Golden Code Development Corporation
1455 Old Alabama Road, Suite 135
Roswell, Georgia 30076 USA
+1 678-352-2301 (Tel)
+1 678-352-2305 (Fax)
www.goldencode.com

Briefing

Golden Code* Trace Suite*

Contents

Executive Summary

PC and LAN systems are very expensive to support and maintain. PCs and Client/Server technologies may have opened up a new level of functional capability but they also greatly increased the total cost of ownership (TCO) of these systems.

There are three primary factors driving the increased TCO:

- complexity of these systems
- decentralized install base and "ownership"
- large numbers

One way of reducing TCO is to enable a deeper look into the complexity of these systems. By placing software agents in key locations in a system one can achieve this goal. If designed properly, these agents can be deployed across a large number of systems and remotely or programmatically accessed as necessary. Traces generated by these agents can be transferred to a central site for analysis and action.

Golden Code provides a suite of trace tools, designed to reduce the TCO of PC and LAN systems. This documents introduces these tools.

What is the Value of Trace Tools?

OS/2's strength has always been its relatively low cost of ownership, in relation to traditional PC solutions like the many flavors of Windows. This comes from IBM's heritage in building systems designed for large organizations. All large organizations need the ability to customize and control a system to a much greater degree than a consumer requires. This key difference is what enables an OS/2 implementation that significantly reduces TCO. It is possible (even probable) that an implementation of OS/2 can be costly and poorly designed. IBM's key design point: the flexibility and control built into OS/2, comes with the cost of increased knowledge and experience required to properly implement an OS/2 system. IBM's technology can be very low cost, if the right senior level resources are engaged in the implementation. This makes the system poorly suited for the consumer market as ease of use was not the overriding design goal.

When measured in industry terms, IBM did not have tremendous success with OS/2. However, when considered on its own, OS/2 really has done quite well. It is widely deployed in the Banking, Insurance, Utilities, Transportation, Manufacturing and Retail industries. It is especially well suited to environments that have a large number of similar sites, such as large branch banking networks. It is still well installed in these environments to this day. IBM even provided a set of management tools and technologies for OS/2 that took OS/2 far beyond the management capabilities of other PC platforms. In many ways OS/2 is still a better managed platform than is possible under Windows. And Linux is not even "on the map" in terms of management capabilities, when compared to OS/2. Even so, it is sometimes quite hard to solve problems, develop software or engineer systems on OS/2. There are a great number of gaps in the software which IBM provides.

IBM's 1998 introduction of WorkSpace On-Demand (WSOD) provided a new level of enablement for reducing TCO. It enhanced the capabilities already available since the late 1980's in Remote Initial Program Load (RIPL). RIPL allows a client workstation to obtain its boot image and permits all applications/data to reside on a server, negating the costly aspects of software distribution/deployment and providing new levels of centralized control over the OS/2 client. The WSOD client is really just OS/2 delivered via a RIPL on "steroids". It is OS/2 and takes advantage of all the management infrastructure that OS/2 enjoys. While this concept of server-managed clients takes OS/2 into a new frontier of low TCO, it also brings with it even more reliance upon strong senior resources for proper implementation and support. In addition, the resulting OS/2

system is so network-centric that new tools and methodologies are absolutely critical to properly manage this environment.

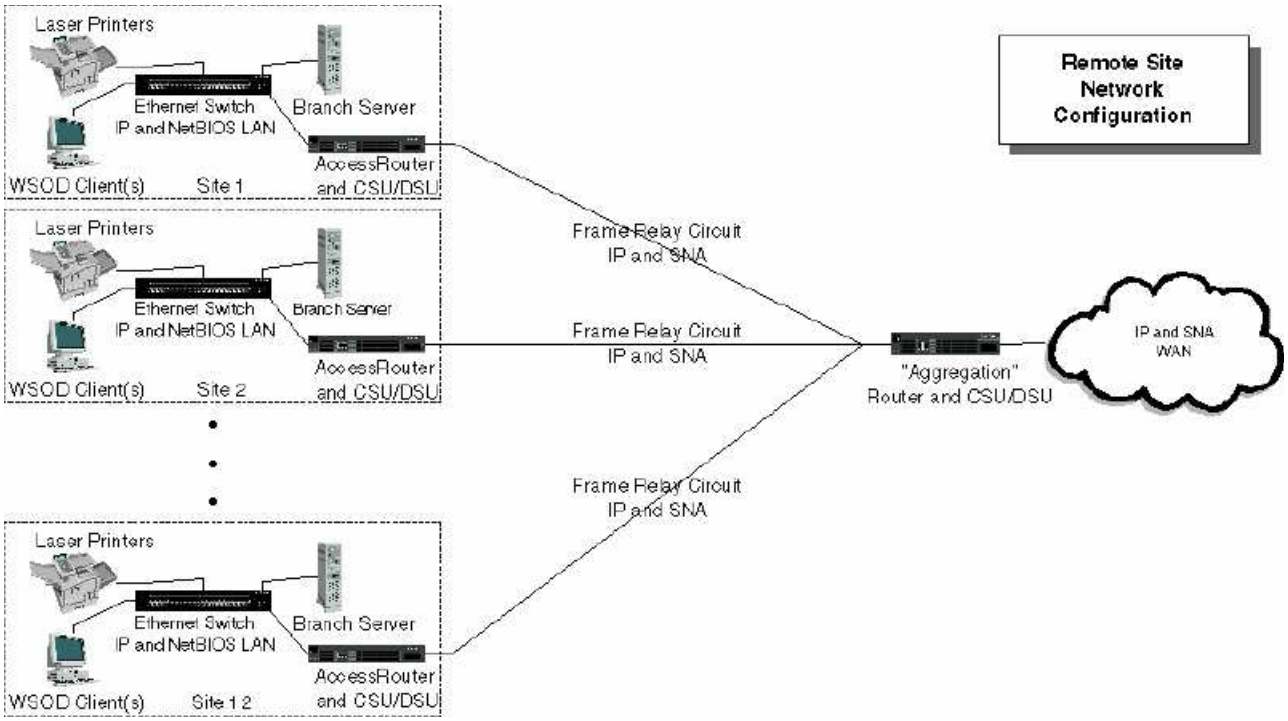
In all OS/2 implementations there exists a significant number of problems for which diagnosis, development and engineering tools are scant or non-existent. Resolving these problems is like trying to find your way out of a pitch dark room. You will eventually find your way out, but how much damage will have been done to yourself and the surroundings and how much time will you have wasted? Who would choose to grope for a way out of the dark room if an alternative existed? If there were floodlights in the room, there would be no problem.

Golden Code Development builds "floodlights" for OS/2!

Real Life Example

The Golden Code Trace Suite is a set of tools and utilities which constitute such a floodlight. The power of these tools can be illustrated with an example from a Golden Code Trace Utilities customer with a very large number of sites (over 4000 sites). Golden Code's Trace Suite is installed at each site. Network Trace for OS/2 is installed on every site server. Kernel Trace for OS/2 is installed on every OS/2 machine including WSOD clients. All systems have full remote access using Netfinity and the servers additionally have Telnet/FTP access. The following is a simplified diagram of their network:

Site Network Diagram



Each site system is comprised of one server and one or more client workstations. All sites have identical systems including LAN Server configuration. Only IP and SNA traffic ever flows over the WAN. This means that the OS/2 server in every site office has its own IP address/subnet and SNA configuration BUT it has the same NetBIOS "machine name" as every other site office. For example, every site server has the name "LANSRV". Each site server is a domain controller managing a domain named "BRNCHDOM". Since no two LAN Server systems ever need to communicate over the WAN, this duplication is not a limitation. This LAN Server domain implementation was designed only as a LAN system.

All sites have a single local "access" router. This device handles the WAN communications over a fractional T1 frame relay circuit to the central WAN. As seen above, multiple offices connect back to a "concentration" router, which handles aggregating the WAN traffic to/from the WAN. In this specific case, 12 offices are concentrated into a single router at a central site. From there, these 12 sites all get access to the rest of the systems on the WAN.

One morning, the first level help desk got a support call from a site. The users arrived at the site and tried to start their WSOD clients, but no clients would boot. The call was shortly escalated to the second level help desk. Here they looked at the site server in question and determined that LAN Server could not be started. The following error message was displayed:

```
NET3056: A system error has occurred

For more information, type HELP NET3056
NET3502: OS/2 error 52 has occurred.
SYS0052: A duplicate name exists on the network.
```

These technicians looked at the common logfiles such as `LANTRAN.LOG` and LAN Server's `NET ERROR` log. Other than this one message, there was no other indication of a problem. No change control had been executed at this site the previous night, whether by electronic or physical means. The users had been using the system without problems the previous evening. In other words, it was known that no changes had been made to the server or site network since the last time the system was successfully used.

The second level helpdesk escalated the call to level 3. They explained the situation. In order to get a duplicate name on the network, two or more of the site servers must be visible to one another. This would happen if they were placed on the same LAN segment or if there was a NetBIOS bridging problem on the router. Normally NetBIOS is not allowed to pass through the router. Since it is not a routable protocol, it can only flow over a router, if there is a bridge defined. On traditional SNA networks that ride over an IP backbone, this SNA transport is made through a bridging technology called DataLink Switching (DLSw). A point to point bridge is made between two "peer" routers and the SNA traffic is encapsulated in IP frames and forwarded to the remote peer. This bridging technology can forward NetBIOS traffic too, but this is normally disabled. So to isolate the cause, the most immediate question is: **where on the network is the duplicate name coming from?**

The level 3 engineer used Telnet to remotely access the site server in question. A trace session with Network Trace for OS/2 was started. This is a simple command line program. After 2 minutes, the trace was stopped and the resulting Sniffer-compatible trace file was transferred back to the level 3 help desk using FTP. This entire sequence had been completed within 10 minutes of receiving the call! The trace was opened in the Golden Code Trace Analyzer and the engineer started scanning through the frames. Using the Trace Analyzer, the engineer was able to see all network traffic from the remote LAN segment. Within 2 minutes of scanning the trace, the failing component that caused the problem had been isolated.

There was something very unusual about this trace. **There was not only one duplicate name (LANSRV) on the network, but actually there were 12 systems that were visible on this segment, that all advertised the same name.** The level 3 engineer knew that the odds were negligible that 12 site servers were installed at this one site during the night (especially without anyone knowing)! Thus the problem had to be caused by a change in the network. **But what network component was at fault?** The number 12 suggested very strongly that the problem was at the concentration router, since this is the one place in the network where 12 site networks connect together.

The level 3 engineer asked the level 2 help desk if there were calls from other sites, reporting the same problem. The answer was that "a very large number" of calls had just come in. The level 3 engineer asked them to see if there were 12 such calls. There were. This problem's severity was now clear: **some change had been made to the concentration router, such that 12 sites were completely off line**. If LAN Server can not start then no WSOD client can boot either. This means the entire site is down.

In many organizations, system support organizations and network support organizations do not always cooperate harmoniously. This happens to be true at this company. It is not uncommon that the network support team will look at its logs and monitors and state categorically that "there is no network problem", pointing instead back to the local server or client as the problem. This method has likely proven to be pretty successful, because in most PC/LAN environments the PC most often IS the cause of the problem. However, with this enterprise, the OS/2 systems are so highly managed, standardized and automated that this rule is no longer valid. Nevertheless, the network support team still tends to rely primarily upon its logs and monitors for conclusive evidence of a problem. Unfortunately, just because the routers and other network devices don't have errors doesn't mean that the network is working as designed. It sometimes takes days or weeks to engage the right personnel from the network team AND convince them to look at a specific component in the network. Once this occurs, a resolution usually is reached quickly.

When severe problems occur, outages that last hours or days can cost hundred of thousands to millions of dollars. The best estimates (based on previous, similar incidents) suggest that it would have taken all day if not more than one day for the proper network personnel to engage in this issue. After all: their network was functioning without any errors or performance delays. They only support an IP and SNA WAN, the operation of the system support team's NetBIOS based servers is not under the network team's control. In this case: 12 sites would have been down for a sustained outage of a day or more! Yet, with the use of tracing tools the system support team was able to identify the failing component within an hour of the problem's first being reported. Since the team had concrete evidence of the exact component that was causing the failure, it was easy to get the network team engaged. The only question that had to be asked was: did the concentration router for these sites receive change control the previous night? If so, was any change implemented that could affect NetBIOS routing between downstream access routers? In fact this is exactly what had occurred and the change was immediately backed out.

How much would it cost your organization to have 12 sites down for a day? If you could implement tools that minimized these outages, would you? What if you found out that the total cost of the tools for your entire install base of 4000 sites was less than the cost of having a one-time outage of 12 sites for a day? The Golden Code Trace Suite is built and priced to meet this exact objective.

Trace Suite Description

There are three members of this suite of trace tools:

- Network Trace* for OS/2*
- Kernel Trace* for OS/2
- Trace Analyzer* (Java 1.3)

The first two products generate traces and the third product is used for analysis.

Network Trace provides a software implementation of a network probe. It leverages the NDIS architecture of OS/2 to provide a Sniffer*-compatible trace of all network frames.

Kernel Trace provides a probe which inserts itself into the OS/2 kernel at key tracepoints. It utilizes IBM's SES "hooks" to write trace data to a file when any of these tracepoints are encountered.

In designing these tools, a conscious decision was made to split the capture function from the detailed analysis function. This decision was reached because this product set was designed to meet the needs of enterprise customers. These customers will most likely want to take traces remotely, then analyze them from a central location. In most situations, an analysis engine/GUI is not necessary at the remote site where traces are generated. This allows the installation of relatively compact capture software at each remote site without the overhead of analysis software on these machines.

Another key design point was to keep the size and intrusiveness of the capture software to the absolute minimum necessary to gather the data required.

- Development and design efforts included extensive performance optimization and minimization of resource overhead.
- These products are designed to leverage standards and facilities that already exist.
- Command line and programmatic interfaces are preferred over GUI.

Both capture tools are designed to be run using remote access or automation technologies such as remote command execution, REXX or batch programs, telnet and IBM's Netfinity*.

Golden Code has developed a separate but related product, the Trace Analyzer, to decode and analyze the trace files created by the capture tools. The Trace Analyzer is written completely in Java and has been tested on several platforms. Currently, the Trace Analyzer provides decodes for many common network protocols and for OS/2 kernel tracepoints.

Where possible, the trace tools are developed in a platform independent manner. The Trace Analyzer, since it is written in Java, has no inherent platform dependencies. The capture agents (Network Trace and Kernel Trace) are platform specific, as they "hook" the system at a very low level. It is currently very rare for these low level interfaces to be platform independent or portable.

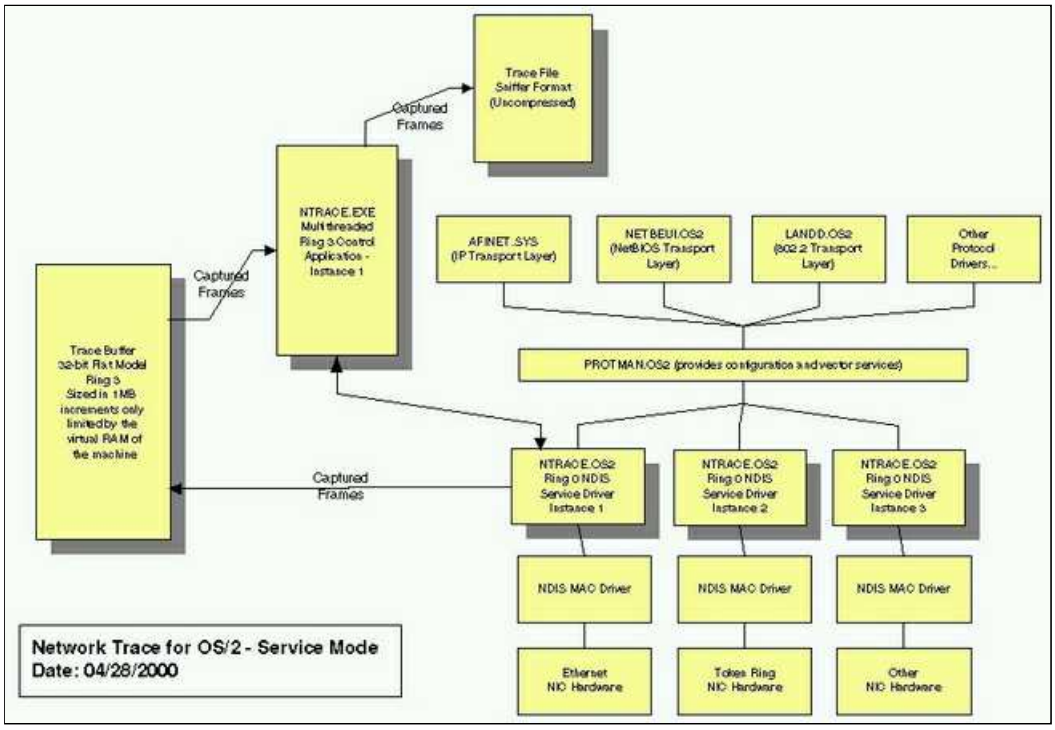
With Golden Code's focus, background and experience in the OS/2 environment, OS/2 was an obvious first target for capture agents. In addition, the OS/2 environment is very conscious of TCO and tools designed to minimize TCO are welcomed.

Network Trace for OS/2

Network Trace for OS/2 is the only software of its kind commercially available for OS/2 today. It is native OS/2 software which captures network packet data to a binary file for later analysis by a separate program. Aside from a running set of statistics about the traffic it is capturing, Network Trace is not meant to provide detailed analysis of the data.

Network Trace makes it easy to gather traces, whether on your local segment, or on the most remote segment of your WAN. You can now have the benefit of a hardware network probe on each of your network segments, but with a software-only implementation. You leverage the hardware you already have, and the software is deployed easily through standard software distribution.

The following diagram shows the high level design of Network Trace:



Network Trace leverages and integrates with the NDIS-based LAN subsystem of OS/2. In its most powerful mode, Network Trace is installed as a layer in the NDIS subsystem, allowing it to copy inbound or outbound network frames, while leaving the operation of the network itself unchanged. The ring 0 device driver(s) are always inserted into the NDIS environment but have been designed to minimize overhead, especially when tracing is not active. The difference in CPU and memory utilization cannot be measured using known methods.

A command line control program running at ring 3 manages the tracing process, the allocation/deallocation of the trace buffer and the writing of the trace file into a standard format.

Network Trace for OS/2 v1.0 was released on 01-29-2000. This product was fully functional, but had known limitations.

Highlights of v1.0 include:

- **Hardware-independence.** Network Trace for OS/2 can be used with a wide range of network hardware. Any NDIS 2.01 compliant MAC drivers can be used. Because it is a software-only implementation, it will work with any speed adapter, including 100 Mbps Ethernet.
- **Power.** Network Trace works reliably in stressful environments, tracing WorkSpace On-Demand* / RIPL boot storms of 50+ machines without dropping frames!
- **Versatility.** Its true benefits are realized when the software is deployed throughout the enterprise. Any remote OS/2 machine can produce traces, when accessed from a central location using existing remote management technologies, such as telnet or Netfinity* Manager.
- **Simplicity.** In order to support the widest range of customer environments, Network Trace has a command line interface. This simplifies remote use and enables automation via REXX. The trace files generated are Sniffer-compatible, making them readable by most existing network analyzer tools and with the Golden Code Trace Analyzer.
- **Low Overhead.** Network Trace has minimal impact on system resources when it is dormant: no more than 31KB RAM utilization and 0% CPU utilization (compared to <1% for v1.1 in "Service" mode).
- **Convenience.** Since it is an NDIS 2.01 module, Network Trace requires no dedicated hardware. No hardware key is necessary. In its standard mode of operation, it does not interfere with the normal network activity of the installed machine.

Network Trace for OS/2 v1.1 was released on 06-09-2000. This product resolved all major limitations in v1.0 and added new functionality:

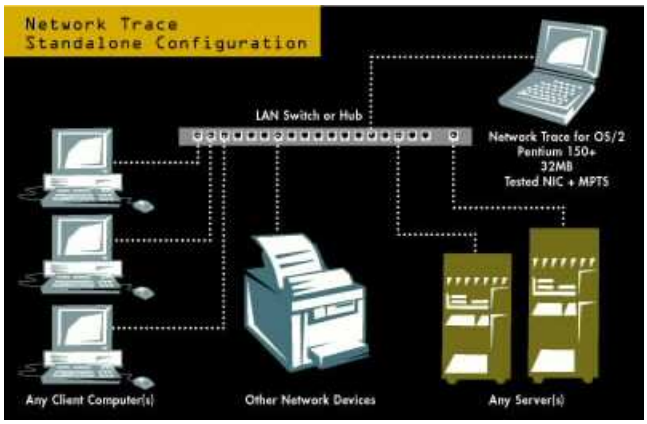
- **Service Mode.** Implementation of a new architecture, allowing much higher performance than the product's original design, now termed "Protocol" Mode. The result is the virtual elimination of frame loss, even during very high network utilization. Using service mode on a dedicated tracing machine, Golden Code has recorded data capture rates as high as 9MB per second, with no frame loss. In addition, this architecture allows Ethernet frames outbound from a tracing machine to be captured, removing a critical limitation of the original design.
- **Statistics Report.** While tracing, the some basic statistics relating to network utilization are displayed. Additional tools are provided for querying the state of network driver statistics and error counters.
- **Error Monitoring.** Key network events are logged as they occur, even while tracing is not active. Examples include the logging of soft and hard errors in a Token-Ring environment.
- **Additional Tested Hardware.** The new IBM 16/4 Token-Ring CardBus Adapter has been added to the list of tested network hardware.

With this addition, Network Trace for OS/2 can be used to take promiscuous traces of Token-Ring networks from a notebook computer. More recently, the family of Intel 82557, 82558, and 82559 Ethernet adapters has been certified as well.

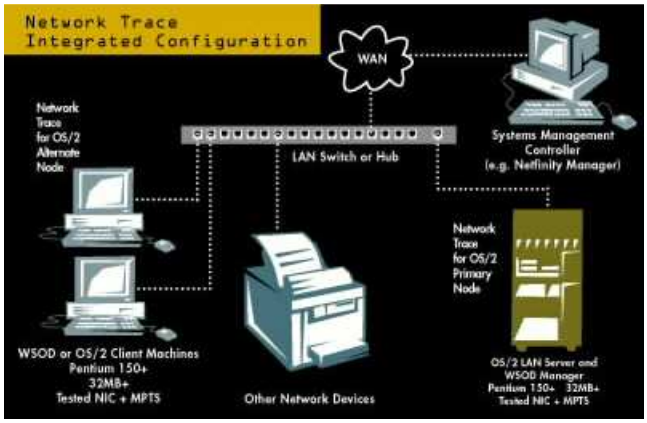
Installation scenarios:

There are two methods for installing Network Trace, "standalone" and "integrated".

The standalone method is optimal for consultants, application developers and those using tracing in a lab environment. It can be installed on a mobile system to allow it to be plugged in at any site that is visited. It does not inhibit the use of the network while tracing or otherwise, so the system does not have to be dedicated to only tracing. The following diagram shows this installation approach:



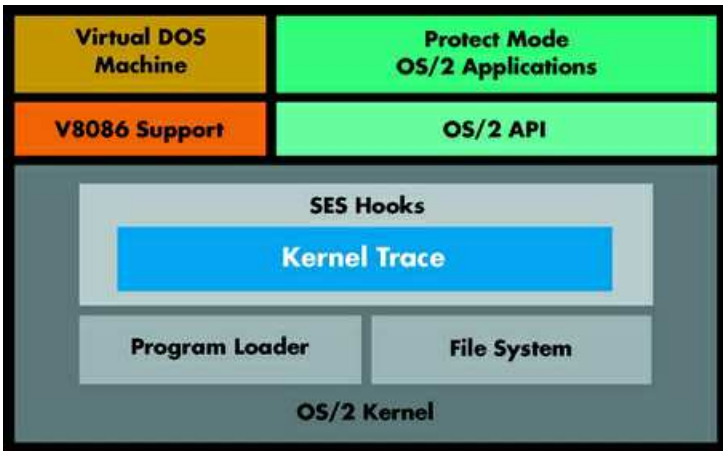
The integrated method is where an enterprise can significantly reduce TCO. By installing it as a software addition to systems that are already deployed and are already being managed/maintained, the cost of deployment is minimized. Since it has minimal impact on system resources and does not inhibit network operation, it is ideal for installing on all remote OS/2 servers in an enterprise. It is hardware independent and works with any NDIS 2.01 compatible MAC driver. In this approach, a trace is never more than a few minutes away. No complicated logistics, travel or other delays need be incurred. **This provides central support, development or engineering teams deep visibility into every LAN segment in the enterprise!** The following diagram shows this installation approach:



Kernel Trace for OS/2

Kernel Trace for OS/2 is software designed to reduce the cost of ownership of your OS/2 systems. It enables any OS/2 machine in your enterprise to generate a detailed trace of key events and requests made of the OS/2 kernel. This trace is written to a binary file for later analysis. Kernel Trace makes it easy to gather traces, whether on a local machine, or on the most remote machine in your enterprise. Its software-only implementation allows easy deployment through standard software distribution.

The following diagram shows the high level design of Kernel Trace:



Kernel Trace inserts itself in between all applications (ring 3) and their use of the OS/2 file system(s) and program loader. It utilizes IBM's Security Enabling System (SES) hooks to not only capture key kernel events but also to copy highly useful data associated with each event. This driver runs in ring 0 and when enabled, it writes each trace record directly to a file of the user's choice.

A ring 3 control application is provided to manage the tracing state and the options used by the device driver. This is a command line program, which simplifies remote use.

Highlights include:

- **Trace Kernel Events and Requests.** Kernel Trace provides visibility into key tracepoints occurring inside of the OS/2 kernel itself. The following tracepoints are recorded:
  - File Access (open, close, read, write, change pointer)
  - File Manipulation (delete, move)
  - File Information/Attributes (query file info, set file info, set file size, set file mode, set path info)
  - File Search (find first, find next, find close)
  - Directory Manipulation (make directory, change directory, remove directory)
  - Program Execution (execute program, loader open, get module, create VDM)
  - Set Date/Time
  - Direct Disk Access (DosDevIOCtl Categories 8 and 9)
- **Power.** Its design as a software-only implementation allows deployment throughout the enterprise. In this environment, any remote OS/2 machine can produce traces, when accessed from a central location using existing remote management technologies, such as telnet or Netfinity Manager.
- **Versatility.** In order to support the widest range of customer environments, Kernel Trace has a command line interface. This simplifies remote use and enables automation via REXX. All tracepoints from all processes can be traced, or the list of tracepoints or processes can be reduced to simplify analysis by using filters. The trace files generated are easily analyzed with the Golden Code Trace Analyzer.
- **Low Overhead.** Kernel Trace has minimal impact on system resources when it is dormant: negligible CPU utilization and less than 470KB RAM utilization.
- **Convenience.** Implemented 100% in software as a device driver using IBM's Security Enabling Services (SES), Kernel Trace does not require any additional hardware to operate. Since all trace data is written directly to a file, there is no need for a secondary machine connected via a null modem cable. While tracing, Kernel Trace does not interfere with the normal activity of the installed machine.

Kernel Trace v1.0 for OS/2 was developed and released to a limited customer set in the third quarter of 2000. Version 1.1 is generally available to the public as of 10-17-2001.

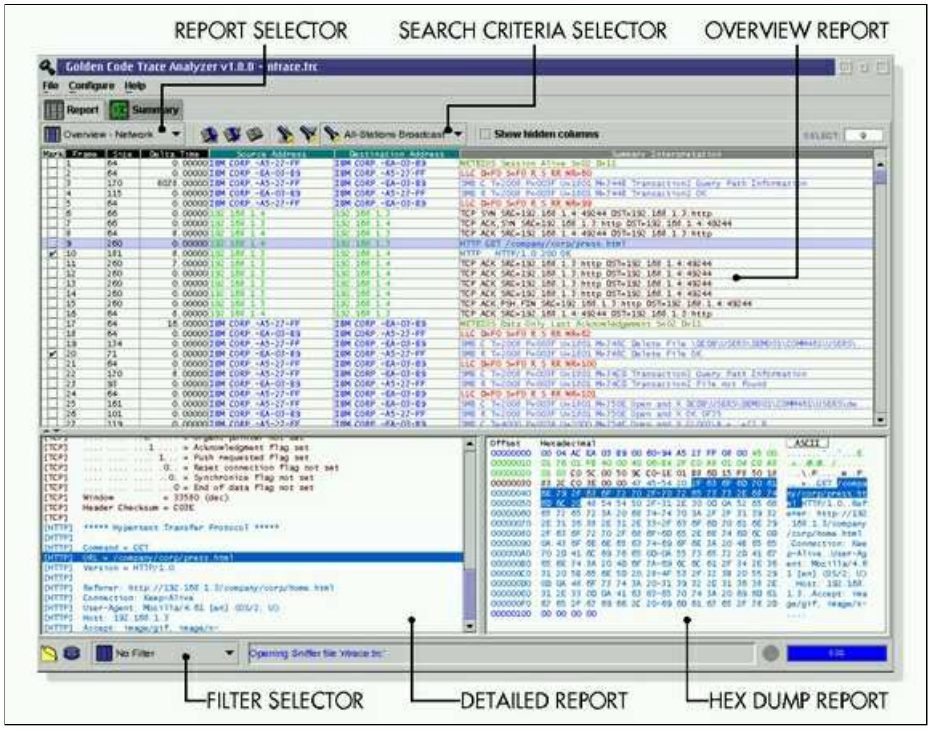
In September 2002, Golden Code Development announced the forthcoming release of Kernel Trace v2.0 for OS/2. Version 2.0 adds powerful new features which provide an even deeper view into OS/2, and a new mode of operation for enhanced usability. New features include:

- **Process Termination Tracing.** This new tracepoint reports exactly when a process is terminated and provides information associated with the terminating process.
- **Callgate Tracing.** Version 1.1 utilizes the subset of tracepoints explicitly exposed by OS/2's SES. Version 2.0 opens the information floodgates by enabling the tracing of *every entrypoint into the OS/2 kernel*, 16- and 32-bit! This means that in addition to the file system access and certain utility function tracepoints available in version 1.1, Kernel Trace now captures information on each call into the kernel for such services as:
  - inter-process communication;
  - memory management;
  - semaphore management;
  - thread management;
  - process management;
  - session management;
  - device driver services;

    - additional file system services;
    - exceptions and signals;
    - reliability, availability, serviceability functions.
- **In-Memory Tracing.** Version 1.1 writes its data directly to a file while tracing, which was found to be limiting in certain circumstances. To alleviate this problem, version 2.0 introduces an option to hold trace results in a circular memory buffer until such time as tracing completes through user intervention, or based upon settings defined at the command line or in a configuration file. The user now has the flexibility to select the tracing mode which is best suited for the task at hand.
- **Extended Command Line and Configuration File Interface.** To accomodate the larger number of features in version 2.0, the command line and configuration file interface to the Kernel Trace control program was vastly expanded. A great deal of flexibility is provided over the set of tracepoints and features to enable, allowing the user to fine tune the result set of each trace.

Trace Analyzer for Java

The Trace Analyzer is a general purpose, graphical analysis tool which decodes and analyzes record-oriented trace files. Version 1.0 was released 10-17-2001. Maintenance builds are released periodically. The following is a screen shot of the main user interface:



Highlights include:

- **Power.** Comprehensive protocol decodes and a host of useful, preconfigured reports and data filters make the Trace Analyzer a powerful analysis tool, even for the first-time user. For the more advanced user, the detailed structure of every protocol is exposed, enabling the creation of extremely robust reports and data filters. Symbolic name aliasing substitutes names of the user's choice for numeric addresses, to enable more meaningful and readable reports.
- **Flexibility.** With a customizable report and data filtering system, the Trace Analyzer allows a user to view trace data in completely new ways. No longer are analysts limited to a small set of immutable, preconfigured reports. If one of the standard reports does not meet an analyst's needs, it is easily enhanced or replaced. Robust data filters are easily created using a simple expression syntax and optional user functions, in order to distill specific information from a trace. The same filters are used for search functions, reducing the overall effort needed to use the system.
- **Versatility.** Written in 100% Java, the Trace Analyzer is designed to run on multiple platforms "out of the box", giving you deployment flexibility. Because it takes advantage of the cross-platform user interface library provided with Java 2, the user experience is consistent across platforms, reducing training time and cost.
- **Ease of Use.** The features of the Trace Analyzer, while powerful, were designed also to be accessible and easy to use. Reporting, filtering, and searching options are available directly from the main interface. New data filters can be generated automatically with a double-click of the mouse. Context-sensitive help is integrated into the application.

Product features include:

- **Support for Network and Non-Network Protocols.** Designed to be a general purpose analysis tool, the Trace Analyzer is not limited to network analysis. For example, in addition to broad network protocol support, v1.0 offers protocol support for Kernel Trace for OS/2.
- **Customizable Reporting System.** Overview Reports are completely configurable, giving the user unprecedented control over the

presentation and format of trace information. Existing reports can be changed or removed. New reports can be created to represent any combination of data from multiple protocols.
- **Flexible Data Filtering.** Using a straightforward and flexible expression syntax, the user can subset the data within a trace to include only those records which meet a defined set of criteria. Data filters can be created using an interactive user interface or can be automatically generated with a double click of the mouse.
- **Flexible Search Engine.** The same set of tools used for data filtering is used to provide a powerful search capability, requiring less of a learning curve.
- **Printing with Preview.** Reports can be printed in color or black and white. A useful print preview feature is provided to customize your print job and eliminate wasted paper.
- **Symbolic Name Aliasing.** Numerical addresses can be replaced with symbolic names meaningful to the user.
- **Unlimited Bookmarks.** Any number of individual records in a trace file may be bookmarked. Bookmarks can be used to create special report views and to define the contents of print jobs.
- **File Save and Export.** Any subset of trace file records can be saved in the format of the original file. Overview Report data can be exported for further manipulation as comma- or tab-delimited text.
- **Export and Import Tools.** Overview Report templates and data filters can be exported and imported for sharing or archival purposes.
- **Summary Information.** Useful summary information about a trace file is presented in its own report.
- **Configurable User Interface and Data Representation.** The user interface itself can be customized by the user, including the formats used to represent certain data types in reports.
- **Online Help.** Searchable, context sensitive, online help is provided. All help content is available in a separate, softcopy format for review outside of the Trace Analyzer program.

The Trace Analyzer was architected to be easily extended with plug-in file format modules, which can be added without requiring changes to the base software. The Trace Analyzer currently reads and writes the following trace file types:

- **Uncompressed Sniffer.** This popular file format is widely used throughout the industry by network capture software. Network Trace for OS/2 stores its trace data in this format.
- **DatagLANce*.** This file format was developed by IBM for use with its now defunct DatagLANce product. Withdrawn by IBM many years ago, DatagLANce was a network capture and analysis package for OS/2. The Trace Analyzer supports this format for backward compatibility.
- **Kernel Trace.** This is the file format used by Kernel Trace for OS/2.

In addition to the Kernel Trace protocol, the Trace Analyzer decodes many common network protocols. The Trace Analyzer's protocol parsing engine has been designed with flexibility and extensibility in mind; additional protocol support can be added to the product without requiring the software itself to be modified.

Protocol support for version 1.0 includes:

- **Ethernet/802.3**
- **Token Ring** (e.g., MAC, BPDU, SNAP, RI)
- **LLC**
- **RIPL**
- **IP Suite** (e.g., IP, TCP, UDP, ARP, RARP, ICMP, IGMP, SNMP, TFTP, FTP, Telnet, SMTP, DNS, TCPBEUI, HTTP, BOOTP, IGRP, RIP, LPR/LPD, POP3, IMAP, NNTP)
- **NetBIOS**
- **SMB**
- **SNA** (TH, RH, and most common RUs)
- **NPAP**
- **Sun* RPC, NFS, MOUNT** (partial)
- **Novell NetWare*** (partial)
- **Kernel Trace for OS/2, including an expanded set of callgate decodes**

Note that support for some protocols is partial; Golden Code recommends that customers take advantage of its free evaluation policy before licensing, to ensure this product meets their analysis needs. Golden Code will continue to aggressively expand the Trace Analyzer's protocol and file format support.

Trace Suite Uses

Both Network Trace and Kernel Trace can be used for similar "applications". The key difference is in their positioning within the PC or LAN system.

The following list of uses are the most common:

- **Problem Analysis.** Information relating to the internal operation of systems and networks can now easily be captured and analyzed. The Trace Suite improves problem determination by providing a simple and cost effective way to take a trace, locally or remotely, from any machine in the network. Traces simultaneously generated from multiple locations can be analyzed and compared to help isolate and diagnose problems.
- **Application Development.** The Trace Suite provides a cost effective mechanism to reduce application development time. Network traffic

between an application server and its client can be traced to provide insight into problems. Kernel events can be traced to provide insight into the operation of a system. While a developer can often track down problems with the addition of debugging or logging code, Kernel Trace provides a tool that allows inspection of highly useful information that can't always be captured within the application itself. It provides this capability without requiring any additional code to be written.

- **Performance Tuning.** Traces can be taken to create a baseline for the performance of a network or a system. Additional traces can be taken at any time for comparison with the baseline. Analysis of these traces may be used to determine whether an application performance problem is network-wide, isolated to a specific machine or application specific. The Trace Suite allows performance issues to be addressed proactively, before problems become overwhelming.
- **Network Health Check.** Traces previously too costly and time consuming to take on a regular basis are now possible, enabling the network administrator to take a more proactive approach to network health. Using REXX, automated traces can be scheduled periodically for each network segment as a preventative maintenance measure.
- **Profiling.** Profiling is the process by which the operations of a system are recorded during active use and then the resulting data is analyzed to understand system internals. The Trace Suite can be used to profile client-server applications and network devices. Even the operating system can be profiled. For example one may determine:
    - Network Session Flows
    - File System Access and Flows
    - Error Conditions

Trace Suite Benefits

The Golden Code Trace Suite is designed to reduce the TCO of OS/2 and WorkSpace On-Demand environments.

By minimizing the time required to gather critical data when a problem occurs, the time to identify and solve the problem is reduced. The capability to gather trace data remotely eliminates the wasted time, effort, and travel expense associated with sending engineers on site to gather traces. A problem's impact, such as downtime, is reduced by days or even weeks.

Technical staff are more productive and effective. Network or system specialists can now remain in a central location and focus on trace analysis rather than trace gathering, permitting a small team to manage a greater number of systems.

With trace capability available at every OS/2 machine on the network, higher levels of service are possible. Tasks previously considered impractical - performance benchmarking, network health checks, profiling - are now within reach.

Both capture products are architected to minimize deployment and ownership costs. They require no specialized hardware, eliminating the cost of purchasing, deploying, and managing such systems over time. Instead, they "piggyback" on the management infrastructure already in place and are installed through traditional change control processes.

The Trace Analyzer provides a cross-platform, highly customizable and powerful analysis system, which enables new ways of deriving important information from trace data.

Volume Pricing Examples

Pricing Example - 50 Sites

Assumptions	Value
Number of Sites (Remote Servers)	50
Average Number of Clients per Site	10
Total Number of Clients	500
Total Number of Systems	550

Product	Single Unit Price	Discount	Discounted Price	Quantity	Extended Price
Network Trace for OS/2 v1.1	$500.00	21.00%	$395.00	50	$19,750.00
Kernel Trace for OS/2 v1.1	$200.00	70.00%	$60.00	550	$33,000.00
Trace Analyzer v1.0	$1,500.00	0.00%	$1,500.00	5	$7,500.00
Total					$60,250.00
Per Site Cost					$1,205.00

Pricing Example - 100 Sites

Assumptions	Value
Number of Sites (Remote Servers)	100
Average Number of Clients per Site	10
Total Number of Clients	1000
Total Number of Systems	1100

Product	Single Unit Price	Discount	Discounted Price	Quantity	Extended Price
Network Trace for OS/2 v1.1	$500.00	28.00%	$360.00	100	$36,000.00
Kernel Trace for OS/2 v1.1	$200.00	75.00%	$50.00	1100	$55,000.00
Trace Analyzer v1.0	$1,500.00	6.67%	$1,400.00	10	$14,000.00
Total					$105,000.00
Per Site Cost					$1,050.00

Notes for Pricing Examples

1. The Trace Analyzer is not deployed in remote sites, but rather used at a central site by support staff.
2. Each site is assumed to have one OS/2 server.
3. Quoted prices do not include shipping, handling, or tax.
4. Pricing provided for each product is current as of October 5, 2001, and is subject to change without notice.

About Golden Code Development

As a consulting firm and independent software developer, Golden Code Development Corporation helps its clients design, build, and manage mission critical, networked computing environments. The company specializes in technologies and techniques which enable the creation of highly managed systems with exceptionally low cost of ownership. Golden Code's core competencies include OS/2, Java, and Server-Managed Client solutions, such as IBM's WorkSpace On-Demand. Its expertise in these areas, combined with a disciplined design and implementation methodology, make Golden Code an ideal technology partner for the enterprise customer.

*Trademarks

Golden Code, Network Trace, Kernel Trace, and Trace Analyzer are trademarks of Golden Code Development Corporation.
IBM, OS/2, and Netfinity are registered trademarks of International Business Machines Corporation.
DB2, MQSeries, and WorkSpace On-Demand are trademarks of International Business Machines Corporation.
Java is a trademark of Sun Microsystems Corporation.
Sniffer is a trademark of Network Associates, Inc.
Other product names referenced herein are the property of their respective owners.