

The Value of TSO in Today's Computing Environment

Abstract

The use of TSO as a tool for supporting the modern mainframe (z/OS) environment has decreased in recent years and both IBM and ISVs have let their development of new TSO based tools and support for existing tools atrophy, favoring distributed computing platforms and web access. While seemingly cost-effective, this trend has not led to increased productivity of the IT Professionals tasked with the support of the mainframe commensurate with the cost. Learn how enhancing the TSO environment on the mainframe can lead to improved productivity.

Overview

For some years now, there's been a migration of Information Technology support away from the mainframe, even where the mainframe remains a critical aspect of the IT environment. Companies that still believe in the value of the mainframe and the z/OS environment for hosting their most critical business data processing needs, don't necessarily believe in using the mainframe for the development and support of itself.

The value of the mainframe to the companies that still use them is in their power and reliability in providing high volume, stable, and secure processing of "transactions" designed to support their business, be it the internal needs of its employees or those of their customers. And this is as it should be. But what is being overlooked in the equation of mainframe computing is its value to the IT professionals whose jobs are to maintain it.

In an attempt to maximize "production" mainframe capacity, IT management attempts to minimize mainframe support "overhead." Moving the computing needs of the IT professionals off of the "expensive" mainframe and onto "inexpensive" PCs and Workstations is viewed as being an effective method of reducing the cost of a company's IT infrastructure. And to a degree, it is.

But does moving IT professionals off of the mainframe make them more productive? In a word, No! Especially when their jobs are to support the mainframe environment. The farther removed the IT Professional is away from the environment they support, the less effective they are.

Who Supports the Mainframe?

In examining the mainframe environment, it is obvious that some mainframe resources must be used to support it. Typically, these resources are used by the Systems Programming Staff. Their use of the mainframe is viewed as a "necessary evil" to keep it up and running. But what about the Operations Staff, the Production Control Group, the Help Desk, Security Administrators, DBAs, and Applications Developers. All have need to use the mainframe to perform their jobs.

With the days of keypunches and card readers long gone away, access to the mainframe world is through interactive terminals. For the mainframe "End User," access is typically through a "transaction manager," such as CICS, or using a Web Browser to access a Web Server that acts as the transaction manager. For the End User, hiding what's under the covers has no impact on

The Value of TSO in Today's Computing Environment

their computing experience. And, with modern computing tools and graphical access, hiding the actual data processing infrastructure improves their experience.

But what about the IT Professional whose job it is to create and maintain the mainframe environment? What do they use? TSO, of course!

However, using TSO uses capacity that could seemingly be better utilized running production. Using TSO requires greater processor size, resulting in larger software charges. Using TSO means not using the latest, greatest graphical tools. And so on. These are the reasons that IT Management use to force mainframe support professionals off of TSO. Or, worse yet, they waste money on the attempt to "webify" TSO. All in the name of reduced cost and increased productivity. And with just the opposite effect!

System Programmers are not likely to abandon TSO for other environments. Their work is so close to the Operating System and its environment, it is their "tool" of choice. (After all, when was the last time they used a keypunch, card reader, and the IEBUPDTE Utility to update a PARMLIB Member?) And tools like HCD and WLM configuration are ISPF Dialogs that require TSO.

So what about the Operations Staff? They primarily use Consoles. But most Operators probably also use SDSF or a similar product that run under TSO. And Tape Librarians are likely to use TSO to query and manage the Tape Library (CA-1, RMM, etc.). And most installation probably have a variety of support and automation tools written as ISPF Dialogs, CLISTS, and REXX EXECs.

Production Control, Processing Services, or whatever it might be called, probably use a scheduling tool like CA-7, which has its own interactive interface (usually a 3270 full screen interface). But like Operations, they, too, probably use SDSF (or the like) and other installation developed tools that require TSO. So, like Operations, they, too, need TSO.

Then there's the Help Desk. Nowadays, the most frequent calls that most Help Desks receive are probably related to Desktop (e.g., Windows applications) or Network problems, not the mainframe. But when a problem call involves a mainframe application, what does the Help Desk use to determine the status of the application and related subsystems on the mainframe? TSO!

Security Administrators that manage the mainframe security environment need TSO to manage it. Even with cross-platform synchronization and management tools, it is important that the administrator have (and understand) native access. Most installations aren't likely to want their mainframe to be hobbled because of a network or security server problem.

Data Base Administrators that manage mainframe Data Base and Transaction Management Systems, be they CICS, IMS, DB2, Oracle, IDMS, ADABAS, or whatever else there is, need to make use of many tools on the mainframe, mainly interactively. Even Batch Jobs and Started Tasks are likely to be initiated or managed from TSO. When any type of recovery, reorganization, or reconfiguration is needed, being close to the system is critical.

The Value of TSO in Today's Computing Environment

And what about the application programmers? These are the IT Professionals that are likely to be the first "pushed off" the mainframe to use "modern" development tools. So they develop applications, most likely that make use of the mainframe Data Base Systems, in a foreign environment. But the work has to shipped to the mainframe for compilation, testing, and eventually production. Sure, work can be FTPed back and forth, but the farther the developer is away from the platform, the more difficult it becomes for them to deal with it, even with all sorts of fancy graphical software development and management tools. So mainframe application developers are likely to make use of TSO, too.

The Professionals that support the mainframe are typically the most highly skilled and highly paid employees that a Data Center has. So they should have the best tools available to allow them to do their jobs. And TSO is one of the most important tools they have.

And while TSO is most valuable and critical to the IT Professionals, there are many things that can be done in TSO that the end user might use. TSO isn't likely to be used for any customer access applications, but simple in-house (i.e., company employee) applications are candidates for this environment. In many cases, it can be more cost-effective to develop and deploy an application using TSO, ISPF, and REXX EXECs than developing an equivalent Windows or Web based application. Even using it to develop a "proof of concept" prototype application can be beneficial to the design process.

How TSO is Viewed in Today's Computing Environment

Many people view TSO as antiquated, clumsy, stale, and restrictive. We'd like to dispel these notions.

Years ago, a coworker made the statement: "TSO may be slow, but it makes up for it by being hard to use." While it was somewhat true back then, it certainly isn't now. TSO is no more difficult to use than any other interactive computing environment. In reality, TSO is much easier to use because it has a consistent set of rules of Command and Parameter Syntax.

What many people view as the greatest weakness of TSO is its non-GUI interface. It seems that many vendors are trying to "webify" or otherwise cover over the TSO 3270 display with a GUI. Why? The 3270 display is excellent (note that we did not say PERFECT) display of text data. When dealing with the typical TSO environment (in fact, almost all human to computer interactions), everything displayed and entered is TEXT. Why waste screen real estate with icons and graphics? With the graying of the Mainframe IT Professional, wasting screen space with useless graphics just contributes to eye fatigue. And other than selecting text for a cut/paste using the PC's clipboard, the mouse is largely useless. Every time a hand has to move from the keyboard to the mouse, a user loses productivity.

Systems Programmers, Application Programmers, Data Base Administrations, Help Desk personnel, Operators, Operations Analysts, Production Control personnel, Security Administrators, and other support staff need to deal with all of the paraphernalia necessary to support the system. Be it Data Sets, JCL, programs, parameter statements, security parameters, logs, or even dumps, ALL of the data they deal with is TEXT. Graphics add nothing but wasted

The Value of TSO in Today's Computing Environment

display space. For the z/OS environment, TSO and the 3270 interface does this better than anything else around.

In reality, which interactive computing environment is best is almost totally subjective. Most people (and we include ourselves in that) think that the environment that they know best (and presumably have used the most) is the best. When faced with learning a new environment, they almost always find fault with how things work differently from that they are used to. Suppose that different car models arranged the floor mats differently. We would be uncomfortable using a car different from the one we most frequently use. The same applies to which side of the road we normally drive on. Aside from the potential safety issues, interactive computing environments are no different. Some people are very adept at switching between environments at will. Some don't. Most are somewhere in between.

Of course, even support personnel that support z/OS exclusively use more than a single interactive environment. Even with TSO as the primary interface to the Operating System, many have to deal with Consoles, Netview, Scheduling Packages, Performance Monitors, and the like. But all of this still doesn't diminish the power, flexibility, and capability of TSO.

The Maturity of TSO

TSO has been around for 30 years. It has matured from an totally line mode add-on to MVT which didn't work all that well, to being a fully integrated interactive environment with full-screen access. But at the heart of it, unlike so many interactive computing environments that have come (and in some cases, gone) since, the basic architecture has remained intact.

Underlying any interactive computing environment used to support and maintain an Operating System and its applications, are commands that do "something." Typically overlaying that is some type of scripting language that can be used to combine commands and functions together. And then, of course, there is some type of full screen, fill in the blanks, selection, scrolling type of interface.

With TSO, IBM designed a command language and syntax, that, with one exception, has been consistent since the very beginning of TSO. This syntax was uniform in the commands that IBM supplied, those that vendors added, the ones that installations developed on their own, and those that were freely traded between installation (the original shareware that PC people seem to think they invented).

So what gives TSO this uniformity in command syntax? It is the TSO Parse Service. As much as both programmers and end users sometime curse it, TSO Parse gives TSO a predictable behavior. The use of TSO Parse by TSO commands means that all commands, regardless of their function, can have a consistent syntax in their parameters. Parameter punctuation is uniform. Keyword abbreviations are anything that make it unique. Prompting for correction, while being at the discretion of the user, is consistent. The original TSO CLIST language also followed (for the most part) the TSO Parsing rules, making CLISTs look to the user as just another command.

The biggest exception to the command uniformity of TSO was one of the two greatest additions to the TSO environment since its inception: REXX. (The other addition we consider in

The Value of TSO in Today's Computing Environment

this category is ISPF.) REXX, in the name of uniformity across the multiple platforms in which it was implemented, chose not to implement TSO Parse type parameters. Instead, REXX has its own parse capability which is purely positional and not keyword oriented. REXX is far more powerful than CLIST processing, so the lack of a TSO Parse facility within REXX is a sore point with a lot of MVS programmers. REXX, when used in TSO, has multiple MVS and TSO specific facilities - address TSO, address LINKMVS, address ATTCHMVS, address ISPEXEC - to name a few. So why didn't it provide an "address TSOPARSE" facility? (As a vendor of TSO based products, we've wondered if there is a market for such an add-on.) It didn't, so some TSO command scripts have been implemented using an "outer" CLIST and an inner REXX EXEC. Not the most elegant solution, but it preserves the TSO syntax uniformity.

Still, in other command environments, such as DOS and UNIX, parameter syntax varies from command to command. Most parameters are prefixed by a delimiter character (usually a dash) and are a single letter. In some environments, parameters are case sensitive, such that "-t" and "-T" can mean DRASTICALLY different things. And sometimes a space is required between that parameter letter and the parameter value and sometimes not. Sometimes the order in which the keywords are specified is significant. (Usually not, but still....)

Interactive computing has moved beyond the "line mode" command environment and into full screen processing. Displaying a screen with a list of choices, a fill in the blanks and over type capability, allowing multiple lines of data to be entered with one interaction, cursor position sensitivity, etc., all provide a major step in computing productivity. The 3270, as simple as it was when first introduced, provided a powerful full screen environment (the original GUI such as it was). Enhancements and additions to the 3270 architecture, such as color, extended attributes, and larger and multiple screen sizes, have been added with full compatibility with the original architecture (a hallmark of the 360/370/390/z computing environment). The introduction of ISPF, with its panel definition capability (whether creating straight panels or using the Dialog Tag Language), has removed the need for most full screen application programmers from even needing to know all the screen addressing and attributes rules and commands. And for those of us that are wedded to the minutiae of low level programming, it can still be done.

There have been a lot of posts on the IBM-MAIN List Server decrying the need for a GUI for IPCS, a GUI of SMP/E, a GUI for this, a GUI for that, ad nauseam, all independent of TSO. Why? What for? To what end? TSO based tools such as IPCS and SMP/E are used a relatively small percentage of the time. They all conform to the 3270 GUI and ISPF Dialog rules. What is in it for IBM (or any other vendor for that matter) to expend the resources necessary to develop a unique graphical environment for each application for the small incremental increase in IT Professional productivity they would provide?

So are we contending that TSO and the 3270 interface is near ideal as is and there is no need for substantial enhancements? Right? WRONG! NO WAY! TSO can still benefit from IBM and ISV enhancements.

The Benign Neglect of TSO

The Value of TSO in Today's Computing Environment

IBM seems to have given up on enhancing TSO in favor of various UNIX System Services shells. Yet for IT Professionals that are responsible for working on and maintaining the mainframe systems, data bases, and applications, TSO remains the most effective tool available on the z/OS computing platform for doing their jobs. While IBM has done little to enhance the TSO environment (with the possible exception of ISPF), other software vendors have provided tools and enhancements that keep TSO a viable interactive platform.

Unfortunately, many of these other software vendors have followed IBM's lead and have abandoned the development of new TSO enhancements and have largely stabilized the enhancement of their existing TSO products. Add to this the overpricing that occurs when software costs are tied to the total capacity of a processor, even when TSO accounts for only a small percentage of the workload, and the market for TSO tools seems to have disappeared.

But what hasn't disappeared is the need for TSO tools and enhancements. As powerful as TSO is, there is room for improvement. We at Laid Back Software believe that many of the short comings of TSO can be solved, resulting in increased productivity for the IT Professionals that depend on it to do their jobs.

What follows are some frequently discussed weaknesses of TSO and how they addressed by Laid Back Software products. This, in no way, negates other vendors' solutions to TSO weaknesses. In fact, our software is designed to compliment the functions of many other products.

What Can Be Done to Improve TSO

One of the most frequent requests for TSO enhancement is the ability to log on with the same User ID multiple times, if not on the same system, at least on each system in a GRSPLEX or SYSPLEX. Frequently cited as a justification (in addition to user productivity) is that other interactive environment have no such restrictions. While this may sound like a good idea, this is one that is NOT a good idea, be it the same or different system.

Dealing with this from an external standpoint, such as operator commands and LOGON RECONNECT, TSO requires unique names for each user's environment. In MVS, this is the User ID, which becomes the Address Space Name. If the same User ID is logged on multiple times on the same system, which Address Space should be canceled when needed? Which one does a user want to be reconnected to when necessary? Which one should messages be sent to? SYSPLEX design (and like it or not, SYSPLEX has become a fundamental part of the MVS architecture) is to allow for "single system image" for the end user (not the Systems Programmer or Operator). So having the same User ID be used simultaneously on multiple systems in a SYSPLEX is just as bad as on a single system.

It is important, however, that TSO users must be able to do more than what a single TSO Session allows. The most common practice is to allow users to have multiple User Ids, allowing for multiple, simultaneous LOGONs. The major user complaint about this solution to the problem is keeping all of their User IDs' passwords in sync. The second most frequent complaint is the need for maintaining multiple security profiles and remembering which ID is which.

The Value of TSO in Today's Computing Environment

There are products, such as PIE/TSO from Unicom Systems, Inc., that increase what can be done in a single TSO session. Such products can greatly increase user productivity, but may not always be cost-effective and may not have kept pace with the continuing evolution of z/OS.

So what TSO needs is a User ID definition (i.e., Security Profile) that can have MULTIPLE personalities (i.e., TSO Profiles). RACF keeps the TSO information for a user separated from the Security Profile by keeping it in a TSO Segment. Because there are so many functions in TSO that are tied to TSO Segment values, such as the PROFILE PREFIX, it is still important for each TSO Address Space to make this unique. When logging on (and not wanting their "default" TSO Segment), suppose the user could specify the name of the segment wanted for that TSO Session. Security authentication (logging on, resource access, etc.) would all be based on the actual User ID. The Address Space Name and PROFILE information would come from the selected TSO Segment. The uniqueness of TSO Sessions are maintained while greatly simplifying TSO User ID administration, both by the Security Administrator and the end user.

The capability described above requires infrastructure changes to both TSO and the underlying Security System (i.e., RACF, CA-ACF2, etc.). These changes are not likely to be made by IBM, so Laid Back Software has implemented an alternative solution.

The Laid Back Software TSO Environment and Administration Manager (TEAM) provides a Secondary User ID LOGON capability which allows a TSO user to have multiple User IDs connected to a single "master" User ID and password. While each User ID does require a unique Security System Profile, once they are built and connected to the User's Primary ID, little has to be done to them. Logging on TSO with a secondary User ID always specifies the password of the Primary User ID, eliminating the need to change the password on multiple IDs when one of them changes.

A related capability is the concept of a Group User ID. A Group User ID is one that does not belong to any single individual, but belongs to a group of users, such as the Systems Programmers or DBAs. A User ID can be created that belongs to a group of users. When logging on using such an ID, not only does the Group User ID and Password have to be specified, the User ID and Password of the ACTUAL user is also required. Only specific Users (e.g., the members of a department) are authorized to make use of the Group User ID.

A Group User ID could be used to provide special privileges to a user that he or she does not normally have. Support personnel sometime require access to sensitive production data in order to perform their jobs, especially in problem recovery situations. However, their day to day activities do not require such access. Giving such access to the Group User ID, but not the individual User IDs in a department provides a method of limiting the distribution of critical access while allowing the users to do their jobs.

TSO Users, like any other computer user, sometime run into problems that disrupt their sessions. If a user loses network connectivity to the mainframe, TSO lets the user reestablish the connect via a RECONNECT function. But this requires that TSO know that the user's connectivity has been lost. When it doesn't, the user cannot recover the session and it usually has

The Value of TSO in Today's Computing Environment

to be canceled. Equally, when a user's sessions gets hung up, it has to be canceled. In both these cases, someone has to be called to perform the session cancellation.

There's no reason why someone other than the user needs to get involved for these common problems. A user should be able to cancel his or her own session. They should be able to force their session to be disconnected so that a reconnection will be successful. This is also useful for someone who is called away from where they are currently connected and wish to transfer their session to where they are. TEAM provides this capability.

A TSO Session is built from the User's Security Profile, a few user specified parameters, and a LOGON JCL Procedure. The user might have some type of complex "setup" CLIST to further customize the session, but the LOGON JCL Procedure has to be built and maintained by the installation support staff. Why? The TSO Environment and Administration Manager lets a user build their own LOGON Procedure in their own data set, specifying what they need without having to resort to either a complex setup CLIST or cajoling support staff into building a special LOGON Procedure for them.

System support staff occasionally need to work on the system from TSO while keeping other users from utilizing it. Standard TSO controls LOGONs only through the setting of a maximum user count, making it difficult to allow support personnel to log on while preventing others from accessing the system. Using the Security System to verify access to a settable resource can let the installation define any number of LOGON environments and restrict access to them. Utilizing TEAM's ability to do this can provide any number of controlled access environments.

TSO Usability Enhancements

One of the hardest problems in TSO to solve is a LOGON failure. Most installations direct the TSO Session JCL and System Messages listing to the bit-bucket. When a user's attempt to log on fails, the information necessary to determine the cause of the failure has been purged. So the user ends up having to involve the Help Desk or a Systems Programmer in determining the cause. Depending on the authority of the person contacted, operations may also have to get involved to enter commands to allow the output that is normally discarded to be captured.

Not only is the productivity of the user experiencing the problem reduced, but one or more support personnel have to get involved - needlessly. TEAM allows the user to change the SYSOUT Class of the TSO Sessions, allowing the error output to be kept. Then logging on using a standard environment (which most likely won't have the error), the use can look at the failed Session's SYSOUT just like a Batch Job. The problem can be solved faster with fewer personnel.

Many years ago, IBM implemented a full screen LOGON that allows the user to change a few parameters like the LOGON Procedure Name and Account Number. They also required the user's password to be specified on this screen, forcing the user to enter the password on this screen. Most installations use a Network Solicitor that has previously verified the User ID and Password and passes the information to TSO when the user selects it. So why prompt the user again? Screen scraping scripts in the Network Solicitor can bypass this, but then how does the user change one of the other parameters?

The Value of TSO in Today's Computing Environment

TEAM has considerable more LOGON parameters than the half a dozen that the IBM LOGON screen provides. But TEAM can accept a user's password along with the User ID and will prompt for both the User ID and Password in line mode, making any automated LOGON scripts simpler. Full screen prompting for parameter changes are only made if REQUESTED by the user. This makes the typical LOGON friendlier.

What TSO Doesn't Need

One thing that many TSO Users have always wanted to do is start foreground processes, such as program compiles, and allow them to execute while then doing something else. Well, guess what? MVS has always had this capability. It's called BATCH! If a hundred TSO Users all decided to run compiles simultaneously, all of them would be forced to run on the system each user is logged onto, none of them could be swapped out, and all of them would fight for the CPU. Batch, on the other hand, especially with the advent of WLM managed initiators, allows the system to fully manage the level of concurrent work that it can handle. Sure, the user doesn't have the instant gratification of having the work done NOW, as batch work may have to wait its turn. SO WHAT! Batch work capabilities is something that MVS has that many other interactive systems don't have. It should be considered part of the power of TSO. So what if the work has to wait a few minutes because of someone else's work.

Summary

TSO remains a critical tool for IT Professionals that support the z/OS environment. While IBM and many software vendors have quit providing software for TSO, there are new functions and capabilities available for all TSO Users. Laid Back Software is committed to improving their productivity.

For more information on the TSO Environment and Administration Manager from Laid Back Software, or to request a ninety day trial, go to <http://www.laidbacksoftware.com>, e-mail us at keith@laidbacksoftware.com, or give us a call at (408) 749-0655.