Inspector Pane**  
Displays additional data for the selected message; e.g. the Object Inspector shows the property values of the specified object.

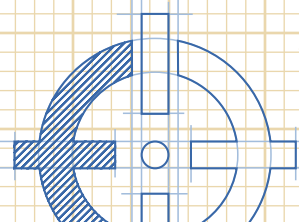
**Organize Messages**  
Create additional views to isolate messages or let the Message Organizer do it for you automatically!

**Call Stack**  
Displays the instrumented method chain for the selected message.

**ScratchPad**  
Monitor data changes without impacting the message log.

CodeSite also addresses several shortcomings in traditional logging/tracing solutions (see opposite side for a complete list). For example, CodeSite does not rely on arbitrary logging levels to control what information gets logged. Instead, developers utilize separate CodeSite loggers to categorize messages. In the screen shot above, logging messages associated with the Expression class are sent using a separate logger that has its Category set to "Expression" and its CategoryColor set to Orange. To stop logging these messages, simply disable the logger—messages from other loggers will continue to be sent.

Ultimately, all CodeSite logging messages are analyzed in the CodeSite Viewer, which includes specially designed tools for inspecting, navigating, and filtering CodeSite messages. There are more than 15 inspectors available for analyzing the details of complex messages—four of these inspectors are illustrated on this page. There are also several tools available to aid in navigating the message log including searching, bookmarks, message type browsing, and method jumping. To reduce the amount of information to be analyzed, the CodeSite Viewer includes sophisticated filtering tools such as the Message Organizer that can automatically create new views to isolate messages of the same application, category, thread, etc.



# Comparing CODESITE and the Trace class in the Microsoft .NET Framework

The Microsoft .NET Framework provides built-in support for code instrumentation and logging with the Trace class and other supporting classes in the System.Diagnostics namespace. The following feature matrix highlights the differences between the capabilities of the Trace class and those of the CodeSite logging system.

Feature	Trace	CODESITE
<b>INFORMATION CAPTURE (LOGGING CLASSES)</b>		
Uses a specialized logging class to send messages that contain information captured during a program's execution to one or more targets	●	●
Create additional instances of logging class for more control over message generation		●
Send text/string messages	●	●
Send simple data elements (e.g. integers, floats, dates, rectangles, etc.) without string conversions		●
Send complex data structures in a single method call (e.g. exceptions, string lists, object properties, collections, bitmaps, text files, memory status, registry entries, system information, version information, xml data, stack traces, custom data)		●
Categorize messages by prefixing category string to message text	●	●
Categorize messages by specifying a Category property in logging class		●
Categorize messages by specifying a message type		●
Record message type, category, data type name, application name, computer, process ID, thread name, date, timestamp, and additional details with each message.		●
Block certain messages from being generated by creating and using separate Switch objects and calling special methods (e.g. WriteLineIf())	●	
Block certain messages from being generated by simply changing the Enabled property of the desired CodeSiteLogger instance		●
Thread-safe logging classes	●	●
Record short-lived transient data (e.g. mouse position) in a separate area from the message log		●
<b>TARGETS</b>		
Send tracing messages to multiple targets	●	●
Send messages to IDE debug window	●	
Send messages to specialized viewer		●
Send messages to log file	●	●
Send messages from multiple applications to the same log file		●
Send messages from multiple instances of an application to the same log file		●
Send messages to viewer or log file on remote machine		●
Control which messages are sent to which targets		●
<b>ANALYSIS</b>		
Message logs are analyzed in a sophisticated, specially designed viewer		●
Message types are visually identifiable		●
Filter message log on various criteria without deleting messages		●
Organize messages into separate views		●
Automatically organize incoming messages into separate views		●
Inspect complex data (object properties, bitmaps, xml data, collections, etc.)		●
Advanced searching and navigational tools available		●
Collapse/expand method blocks		●
Analyze timings between messages		●
Customize appearance of messages based on category or message type		●
Export messages to XML		●
<b>DEPLOYMENT</b>		
Configuration file support	●	●
Remove method calls from generated code by removing conditional compiler symbol	●	●
Redistributable Client Tools (Dispatcher, Live Viewer, File Viewer, Controller)		●
Automatically manage multi-part log files		●
Automatically name log files based on date		●
Limit amount of logging information retained in log files		●
Utilize path variables to dynamically control and limit where log files can be saved		●

