DINO WHITEPAPER

EVALUATING z/OS SRM PRODUCTS

"Realities," Part 1

Dino-Software

Realities of z/OS SRM Tools - Part 1

Management of data on z/OS Systems is complex. A handful of software programs are available, but are they really a "tool?" Do they simplify or complicate? Do they contribute to savings or drive up cost of operations? Do they actually maximize limited resources, human & physical? Are they an asset, or are they a detriment?

Often, more time, money, and resources are spent on installation, configuration, customization, training, and the subsequent cumbersome maintenance of the altered product; this ultimately adds up to the overall cost outweighing the intended benefit.

In today's fast paced and agile environments, classical SRM products have lagged because of their aged and rigid architecture.

This two-part paper will explore key factors which should be considered when evaluating and selecting a z/OS SRM solution. Part 1 will examine Implementation Factors. If the product is difficult to install and customize, it may never come out of the box. Part 2 will dive into important elements of usability. If the product is cumbersome and inconsistent, it will be difficult to learn and prone to errors.

Installing and Customizing an SRM Tool

Implementing a typical SRM tool will consist of the GUI component, or web interface, and the host engine which performs the actions.
Installing and maintaining these components can be major projects in themselves. The following aspects should be considered during the evaluation and selection process:

Network Considerations

To manage all systems from a single interface, that interface needs to have access to the Host systems to be managed. Generally, there are two ways to achieve this:

- A controlling System or server can be set up which receives the requests and then funnels them to the other systems via proxy servers.
- The GUI has direct TCP/IP access to each System.

If proxy servers are used, the reality of finding a System in the Enterprise that has access to all other Systems is usually problematic due to data segregation issues. The net result is either the SRM tool doesn't manage some Systems, or there are two or more configurations necessary, meaning it is not an actual "Enterprise view" of the environment.

In most situations, a PC will have access to all Systems needing to be managed. This being the case, a GUI that communicates directly with the Host z/OS servers will be able to access all Systems with a minimum amount of network configuration. This is a major plus.

Security Considerations

An SRM tool is only going to be practical if it can be used to take actions of choice against, for example, the data being displayed. Actions will generally take two forms: the running of a JCL deck or the issuing of a command.
If a command is being issued, then under whose authority is the command issued? Usually, it is the SRM server. This presents a security nightmare and an extra workload; and, if the server has excessive privileges, anyone with access to the server inherits those privileges.

A second issue is the automation components within the SRM tool. Nearly all SRM tools have some sort of scheduler that handles submission of commands, for example, at specific times. Again, whose authorization do those automation processes receive? If it's the server's, then once there is access to the server, the security norm is effectively bypassed.

A well-designed SRM tool will ensure that authorization is taken from the issuing user's security profile; and, in the case of automation, the option should be available to inherit the authorization of the defining user rather than the SRM server.

Parameters Considerations

Each System on which the SRM engine will be running, needs to know many things about the environment in which it will be running. This ensures it can correctly obtain the information needed; for example, if CA1 Tape is active, it might need to know the name of the CA1 Tape Catalog.

In most cases, this information is provided via startup parameters which the user customizes on each system where the engine is running.

This parameter customization substantially increases the installation time. Also, if the environment ever changes, it adds another level of complication, whereby it must be ensured that the SRM tool is always modified accordingly. An ideal SRM tool should be able to automatically detect the resources it requires, thereby reducing the impact of environmental changes and customization timelines.

Software Versioning Considerations

Having gone through the initial workload of installing the SRM product, at some point, a new version will be available consisting of Host and/or GUI changes. What, however, if the new GUI cannot talk to old versions of the Host component?

In a large Enterprise, the opportunity to upgrade the Host component on all the z/OS systems at the same time does not exist, and a staged upgrade process will occur.

Having differing Host Levels might mean the GUI can only talk to certain hosts. In those cases, the solution is to have multiple GUI versions, each talking to the hosts whose software version they support.

For an SRM tool to be usable and manageable, it must allow for disparate host software versions and simply ignore features if they are not available.

Resource Consumption Considerations

Most z/OS SRM solutions will, at some point, submit background processes to scan the environment and create databases of information about the environment which the GUI can then analyze.

These processes are infamous for eating up all the available resources when they run, often to the detriment of other processes.

The tools usually provide the ability to reduce the frequency at which these processes run, but then the accuracy of the information is reduced because the data is aged and often bears no relevance to the environment at the time the query was run.

The best solution is resource efficient data collection processes and giving the user the ability to clearly and quickly specify whether to use historic information or current information, i.e., a dynamic scan of the environment.

Customization Considerations

Go to 10 different z/OS sites, and it is likely that they have the same requirements for Storage audits, Storage reporting, etc. Sadly, z/OS SRM tools don't seem to be delivered with these requirements. The catalogs need backing up, the catalogs need auditing, admins need to know who is using the most space, and they need to audit

the DFSMShsm environment. So why does the SRM tool not do this automatically? In the end, someone needs to customize each and every LPAR where the SRM tool is running to ensure the tool is performing these tasks.

There are two issues here: the first is that it needs to be done at all, and the second is that it must be done on each LPAR/Sysplex individually, rather than providing the option to select a process to be run daily on all Systems/Sysplexes.

This customization is not a simple task; it is a major project in itself. Very often, the time and expense of this is not factored into the overall costs of the SRM tool.

An ideal SRM tool will remove the need for lengthy and sometimes redundant customization projects. It will give the user the power to define things once and apply it across the enterprise.

Conclusion of Part 1

Evaluating SRM tools can be an exhausting task. The best tool becomes an asset to the business by helping streamline processes, ensure consistency, and proactively protect business continuity and resiliency.

The second part of this article will take a closer look at ways an SRM tool should help alleviate headaches for users and improve their productivity through some essential usability characteristics.

dino-software.com | 703.768.2610 | sales@dino-software.com