Interception Technology Enhances Application Functionality

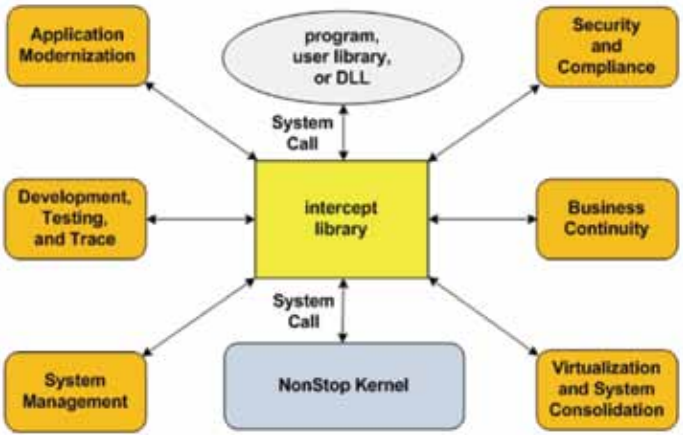
Jack Di Giacomo
President
TANDsoft, Inc.

Introduction

Companies make massive investments in both third-party and in-house applications in order to operate their mission-critical NonStop environments. However, those applications come with limits to their functionality – limits that often cannot be altered without prohibitively expensive, system-wide modifications. How does a company address application enhancement requirements? Interception technology is one solution. Interception technology enables the capture of a variety of system and database calls in order to enhance application behaviors. The result is the creation of new application functionality without the need to make source code changes, of particular importance when the source code is unavailable. *"Interception Technology Enhances Application Functionality"* will define interception technology and its benefits, the architecture of interception, the origin of NonStop interception, and the areas in which interception technology is being employed today. They include application modernization and development, system consolidation, security, business continuity, and PCI compliance.

Let's Define Interception Technology

With help from Wikipedia, let's define interception technology as it applies to computer programming. Interception technology covers a range of techniques that can be used to alter or augment the behaviors of applications, operating systems, or other software executables by intercepting function calls or system calls passed between computer components. The code that handles intercepted function calls, system calls, events, or messages most commonly is called a "hook." In the NonStop world, a hook is known as an intercept library.



An intercept library, or hook, sits between an operating system and a program, a user library, or a dynamic-link library (DLL). As a process carries out a function (read / insert / update / delete), the attached intercept library captures the system call, modifies the call to include whatever new functionality is intended, and sends the modified call to the operating system, represented here by the NonStop Kernel. When the intercept library receives a response from the operating system, it sends the modified response to the process. At no time is the process aware that any modifications were made to its original system call.

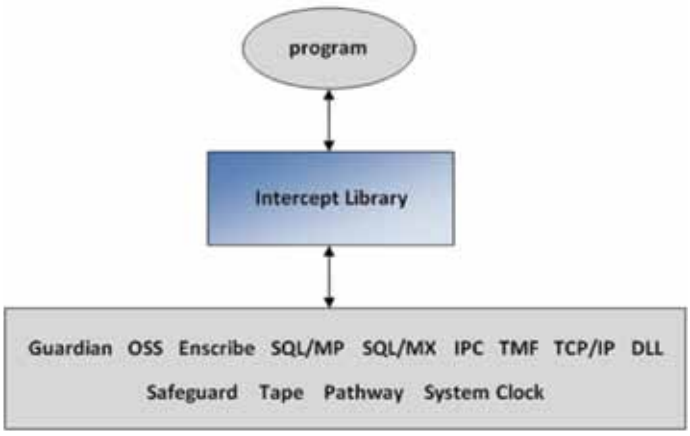
Intercept libraries can be used for many purposes – debugging, tracing, application monitoring, statistics collection, testing, replication and synchronization, encryption and tokenization, virtualization, etc. All are examples of enhancing an application by extending its functionality beyond its original purpose and beyond the limitations of its source code.

The Very Best Thing about Interception…..

….is that no source code changes are required. No program recompiling is required. In fact, with interception technology, you don't even need access to the source code. If you bought software from a third-party provider, you don't need the source code. If it's HP code, you don't need the source code. Even if you wrote the application in-house and have the source code, you don't need the source code. The program logic does not change because an intercept library linked to an executable works seamlessly at the machine layer, thereby eliminating the necessity to reimplement the function for every program. An intercept library can be written in TAL and linked to a program written in C, C++, Java, or COBOL. Language standardization is not an issue. Imagine the savings in development, testing, and support-related costs when interception is used to augment program behaviors.

The Architecture of Interception

As defined earlier, interception technology enables the capture of a variety of system and database API calls in order to enhance application behaviors. Interception takes place at the native API



level, so any program written in any language can interact with the operating system without the need to alter the original program logic. For instance, programs call NSK (NonStop Kernel) system procedures to carry out specific functions, to access databases, and to utilize system resources. Cases in point - database I/O, IPC, communications, security capabilities, and current time. An intercept library is able to seamlessly capture, monitor, and modify these interactions with no need for program changes.

Intercept libraries are programs themselves –specialized layers of code that can be user libraries or dynamic link libraries (DLL). In the NonStop world, the intercept library captures Guardian or OSS calls from programs, user libraries, or DLLs [private, public (licensed and unlicensed)]. NonStop-specific intercept libraries typically support TNS, TNS/R, and TNS/E environments – file codes 100, 700, and 800, interpreted code – accelerated and native. Intercept libraries can perform both capture and logging.

The figure here is a simplified example of an intercept library. An application process or program provides services to its end users. As part of its operation, the application process typically makes several calls to operating-system services. These calls, for instance, may be requests to read an incoming message, to generate an outgoing message, or to manipulate a database.

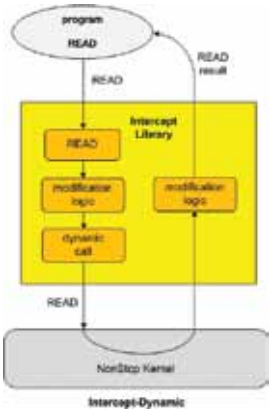
The intercept library positions itself between the application process and the operating system in such a way as to capture calls made to the operating system. Rather than the operating-system function being invoked on a call by the application, the intercept library is invoked instead. Thus, when the application makes an operating-system call, e.g., to read an incoming message or to issue an update to a database, the intercept library, not the operating system, is invoked and processes the call. The intercept library can modify the execution of the operating-system function in any way that it has been programmed to do. The response to the modified operating-system call is then returned to the application process. The entire operation is so seamless that the application process has no idea that its call was intercepted and modified.

For instance, an intercept library might intercept all interprocess messages being sent to other applications to add information to those messages for enhanced processing. Alternatively, an intercept library might capture database calls (read, insert, update, delete) to make decisions about the validity of the updates, to modify the updates according to application rules, to record the update activity, or to make other updates such as to a change log. For events or transactions, the intercept library might invoke other application services such as security monitoring and authorization.

A closer look at intercept libraries reveals two common interception methods – Intercept-Dynamic, which occurs during run time, and Intercept-Retarget, which also takes place during run time except that it must be "prepared" prior to execution. Although compiled differently, they nevertheless perform the same functions.

Intercept-Dynamic

A dynamically loaded intercept library (DLL) is not compiled as part of any application but instead is compiled separately. It is linked, or inserted, into the application program only when the application program is run as a process. Therefore, it can be changed easily without having to rebuild the application. This very flexible DLL also can be shared by other processes.



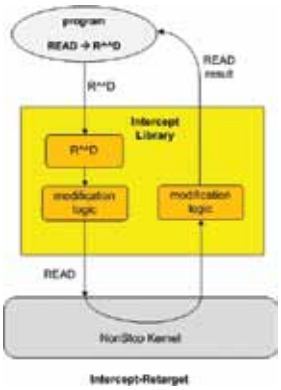
Intercept-Dynamic

In the Intercept-Dynamic figure, a program invokes a system procedure READ and requests that data from the NonStop Kernel be sent to the program. Instead of the execution being transferred immediately to the system procedure READ, the intercept library captures the call via its own READ procedure, makes any call modifications that have been requested, then dynamically calls the system procedure READ. Dynamic calls are used to locate and navigate to the physical address of the system READ procedure. In the absence of dynamic calls, an intercept library will only READ itself (a recursive call). The system READ carries out the requested function, and the call then returns via the intercept library. At this point, the intercept library may make more modifications before it returns the requested data, modifications included, to the program, thereby completing the original system call.

It is important to highlight that modifications can be made from the program to the operating system, from the operating system to the program, or to and from both during the same process.

Intercept-Retarget

An Intercept-Retarget intercept library is also separately compiled and is linked to the application program at the time of execution. However, when an Intercept-Retarget intercept library is used, the application program must be "prepared" in advance with a step that is performed only once prior to execution. Operating-system calls made by the program are 'retargeted" to the procedures in the intercept library.



Intercept-Retarget

In the Intercept-Retarget figure, a program is "prepared" by assigning it an intercept library before any calls are executed. The program's procedure calls are renamed to the intercept library's equivalents. For example, calls to system procedure READ are changed to R^^D. Next, the prepared program and its linked intercept library are simultaneously executed. The intercept code, also named R^^D, is invoked instead of the system procedure READ. The intercept library linked into the program captures the call, makes modifications, then directly calls the system procedure READ. There is no need to make a dynamic call because the system READ procedure is clearly different than the intercept library's R^^D procedure. Upon the call's return, the intercept library may make more modifications to the API contents before it returns the requested data to the program, thereby completing the original system call.

Although not as flexible as Intercept-Dynamic, Intercept-Retarget produces less overhead because the direct system API call is faster than the dynamic system API call.

NonStop Interception Began with Tracing

Tracing is a specialized use of logging to record information about a program's execution. Real-time debugging and diagnostic analyses are common examples of trace utilities. Tandem old-timers like this author remember the days when interception technology was limited to communications-based tracing. Internetwork and cross-platform communication was and still remains highly complex from a development standpoint. Remember TIL (Tandem-to-IBM Link)? TIL opened a data path between one NonStop or a network of NonStops and an IBM 370 or compatible system equipped with IBM's 3803-1 or 3803-2. Remember CUP, CMI, ptrace? Those early trace utilities eased the complexity by capturing, or intercepting, the communication protocol inputs and outputs between the NonStops and other NonStops, between NonStops and IBMS for debugging and diagnostic purposes. Even then, trace tools possessed the same powerful benefit as their interception descendants. Source code modifications were not and are not required.

It took a few years for trace capability to evolve from communications-based solutions to the broader spectrum of process-trace. CUP, CMI, and ptrace were Tandem-developed tools. It was third-party providers, many of whom began their careers as Tandem developers, who expanded the possibilities of using interception technology with other applications and for purposes beyond debugging and diagnostics. Modifying database input/output, setting breakpoints on NonStop system procedures, process-to-process migration and interaction, API interception in virtual environments, detecting and resolving application deadlocks, educational tools to document program behavior and data flow, and the list goes on.

Encryption and decryption for data in-flight and at-rest are one of the latest uses for interception technology, and future solutions are only as far away as are the problems they need to resolve.

How NonStop Customers Use Intercept Technology

NonStop customers employ interception in a variety of ways to extend the functionality of their applications.

Application Modernization

Interception technology can provide automatic TMF protection of non-audited Enscribe files. Added TMF protection facilitates synchronization of DR databases. Interception also can enable the conversion of Enscribe files to SQL tables (Enscribe OPEN, READ, WRITE converts to EXEC SQL OPEN, FETCH, INSERT) in order to enable access to the database for OPEN tools.

Development, Testing, and Trace

List program system procedure calls made to the NonStop Kernel. Identify deadlocks and program sequencing errors. Monitor process stack usage to avoid abends caused by stack overflow. Learn programming techniques (NonStop, Enscribe, multi-threading, Nowait I/O, IPC, TMF, Startup, DEFINEs, malloc). Add Enscribe file-format modifications without the need to reprogram. When Enscribe database field formats change, modify only those programs that use the new format, not ALL programs that access the new format. *Example - date format changes.*

System Management

Files purged or deleted accidentally can be recovered from a recycle bin. Scripts can be executed upon process termination. Low-pin resources can be optimized across all CPUs. Workloads can be balanced between CPUs and disks.

Security and Compliance

Use interception technology to protect sensitive data at rest (Enscribe, SQL/MP) or in transit. Enforce security policies for authorization, authentication, and password changes. Replace sensitive data, such as Primary Account Numbers (PANs) or Personally Identifiable Information (PII) with powerful tokens or format-preserving encryption (FPE). Interception helps companies achieve industry compliance (PCI 3.4, SOX, HIPAA).

Business Continuity

Replicate Enscribe, SQL/MP, and SQL/MX DDL changes to a backup site. Automatic TMF protection of Enscribe files – *Insert TMF transactions BEGINTRANSACTION, ENDTRANSACTION.* Replicate Enscribe unaudited files or Enscribe file modifications to a backup site.

Virtualization and System Consolidation

Interception technology plays a valuable role in system consolidation. Time-Zone simulation - allows Guardian and OSS applications to operate within any virtual time zone. System clock simulation – allows Guardian and OSS applications to operate with any virtual system clock or current time value.

Don't Be Scared of Interception Technology

Ignore whatever it is you have heard about the challenges of interception technology. Intercept libraries are not hard to manage. They are not difficult to install. They do not create a massive increase in overhead, and they do not slow down your system.

Yes, there may be some increase in overhead. But today's faster processors minimize to almost nothing any potential impact.

And yes, interception can be challenging, in part because it is so different for TNS, TNS/R, and TNS/E environments. However, utilities exist to simplify the tasks of associating, managing, and removing intercept libraries from programs.

For example,

- Installing, or associating, intercept libraries to qualified programs automatically whenever programs are recompliled. *ADD or Remove Library for all programs on $data.servobj.* where file code 100 and owner = qc.**
- Using intercept libraries to identify programs and processes. *List all processes running within a specific Virtual Time-Zone.*
- Removing intercept libraries from qualified programs.

Many great interception solutions are offered by NonStop partners who specialize in interception technology. They provide excellent alternatives to building intercept libraries in-house.

Summary

Interception technology once existed only in the realm of communications-based tracing. Today, it is a powerful means by which to enhance the behaviors of mission-critical applications without the need to make costly, time-consuming modifications or complete recompiles. Interception technology never alters original application logic, does not require source-code access, and is language-ambivalent because it works at the native API level. Intercept libraries are not hard to manage. They are not difficult to install. They do not create a massive increase in overhead, and they do not slow down your system. NonStop customers take advantage of existing interception utilities in a number of NonStop environments and for a variety of functions. The technology is so flexible that it will continue to be the solution of choice for technical challenges that arise in the future.

TANDsoft Inc.

TANDsoft Inc. specializes in interception technology. Company products include the OPTA2000 virtual clock- and time-zone simulator; FileSync for automatic file synchronization, replication, and data deduplication; the OPTA suite of interception and trace utilities (OPTA¬Trace Online Process Tracer and Analyzer, Recycle Bin, EMS Alerts Online Startup and Termination Capture Utility, Low Pin Optimizer); Stack Monitor, which alerts developers to the impending threat of a stack overflow; Command Stream Replicator, which logs and automatically replicates TMF-audited/unaudited FUP, SQL/MP and SQL/MX DDL structure and other environment changes to target systems; AutoLib, which automatically loads a user library or a DLL for executing processes; the Enscribe-2-SQL and TMF-Audit Toolkits and the new Enscribe-2-SQL Data Replicator, all of which offer flexible, affordable alternatives to more expensive conversion products or manual conversion techniques; E2S-Lite, which permits efficient, low-cost Enscribe modifications without the need to change a program's source code; and SDI (Sensitive Data Intercept) for Enscribe and SQL/MP.

Jack Di Giacomo has 30 years of experience in the design, development and support of NonStop software solutions. He is the president of TANDsoft, Inc., a company specializing in interception technology. TANDsoft delivers quality interception solutions for virtual clock- and time-zone simulation (OPTA2000), application modernization (Enscribe-2-SQL and TMF-Audit Toolkits and Command Stream Replicator), business continuity (FileSync), and sensitive data intercept (SDI). Contact Jack at jack.digiacomo@tandsoft.com or at 514-695-2234.