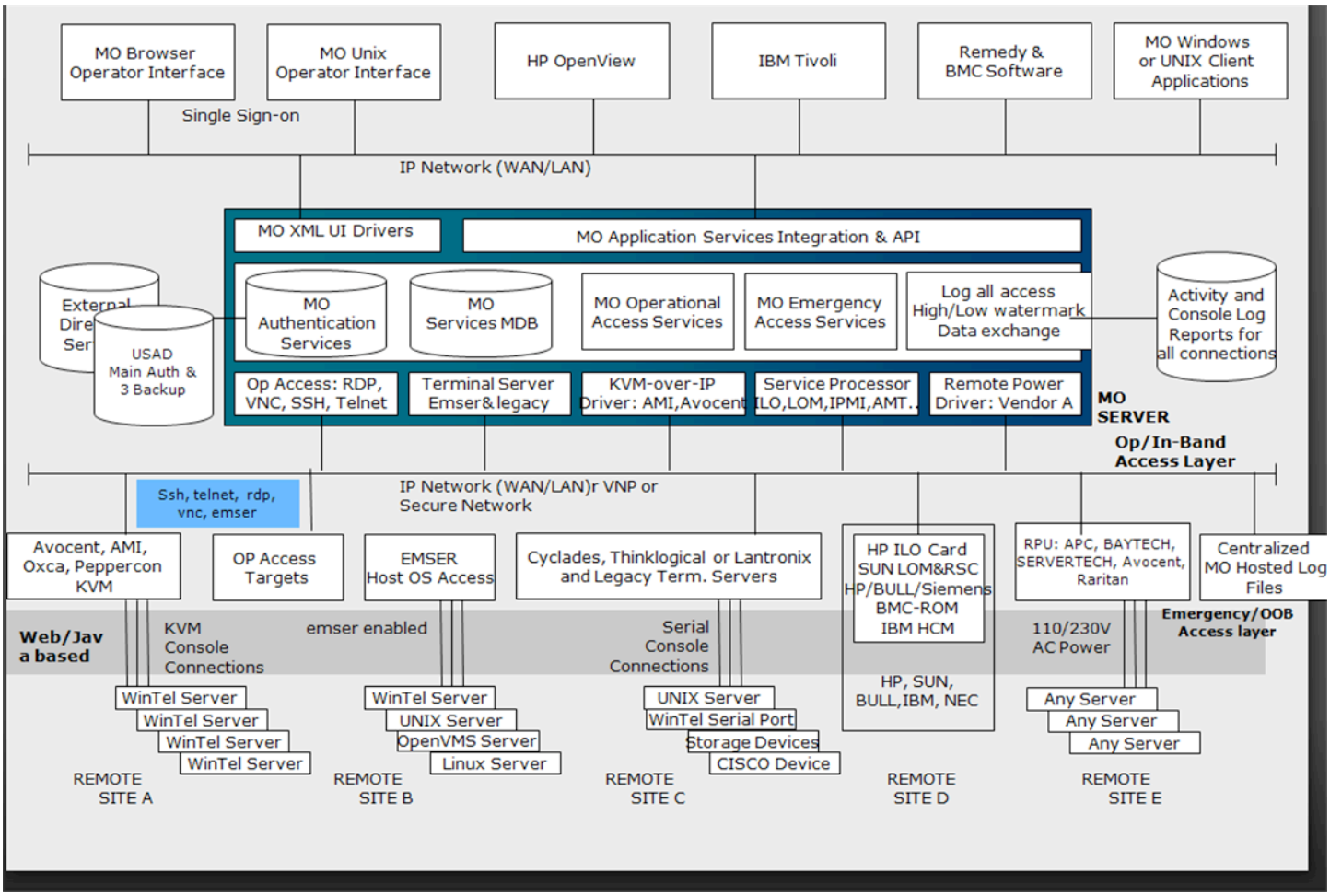
MO Architecture
Delivering Collaborative Problem Solving from Anywhere to Anywhere



See Appendix A "Architecture Diagram"

The MO architecture delivers anywhere to anywhere collaborative problem solving using **vendor agnostic,** out-of-band access via KVMs, Service Processors (SP), Console Servers (CS), System Controllers (SC) and Serial IO cards and in-band access via RDP, VNC, SSH, telnet, SSH and MO's super/emser services. A patent -pending mop or MO pipe allows a single connection to SP, CS,SC,Serial IO card port,SSH, telnet, or super/emser to be shared by many geographically disparate problem solvers including system, network, storage, security admins. Each admin has the same shared access and view. One admin team transitioning admin responsibilities to another team or companies merging admin teams to manage geographically dispersed data centers may transition and merge knowing that they share access.

This software solution's services are securely and easily distributed/redundant and share data real-time via a patent-pending MO collection data backbone. The collected data is stored locally in memory and a SQL database for use by the services, service applications or management frameworks using the data. Devices names, access paths to devices, peer and direct provider names of distributed services , log file names/configurations , event configuration, user authorizations, KVM, CS, SC, etc. configurations, power management, command automation configuration make up the real-time collected data shared across the collected data backbone.

This document delineates how MO's, once known as C-LIM's and CCM's services evolved and their specific purpose.

MO Architecture is Distributed

This paper presents the MO architecture in terms of:

1. A history of MO architecture's evolution
2. Evolution of Service daemons and their functions
3. Service Daemon Data Flow

History

Distributed, Centrally Distributed History

Since 1984 we have been providing software solutions that bridge disparate worlds. Our initial work was developing DECnet, LAT and MOP protocols for non-DEC operating systems and hardware to bridge TCP/IP and DECnet networks. MO has evolved to bridge islands of computing and remote management hardware like different vendors KVMs, console servers, and system controllers. From the beginning, the MO architecture adhered to the concept of dividing functionality along the functions performed by each service. In the spirit of the ISO Network Layer stack, each service, "layer", performed its functions independently. Each service evolved to share and use the collected data backbone. Each of the services has evolved to provide/deliver a specific and distinct function much like the layer in a network stack. Since our services were modeled on the network stack, they are well suited to be distributed over a network. Just as the protocol layers communicate with each other using specific and well designed message formats so too did the initial service daemons. The message exchange evolved to use the collected data backbone to ensure the data each service needed was readily and locally available in memory.

The MO architecture's first iteration was developed under the sponsorship of IBM and PSA Peugeot Citroen in Paris, France, 1994 during a very hot summer in an unfinished and not yet air conditioned NOC. PSA had a NOC in a Paris suburb. This NOC used the corporate TCP/IP backbone. Their 10 major manufacturing facilities used DECnet. PSA needed to manage the DEC hosts and network over their TCP/IP backbone. The architecture had to support centralized control and configuration of DEC hosts and DECnet network. An appliance was placed in each plant that communicated with central NetView and System Manager IBM appliance over the corporate TCP/IP network. DEC and DECnet specific information was collected by function specific daemons over the DECnet/MOP/LAT network and passed back to the Paris-based NOC. The NOC used our daemons to gain access to the DEC hosts and the terminal servers that supported LAT access to the DEC hosts.

Over time we deployed our service daemons to AIX, Windows, OS2, Linux , major vendor UNIXes, and openVMS across a multiplicity of hardware including: RS6K, PPC, PA-Risc, Intel, Sparc, Sparc compatibles, Motorola 68K, and Cray to name a few. Deploying our services to multiple operating systems and hardware platforms has made them highly robust: each OS/platform presented its own unique issues that were overcome as we implemented. This highly diverse operating system experience also provided us with the means to quickly deploy our service daemons onto any new OS or hardware.

Placing the collected data backbone info into a SQL database and providing service application binaries allows management frameworks to add remote management access and log file viewing, monitoring and scanning as well as changed file alerts. Our first implementation is with eyeOS: see http://moproject.pbworks.com/WE http://moproject.pbworks.com/LF use my email corrigan@ManageOperations.net password: eyeOS137. Next implementation is on http://oblong.com using g-speak based system. We hope System Director will allow an integration.

Evolution of the MO Services

MO services have evolved over time, adhering to the principle of each service being independent and function specific. Each service was stand alone and performed specific function-related tasks that range from system resource/log file/applications monitoring to remote management access to automatically configuring console servers for immediate deployment to remote power control and monitoring.

Each service operates independently and may be co-located on a single appliance/server or distributed across many servers/appliances.

Each MO service follows a template that allows facilitates the timely addition of new services. This MO service cookie cutter template was back fitted to the original MO services

<u>The MO Service Template</u>

A MO service template has evolved and continues to evolve. Each service:

- connected to our operations service, kopd. The kopd ensure a service was always running; if the service stopped the kopd, operations service restarted it.
- installed itself and created the directories and files it needed and took further configuration from other service daemons
- allowed incoming connections from OPS service, kopd, or other service daemons; certain services are evolving to to find other MO services and connect out.
- used the Data Access Facility or DAF to ensure uniform and highly secure data exchange; evolving to a single and uniform data exchange facility, also made code maintenance easier.
- shared all of its data with MO collected data backbone or mocd.
- used the mocd, collected data backbone data for User Interfaces' display lists
- had a command line based and graphical-based configuration tool for service administration.
- interacted with it service specific applications that were also command line and graphically based; graphical based evolved to use GTK based applications and eyeOS enabled web desktop; eyeOS is the linux of cloud computing.

The MOD evolution

The mod service supports remote access, power control, and writes keystrokes and output to a unique file for each accessed device. This service was the core of the 1994 release of our consolidated console manager. The Poly Center Console Manager was the only console manager product as the time. PCM was sold to Computer Associates.

This service allowed the mapping of device names to specific console server ports; the user only need remember the device name and not have to know what console server port was the way to the device. Unlike PCM, the mod provided management of all terminal servers and console servers including configuration of each port. PCM only worked with serial ports as did the mod initially. The mod service evolved to include opening connections to KVMs, service processors, Active Management Technology (Intel vPRO), IBM system controllers: HMCs and RSAs, Uplogix's PODs and many more.

The mod service maintains a constant/persistent connection that supports sharing a single command window between multiple admins delivering collaborative trouble shooting:

- console server ports attached via cable to the device's serial command port
- service processor command lines including Intel IPMI based, SUN RSC/LOM, HP ILO/RILOE; this includes SMASH on those SPs supporting it.
- unique to mod: SSH and telnet sessions that are always connected and allow multiple admin sessions within the one connection
- HMC's LPAR console
- Uplogix's POD devices such as Solaris SUN and CIsco Routers/Switches.

User configurable shortcuts or command sequences allow the user to send a break sequence (if authorized), view past log activity, execute a file containing recorded command sequences, make the connected session read-write or read only to name a few.

The mod service evolved to support "gang connects" to complete repetitive work across multiple devices. The gang connect allowed one window to accept commands that were sent to multiple devices. A boot device firmware upgrade for 200 SUNs completed in the time that it took to do one firmware update. The admin gang connects to all 200 SUNs from a single command window and brings them to the boot prompt, applies the firmware update. Admin gang connects to 600 servers and applies in the command window an application update. All 600 are updated in the time it takes to do one. All output and inputs from all 200 SUN boot device or 600 servers are stored in unique access log files.

The mod service captures all I/O coming from or going to a device including serial card I/O, console servers, a wholly unique persistent SSH/telnet connections, SP command lines, HMC and Uplogix attached consoles. The mod service was expanded to write captured keystrokes and output (I/O) to a local disk file as well as simultaneously to one remote mod or many other remote mods. The remote mod ,real-time, writes the access log file to its local disk. Security folks are able to scan/ monitor all capture keystrokes of access logs on their security servers without having to go out to the specific servers/appliances where mod runs. A side benefit was it protected against modification of access log files on any MO server that may have been done to hide what some may not have wanted to be known. The mod may be configured to scan for certain patterns for real-time alerts like servers at a boot prompt. The log file scanning, viewing and monitoring is done by the mologd service.

The mod service evolved from just presenting the login window for service processors and KVMs to doing the login and presenting the virtual KVM screen or command line of all service processors and most KVMs. In addition to logging into a KVM, SP and any device supporting login prompts; the service allows the admins to get to work and not have to fumble for passwords or navigate many dialogs to get a virtual KVM window.

The mod service has evolved to provide a conduit for returning system resource (CPU, Memory, Disk Usage and application monitoring) data via a mod to emser pipe.

The mod service has evolved to interact with our emser and super services to provide auto configuration of all major vendors console servers. The mod interacts with emser to configure each console server port including baud rate, stop bits, etc. Devices attached to the console server are discovered and added to the available devices.

The mod service initially provided a rudimentary access to remote power units via out-of-band access via a console server; the user was required to enter the remote power unit vendor specific commands. From this rudimentary interface, mod evolved to “know” the vendor specific commands for power off/on/cycle; mod supported plug associations that included more than one plug/outlet to be configured for each device connected to a remote power unit. For service processors supporting power operations, the power off/on/cycle sequences were added to the mod service. This service provided the ability to power off/on/cycle without the user knowing the service processor or remote power unit vendor’s specific commands.

The mod services allows command sequences to be recorded or captured to a file. The recorded commands may be played back by selecting the command file from a list during a command line or GUI session. Senior admins may train junior admins and allow them to execute complex tasks while watching the commands execute, the junior admin is accomplishing what a senior admin could do and learns while the commands roll by.

The command line binaries represent the functions mod supports; the commands and their description are in MOD Application commands. The GTK applications and eyeOS enabled application are also listed in the table. All memory mapped database maybe viewed or modified with mdbbrowse followed by the name of the service.

The mod related files for mod service are:

- /var/log/MO/acceeslog contains all device console log files and the archived console log files
- /etc/MO/database/clim.mdb, tsm.mdb - clidump output in /var/log/MO/*mem* and *disk* files
- mod-emser use cacert.pem, keyfile.pem and dh1024.pem in /etc/MO
- /etc/MO/daf.config delinates which console servers, servers, managed nodes, etc. are allowed contact
- /var/log/MO/export conatins the text versions of the mod configurations

The climcp command allows the export of the entire mod configuration. This configuration is saved to a text file, passwords are encrypted. Each of the services supports the ability to save their configuration from a saved “exported” file or in the case of three services store the configuration in an XML specification. This feature is also used to take a spreadsheet of devices, their addresses, console server or KVM or SP connection addresses, names and convert that spreadsheet to climcp commands. The spreadsheet to climcp commands allows devices to be configured in advance of the MO server software.

Commands using the mod Service:

MOD command	Description
clif	establish console session as if directly connected to device command serial port via console server, over the network to SP command line, emser/super, SSH or telnet sessions and many other access methods as well
nclif	ncurses based clif
motargetstatus	returns the MODEM signals and connection status of a device connected to a console server running emser/super
moshowcpuhistory	display CPU usage of all managed nodes. A managed node is connected via a mod to emser/super pipe.
moshowidlecpu	display all managed nodes whose CPU is below Uptime institute's 3 % active threshold that should be removed or retired. A managed node is connected via a mod to emser/super pipe.
mopoweron/off/cycle	controls power actions on single device or multiple devices in a specified order - allows sequenced power down
morestart	restarts a single device or multiple devices in a specified order; requires kopdx/emser or super to be present on device
moshutdown	shuts down a single device or multiple devices in a specified order; requires kopdx/emser or super to be present on device
eloop	loops the mod-emser pipe returning emser uptime and emser enabled devices uptime; it also returns version and status
climcp	command line configuration program for device, power units
we	we and the cloud computing, eyeOS WE application ICON; each of these graphical interfaces has a number of action icons to power on/off/cycle, look at resources, connect, gang connect, etc.
pc	pc allows for direct control of remote power units
ctop	ctop is "top" on steroids and uses the mod-emser/super pipe to allow process list review of multiple servers in same window, disk and memory use, activating CPU and mounting/un-mounting disks on all managed nodes. A managed node is connected via a mod to emser/super pipe.
tm/spm	tm is the equivalent of climcp with graphical interface; spm allows configuration of many advanced mod features
clidump	dumps the contents of the in memory and persistent store mod configuration

The MOCD Evolution

Each service shared information initially with Major Management Framework such as BMC Patrol, Tivoli TEC/NetView, HP Operations Manager/OV centric. MO services evolved to use our a highly distributed and resource efficient MO Collected Data backbone or mocd service. The relationship between MO servers may be defined as peer or provider or placed in “auto-discovery” mode. A peer relationship means all information is equally shared amongst those targets defined as peers. A direct provider relationship is a directed graph. The relationship is one MO server acts as a provider of services to one or more MO servers. A simple example is MO-1 has a direct provider MO-2 and MO-2 has a direct provider MO-3. This means as in a directed graph: MO-1 knows all about the services MO-2 and MO-3 are providing and MO-2 knows all about MO-3. This simple example may easily expand to any number of combinations. With auto-discovery set and the number of hops to search for other mocd services set, an automatic peer relationship is established.

The primary purpose of mocd is to get MO service information disseminated in an efficient and timely manner so that all defined relationships have the data they require to perform services across a distributed number of platforms hosting one or many MO services in addition to mocd. The exchange of data is established by one of the three relationships detailed above. Only data that has changed is sent as an update after initial exchange of data. The throughputs are very high.

The mocd service is a memory mapped database to allow high speed access to information. We were pleasantly surprised to find that the memory mapped services on the supported OSs including linux, unix, OpenVMS and Windows were identical. The mocd collects information from each local MO service including mod (remote access and power) , mologd (file log viewing/monitoring/ scanning), moevmd (event configuration), mousad (user authorization) and kopd (system management, system resource management, application availability). The local mocd shares information with those mocd that have been auto-discovered or have had a peer or direct provider relationship defined.

The mocd allows data to be exchanged between as many mocd as you like. All applications use the mocd to get their display info and to know where the particular service is offered, locally or remote.. The delivery of a given MO service may be locally or remotely provided. The applications using a given service gets its display info and details from the mocd. The display info may contain information about remote services and provides info for sure about the local services. The mocd evolved from a daemon called kids or Ki Data Services.

the mocd related files are c/etc/MO/config/kids-configuration.xml file and the /etc/MO/database/ kids.mdb. T The relationship of the local mocd service to other/remote mocd services is set in the XML file. The stored collected data is stored in /etc/MO/database/kids.mdb; kidscp dump places all mocd configuration in /var/log/MO/mocd.log.

MOCD Command	Description
kidscp	allows for the configuration of the local mocd service relationship with other/remote mocd services. Allows connection to other mocd services for purposes of viewing remote configurations
spm	spm allows configuration of mocd advanced features

The MOLOGD Evolution

The MO log file service, mologd, displayed the access console logs and scanned those access log files using a REGEX based pattern.These access console log files were written for each device by mod. The service of viewing, scanning and monitoring the log files was a distinct function. The mologd evolved to search for changed files using the CERT recommended files to look for changes in size, CRC, date of creation/modification or if the file has gone missing.

The mologd evolved to include the ability to view/scan/monitor remote log files from a local MO server. The log files were extended to include system event files such as

- *nix messages file or any file specified in the syslogd.conf.
- openVMS operator.log and other log files
- Windows resource event logs including system, network and security.

A version that interacted with emser and kopdx, called lfmx was spawned. The lfmx was establish to operate with the mocd on the MO server that accessed the three MO smart services. The lfmx is also merged with the super service detailed below.

The remote mologd or lfmx accesses log files on the server where they reside and interacts with the MO server as a managed node to view a file or report a found pattern. The definition of the emser service and its evolution is detailed below. The mologd and lfmx may be directed to scan any log file looking for REGEX patterns and generate alerts via the mod-emser pipe in the case of lfmx. In the case of mologd on the remote, the mologd on the local MO server establishes a pipe/channel. The lfmx sends alerts via the mod-emserr pipe and mologd to moevmd locally or remotely. Archived or saved versions o f log files are appended to the current log file by default so that a complete history may be viewed.

The mologd evolved to support one or many mologd service(s) providing configuration services for local and remote log file management. This log file scanning and monitoring is realtime and will continue if the connection between emser-lfmx or mologd-mologd is disrupted.

The command line binaries represent the functions mologd supports; the commands and their description are in MOLOGD Application commands. The GTK applications and eyeOS enabled application are also listed in the table. All memory mapped collected databases maybe viewed or modified with mdbbrowse command followed by the name of the service.

The mologd applications allow a pattern to be found and a view of the lines before and after as well as using patterns as delimiters within a log file. The mologd applications evolved to send this specific view of the log file via email, to save file to a storage device or to print to a printer. The mologd accepted TAG inserts into any access log file to denote a marker in the log file to refer back to at a later time for more study. Think of it as an electronic book mark in the viewed log file

The related mologd services files are:

- /etc/MO/database /logd.mdb.
- the lfmcp dump command places the complete log file manager configuration into /var/log/MO/mologd.log for display
- lfmcp export command allows the mologd configuration to be stored as lfmcp configuration commands; this file is used to restore a failed mologd service configuration

The mologd service applications include:

MOLOGD Command	Description
lfmcp	is the command line version of the GTK lfm application. It allows the user to configure or restore mologd entities.
aviewlog	allows viewing the log file from the command line using options such as specifying a pattern to find in the log file, lines before and after, view only last lines of log file, etc.
lf	is the GTK and eyeOS enabled application. the GTK application is /usr/MO/bin/lf; this application allows viewing of the log files
lfm	is the GTKapplication. the GTK application is /usr/MO/bin/lfm; this application allows log file management locally and remotely

PLEASE NOTE: all entries in the /var/log?MO/accesslog console log files are time stamped

The MOEVMD Evolution

The moevmd service started life evmd service. This service name conflicted with evmd that started on TRU64 servers. The moevmd allows the definition of all event types in terms of importance of the alert, what action should be taken when. The mocd was expanded to send any and all events to multiple major frameworks such as HP OpenView, IBM Tivoli NetView and TEC, BMC Software's PEM, SNMP traps. There are dynamic event configurations that suppress large numbers of duplicate alerts and actions taken on those alerts.

The movemd service allows time of day specific actions to be taken.Events may be turn off for certain hosts for a pre-specified period of time when you know the host/device will be down or under service.

The moevmd service may send events to all major frameworks , other moevmd services, use SNMP, TeMIP , BMC PEM. The mologd automatically creates events when log file scan patterns are specified.

The moevmd service allows a binary to be called with arguments passed from the event. The most effective use of the executable called in response to an event notification is the use of the mod services clif command; clif command using the variable arguments such as name of the device/ target allows the immediate connection to the device via the most expeditious mod connection. If the server is at the boot prompt, a console server, KVM or SP will be automatically invoked as the defined ssh or telnet will fail. The moevmd also allows these commands or actions to be taken only if the operator specifically selects that the action be taken; moevmd service calls that carbon meets silicon!

The related moevmd services files are:

- A text version of all events is stored in /var/log/MO/kievmd.eventlogs.
- evmcp dump allows the moevmd configuration to be displayed in /var/log/moevmd.log
- evmcp export allows the moevmd configuration to be stored in evmcp configuration commands for easy and quick restore
- /etc/MO/database/evmd.mdb contains the in memory moevmd service database.

The moevmd services applications include:

evmcp	is the command line version of the GTK evm application. It allows the user to configure or restore mologd entities.
aviewlog -events	allows all events to be displayed in a list from the command line

evm	is the GTK based application that allows events, their subsequent action response (if any) , event priority, time of day response, event forwarding to be configured,
ev	allows all events to be displayed in a list using the GTK application GUI. greater detail is store for each event and is available by a simple click

The KOPD Evolution

The kopd service initially provide the means to restart any stopped MO service daemon. The kopd stored for each service's:

- start and stop commands
- name of binary,
- poll interval to verify service is operational
- id as a MO service as opposed to all other binaries

The kopd service made a connection to the MO service and sent keep alive messages to each service over a device socket/id. If the service did not respond, the service was restarted, i.e. killed and started. The time period of the delay in response from a service daemon evolved to allow a weighted factor so services were not being restarted just because they were busy.

The kopd service provided the first command line interface to the local server. The kopd service supported a command command that allowed the specification of a native command while connected to a kopd enabled server. This command command evolved to include the ability to launch the command window for any given OS. An uniform set of commands: show process, show files, show cpu and show memory returned the exact same output for each host OS including all major UNIX vendors, all linux platforms, Windows, and OpenVMS. A MAC OS X Leopard release is planned for fall 2009. **See Appendix B" Uniform command and output"** two examples Linux and Windows.

The kopdx code is a binary and relies on kernel calls or documented system interfaces to retrieve server resource information and process lists. The kopd service uses less than one percent (1%) of available CPU and addresses the generally unacceptable overhead scripts generate when gathering this information. The information returned was vendor independent and highly accurate as well. This portion of kopd project was started in 1998 and completed in 2001. This service does far more than any similar service from a major framework vendor or even the smaller vendors such as groundwork.

The kopd service evolved to run on all operating systems and was released as kopdx a remote kopd service that provides system resource, application availability and restart and shutdown capability. This information is stored locally and also provided to a MO server requesting the information via one of the MO service applications detailed above in MOD applications as well as via the command line kopcp application. The servers, console servers or VM guest OSs running kopdx or kopd are labeled managed nodes. A manage node will soon have a single service called super discussed below. the super service will replace lfmx, emser, kopdx with a single super service that be placed on remote servers and virtual machines to all virtual machine management independence You manage your servers independent of VMware, Hyper-V, Xen Server, Ironside, etc

The MO services are being modified to find and connect to existing kopd services that designate they are looking for managed nodes. The service local to the kopd will also connect to kopd if kopd does not connect to them and will start kopd if for some reason kopd were to stop.

The relevant kopd service files are:

- /etc/MO/config/kopd-configuration.xml which stores the kopd service daemon and applications to be monitored

- /etc/MO/database/rsmondb.mdb which holds the resource information that is current as well as daily weekly and monthly averages.
- kopcp export places a backup of the /etc/MO/config/kopd-configuration.xml of the current kopd service configuration

The kopd service applications include:

KOPD command	Description
moshowcpuhistory	display CPU usage of all managed nodes. A managed node is connected via a mod to emser/super pipe.
moshowidlecpu	display all managed nodes whose CPU is below Uptime institute's 3 % active threshold that should be removed or retired. A managed node is connected via a mod to emser/super pipe.
mopoweron/off/cycle	controls power actions on single device or multiple devices in a specified order - allows sequenced power down
morestart	restarts a single device or multiple devices in a specified order; requires kopdx/emser or super to be present on device
moshutdown	shuts down a single device or multiple devices in a specified order; requires kopdx/emser or super to be present on device
ctop	ctop is "top" on steroids and uses the mod-emser/super pipe to allow process list review of multiple servers in same window, disk and memory use, activating CPU and mounting/un-mounting disks on all managed nodes. A managed node is connected via a mod to emser/super pipe.
we - resource action icon	the we GTK application - action icon allows users to get a view of any managed node's resource usage: CPU, Disks, Memory and process list.
kopcp	configures service daemons and applications to monitor to restart if they fail. Tthe kopcp application may connect to any kopd/kopdx/super and stop or start process, mount or unmount disks.

The Future for KOPD

The kopd service future is providing a file that lists the day to day, weekly, monthly and annual tasks that would be done by the administrator. The file of tasks might be expressed as expect commands and would server to document the tasks for a year. the time of day to run is also contained in the file. the results of these commands are written to a log file that is scanned by the mologd/lfmx/super service to find any failures.

The EMSER Evolution

The emser service released in late 2004 on all major console/terminal servers. This service/daemon runs on all major linux based console servers, all versions of Linux, major UNIX, Solaris, HPUX, AIX and Windows. The emser service evolved to provide a command line access that is VM framework independent and instant as well as highly secure accecss.
See Appendix C Emser in Detail: What is EMSER?

The emser provides command line access via serial ports or provides a command window directly on the host device, i.e., every major console server vendor console server, desktop, VM guest OS, notebook or server.

MO and emser team to provide single software solution that spans major systems and console servers. Tteh emser services allows hardware selection to be on price and merit. The emser service provides a consistent, uniform, extremely efficient, highly secure, login-less access to authorized MO server users, and time saving access that is logged to a unique access log file.

The emser service for console servers basically works in this fashion:

- emser attaches to the console ports on the cyclades, lantronix, raritan, digi, dnpg, opengear, thinklogical, basically an linux based console server.
- mod establishes a pipe/channel to emser - a high water and low water mark threshold for data being transferred/exchanged. An alert or event indication is sent when too much activity is going on or too little
- all console I/O is multiplexed across the the pipe between the MO server and emser
- using emser_ttys file allocates console ports for specific MO servers or not allow any port to be configure to ensure you have reserve ports in case or one of the other ports failing
- An emser service application is emstty. if you need to access the console port while you are on the cyclades. using emstty ensures all access to the port is logged even when accessing on the cyclades, i.e., when logged into the cyclades.

The mod to emser connection/pipe supports

- one connection rather than 48 connections. or two rather than 96 connections per console server; you can do the math to see the savings across 40 console servers. The mod service has less sockets to look after making the code more efficient in CPU and network resource use. Less delays in ensure data is captured when establishing connections to ports. 48 or 96 ssh connections take longer than one or two connections
- a connection that is secure: 1024 bit encrypted, message exchange is proprietary, certificates checked on both sides of the connection.
- quick access to the command line of cyclades or any console server - no login is required - instant access and the access to the command console is logged. the emser service automatically sets up a emser device name for the console server when the mod-emser connection is established
- saving time that is involved with many console server configuration programs, additional steps such as setting up shared access to a cyclades port and the extra step of establishing a persistent connection to the port so as to capture any data coming from the serial command console of the device connected to the console server.
- simple control of port allocation between MO servers
- quick configuration for the console server (the console server name and IP address is all that is required) , the mapping of port to server name, baud rate etc. is stored on the MO server. if

the cyclades or other consoles server needs to be replaced, assign the same IP address copy emser and start emser. You are done
- vendor independence as no knowledge of vendor specific commands is required - there is no need to know how to configure the cyclades or any other console server or are you required to go thru the hassle of configuring via vendor specific commands.
- replacement of the cyclades or any other vendor's console server with another vendors console server you load emser on the new vendor's console server give it the same IP address and you are ready to go.

Bottom line: emser is much simpler to use, more efficient, vendor independent.

The emser service was expanded to operate on Windows, Linux, Apple OS X Leopard, all major UNIX, OpenVMS. The mod clif application is used to gain access to the command window via "-ems" or oer the mod-emser pipe.. The mod opens one connection to the emser-enabled server or VM guest OS. All clif sessions are multiplex over the mod-emser connection and all I/O from each session is stored and time stamped in HN_emslog file in /var/log/MO/accesslog directory.

the clif application supports overriding the user name specification if allowed when connecting to the server over a mod-emser connection. There is no login process so access is instantaneous and not dependent on an external authorization other than mousad service authorization that is discussed in detail below. The emser service places an entry in the wtmp file so programs checking 'out-dated/ old-fashion logins" will have a record of the access over the mod-emser pipe/connection.

More than one mod service or multiple MO servers may have access to the emser service on a server. The same is true of the mod services wehn establishing a mod-emser connection with console server.

The relevant files for emser are
- emser_ttys that allows the individual ports on a console server to be assigned to one or more mod services.
- the cacert.pem, keyfile.pem and dh1024.pem files to establish secure connection
- the /etc/MO/daf.config that defines the managed nodes and console server access list.

The emser applications include:

EMSER commands	Description
clif -ems "device/target name"	the clif application allows the user to take advantage of the mod-emser connection/pipe that has been established.
emstty	is on a console server or local host connects to the specified port number. all I/O is logged as if a clif session had been started remotely
eloop	loops the mod-emser pipe returning emser uptime and emser enabled devices uptime; it also returns version and status

The MOUSAD Evolution

The mousad service enables the devices to be assigned to groups. The groups have roles assigned to the devices in a group. The user is assigned roles. The roles specify the action allowed for each role and for each group of devices.

The mousad service was tightly coupled with mod service; the mousad and mod services are decoupled. the mosuad service allows a role of specific actions/access methods to be applied to a group of devices for a specific user. The role actions include what service application the user may use for the device in the group. The service applications are divided in two two catagories: one is product administration, the other is for accessing devices, looking at events, power controls and log files.The role includes actions such as allowing break to be sent or ems as root access over mod-emser connection.

The relevant mosuad service files are

- /etc/MO/config/usad-configuration.xml. the user authorization configurations is stored in the usad-configuration.xml and written into memory for fast access.
- /etc/MO/daf.config that allows one mousad service to established as the authorization service

MOUSAD commands	Descriptions
uacp	the uacp is the command line version of the GTK uadm application; the uacp export creates a backup copy of the usad-configuration.xml file
uadm	the GTK graphical interface for defining roles, groups and users
spm	a GTK application that allows redundant mousad service authorization servers to be defined in the case of mousad service failure on primary

The SUPER Evolution

the super service is underway to combine the emser/lfmx/kopdx service to reduce the service to one from three and reduce the foot print of largely ahred libraries to a single set of shared libraries. The super service will provide the services detailed above by emser, mologd/lfmx and kopd/kopdx services.

The super service future is providing a file that lists the day to day, weekly, monthly and annual tasks that would be done by the administrator. The file of tasks might be expressed as expect commands and would server to document the tasks for a year. the time of day to run is also contained in the file. the results of these commands are written to a log file that is scanned by the mologd/lfmx/super service to find any failures.

The super service will look for mod services that have enabled connections from “emser” or super services. This connection will allow the super service to make a connection out from behind a firewall.

The MOSCHED Evolution

The mosched service provides scheduling procedures to be run on specific devices at specific times. Since the clif application is used and the mod connections to devices, the cron command may also be just as easily used.

The Semi-retired Services: BUFORD, SNORTD, Network, Biz Process

The buford service or backup failover recovery daemon service has been retired. The mosnotrd, network service and biz service will be turned up at user request.

The Future

The MO collection data service and the template for creating services bodes well for MO services. Services to gather asset information available via Intel Active Management Technology or vPro is planned for release. Our architecture does well in a single or dual server configuration and also does well when widely distributed. the ability to group devices allows for tens of thousands of servers to be defined.

The clod computing front end based on eyeOS (http:/eyeOS.org) allows for the User Interfce to be presented anywhere in the cloud while utilizing to private and protected collected data to connect anywhere to perform remote management access, remote power and log file viewing, scanning or monitoring.

Wizards

A deployment aids include our wizards that discover hardware and then automatically add it to the mod known devices/targets saving a deployment time. the ILOWIZ and HMCWIZ are examples of these time saving auto-discovery with auto-configure built in.

Access Hardware History

In 2002, we approached each major access hardware company offering to provide a single management and access interface for each of their disparate console servers. We had needed to support 6 or more vendors' proprietary configuration interfaces for console/terminal servers; the process was time consuming for us and in some cases such as Cyclades, DIGI and DNPG we had to supply the vendor with a command line interface that did not exist so the user did not have to edit files to accomplish configuration. The configts and tsxcp on cyclades had its oriigins with the code we freely provided to cyclades as they had no command line configuration.We proposed to these vendors standardizing the management and access interface to ensure uniformity across vendors and simplify our coding efforts to ensure higher quality of configuration and access service. Our efforts were rebuffed by these vendors.

These vendors did not want to have a single management and access interface that was standard across multiple vendors; they resisted this effort as it reduced console servers to a commodity. A commodity that allowed the customer to purchase the console server that provided the best functionality, such as redundant power supplies, console ports or network connections at the best price. Basically, it would break the strangle hold the vendors had on their customers to ensure that once they deployed their console servers it would be cost prohibitive to remove and replace them, a practice aptly named "back hoeing". What made this resistance all the more frustrating was that

these vendors were using an open operating system, Linux, to establish this strangle hold. As so often is the case the cloud had a silver lining: a linux based operating system allowed us to develop a uniform configuration and access interface called EMSER.

In 2004 under the sponsorship of then Daimler Chrysler and the support of Logical Solutions we completed EMSER for all Linux based console servers. Raritan and Lantronix proceeded to close their Linux based servers making it more difficult to deploy EMSER. Both companies have since openned their servers on a case by case basis. EMSER was extended later that year to also run and provide command line access to all operating systems including Windows, UNIX, LINUX and OpenVMS. **See Appendix C Emser in Detail: What is EMSER?**

The mod service has provided the means to gain access to virtual KVMs from raritan, avocent and OXCA as well as the DELL Drac and HP ILO/RILOE, INtel's IPMI and AMT, IBMs HMCs controlled LPARS without the user having to log into each remote management hardware.

Remote Power History

In 1999, HP and UNILEVER sponsored the development of remote power control via remote power units. The first implementations included working closely with Bay Tech and Server Tech to supply the ability to provide a standardized, uniform interface to configuring, monitoring and control remote power units. We also provided the ability o associate multiple outlets from multiple remote power units to a single target. This feature was essential for UNILEVER to be able to point and click to control power for servers with redundant power supplies. They had to turn off two sets of outlets on two separate remote power units. We have added any number of additional vendors in 2000-3 including Avocent, Raritan and WTI.

In 2005 we extended remote power control to service processors found on HP, SUN, DELL and IBM systems including Intel's IPMI, HP's ILO/RMC/SMC/MBM and SUN's A/LOM/RSC. In 2006 and 2007 we added support for Intel's Active Management Technology and AP Rack and Switch power units.

The goal with remote power was the same as console server implementations: provide a single uniform, standard interface for configuring and accessing the remote power units allowing the user to select remote power units and/or service processors that were the best fit and reducing the number of tools required to manage power by eliminating the need for vendor specific software tools.

Fail-over/Fail-safe History

The design goal of making the deployment easily fail-safe and to easily replicate was achieved in 1999 under the sponsorship of HP, GSK, and UNILEVER. This first generation was accomplished by providing a backup/failover Service that replicated configurations to up to three hot standbys. The next, current generation established three levels Peer, Provider and Auto-discoverys that may be replicated in an unlimited fashion and run active all the time, rather than in hot stand –by. The configuration was greatly simplified as well.

Security

The exchange between desktop, MO peers, MO providers is over a highly secure 1024 bit encrypted links. The security is further enhanced as the message exchange is proprietary and thus adds a level of complexity to anyone wishing to break in. The access to any given MO peer server, MO provider

or EMSER-enabled access hardware is controlled by these entities using access and privilege lists that are resident on each entity.
In a separate paper we cover the security that has been implemented. This security controls what users any access what targets and what actions they are allowed to take.

Appendix A: Architecture Diagram

MO applications provide a web, GTK based GUI and command line interface to the MO services. There are integration modules for major frameworks including OpenView, BMC Software and Tivoli. These applications include console connection, log file viewing and scanning, remote power control, event viewing and action response, and monitoring compute resources and applications' availability.

The MO applications reside on desktops running the MO application client or access the MO server's Web Server. These applications also are available on the MO server itself. The applications are directed to the Service provider that has a connection to the target. The Services run on MO peers and Servers. The console servers and targets run emser, application/resource monitoring and log file monitoring and viewing.

The software is engineered to work well with very large numbers of targets under its control or a small number. A deployment with multiple peers controlling many MO servers handles 1000s to 10000s of device/targets. A single MO server may handle as many in excess of 2000 servers or as few as 1.

The first layer is User Interface: GUI, Command line, and Web;The architecture has allowed for and has as its basis an Uniform and standard configuration and access interface for remote console servers or remote power entities, service processor or remote power unit. We are committed to providing KVM users with a standard configuration interface. The web interface now using a cloud computing web based desktop from eyeOS http://eyeOS.org . This interface ensure that the UI may run anywhere and access the MO services from anywhere. We include in the first layer the ability launch rdp/rdesktop or vncviewer and automatically log into the device supporting these protocols / access methods. the first layer is where the applications using the MO services live.

The second layer is Services: log file scanning and viewing, console access via KVM, console server, serial line card, Service Processor (iLO, AMT, etc); console access exists for operational access using telnet, ssh, RDP, VNC and EMSER. This layer has event management and console server and line card configuration. Remote power configuration and control is a service delivered via Service processors or remote control units. this second layer is where the MO services live.

The third layer is the IP network that supports operational access and access to the access hardware infrastructure as well as the access hardware itself including KVMs, Serial attached ports, Console servers, Service Processors. The Service Processors remote power control as well as Remote Power Units are present. The remote management access hardware lives in the third layer.

The fourth layer consists of the devices/targets that are being accessed via console port, service processor or directly using our services or ssh/telnet/rdp/vnc. Targets connected to remote power units or having Service processors that allow access and/or remote power operations.

Our solution is software; this software may be loaded onto any of the supported platforms that include major UNIX and Linux platforms on a variety of hardware platforms. Our software primarily packaged as a hardware application.

Large numbers of targets with high fault tolerance will benefit from redundant MO peers located in different Geographic locations with desktop clients running anywhere in the world. The MO peers control the multiple MO servers deployed to the specific sites as single servers or redundant servers within the same site. A small installation might have a single MO server embedded on a console server accessed via desktop client or browser. The possible deployments are highly flexible.

Appendix B: Uniform commands and output

The kopcp uniform commands are as follows:

show cpus	Display CPU information
show filesystems	Display filesystem information
show memory	Display memory information
show processes	Display process information
command	Issue a native system command
reboot	Reboot the system
shutdown	Shut down the system
start	Start one or more daemons
stop	Stop one or more daemons or a process

Two Examples:

Linux

```
KOPCP@rackmo>>show cpu
CPU #    Idle    User   System    Wait
  2    99.86%    0.01%    0.06%    0.05%
  1    99.83%    0.06%    0.08%    0.01%
  0    99.73%    0.04%    0.07%    0.13%

KOPCP@rackmo>>show files
  SIZE   AVAIL    USED          NAME               DEVICE          TYPE
------- -------  ------- ------------------------ ----------------- ---------
   608M      0M  100.00% /mnt/virt                /dev/hde          iso9660
  3798M   3798M    0.00% /var/lib/xenstored       none              tmpfs
609250M 407907M   33.04% /netmount/bigdisk1       172.16.16.232:/vol/exports/bigdisk1      nfs
609250M 407907M   33.04% /netmount/bigdisk        172.16.16.230:/vol/exports/bigdisk       nfs
  3798M   3798M    0.00% /dev/shm                 tmpfs             tmpfs
    98M     65M   30.10% /boot                   /dev/sda1         ext3
2069263M 1091001M    0.63% /                     /dev/mapper/VolGroup00-LogVol00      ext3

KOPCP@rackmo>>show mem
real memory:  2929664k total      71068k free    0.00% used
swap memory:  1769464k total    1769204k free    0.01% used
page memory:        0k total          0k free    0.00% usedOPCP@rackmo>>

sho proc
   PID    SIZE    RES SIZE     CPU    NAME
-------- --------- --------- -------  ----------
```

```
  31193    62016k     1396k   0.00%  sshd
  31193    62016k     1396k   0.00%  sshd
  30911    52668k     2332k   0.00%  moeyeosd
  30909    51008k     2352k   0.00%  motcprelayd
  30726    47672k     1900k   0.00%  kopcp
```

Linux example continued

```
KOPCP@rackmo>>command ls
kids-configuration.dtd
kids-configuration.xml
kids-configuration.xsl
kopd-configuration.dtd
kopd-configuration.xml
kopd-configuration.xsl
moeyeosd.ini
motcprelayd.ini
relay-configuration.dtd
relay-configuration.xml
relay-configuration.xsl
```

Windows - connecting to a Windows Vista from a CENTOS 5 32 bit

```
KOPCP@phi>>connect target uzziel
KOPCP@uzziel>>show cpu
CPU #    Idle     User   System     Wait
  1    90.37%    1.55%    7.76%    0.31%
  0    95.65%    0.62%    3.26%    0.46%

KOPCP@uzziel>>sho files
  SIZE   AVAIL    USED          NAME                DEVICE          TYPE
------- -------  ------- ------------------------ ------------------ ---------
 95393M  10627M   88.85% C:\                    Local Disk (C:)        NTFS
    0M      0M    0.00% D:\                   Compact Disc (D:)
    0M      0M    0.00% E:\                   Compact Disc (E:)
    0M      0M    0.00% F:\                   Compact Disc (F:)

KOPCP@uzziel>>show mem
real memory:  2097151k total     261768k free   87.51% used
swap memory:  2097024k total    2023240k free    3.51% used
page memory:  4194303k total    1816036k free   56.70% used

KOPCP@uzziel>>show process
   PID    SIZE    RES SIZE     CPU    NAME
-------- --------- --------- -------  ----------
  446940    19188k    11736k   0.00%  kopdx
  448428    48304k    29548k   0.00%  OCBrowse
  447712     4654k     3526k   0.00%  OCFix
  448428    48304k    29548k   0.00%  OCBrowse
  447296     4634k     3502k   0.00%  OcFix
  431076     2907k     1391k   0.00%  taskeng
  431076     2907k     1391k   0.00%  taskeng
```

```
 429096   153221k    63205k   0.77%  procexp
 431076     2907k     1391k   0.00%  taskeng
 423836    87389k    47201k   0.00%  Dwm
 420492     5731k     2779k   0.00%  unsecapp
 420428   113929k    46097k   0.77%  explorer
 377108     4750k     1922k   0.00%  putty
 377832    12266k     2282k   0.00%  mxserver
 358084    99468k     9808k   0.00%  java
 357940     1823k      279k   0.00%  jp2launcher
 230620   353227k   129579k   0.00%  firefox
 322404   113018k    45254k   0.00%  EXCEL
 377108     4750k     1922k   0.00%  putty
 277636     2434k      270k   0.00%  CMD
 260260     3570k     1354k   0.00%  cmd
 225972     5446k     2462k   0.00%  winvnc4

KOPCP@uzziel>>command dir
 Volume in drive C has no label.
 Volume Serial Number is A0F7-A1E5

 Directory of C:\Windows\system32

07/06/2009  05:39 PM    <DIR>          .
07/06/2009  05:39 PM    <DIR>          ..
04/29/2008  08:15 AM    <DIR>          0409
09/18/2006  05:28 PM             2,151 12520437.cpx
09/18/2006  05:28 PM             2,233 12520850.cpx
09/18/2006  05:32 PM         1,228,100 8point1.wav
01/18/2008  11:33 PM           136,192 aaclient.dll
01/19/2006  12:41 PM           375,296 accelerometercp.CPL
08/07/2008  02:33 PM            14,640 accelerometerdll.DLL
01/17/2006  01:01 AM            53,248 accelerometerST.exe
01/18/2008  11:33 PM         2,515,968 accessibilitycpl.dll
11/02/2006  03:28 AM            39,424 ACCTRES.dll
11/02/2006  05:46 AM             7,680 acledit.dll
01/18/2008  11:33 PM           127,488 aclui.dll
11/02/2006  05:46 AM            38,912 acppage.dll
11/02/2006  03:11 AM             2,048 acprgwiz.dll
06/26/2004  11:21 PM            50,688 acropdf.dll
01/18/2008  11:33 PM           167,424 ActionQueue.dll
01/18/2008  11:33 PM         1,405,952 ActiveContentWizard.dll
01/18/2008  11:33 PM           204,800 activeds.dll
01/18/2008  09:43 PM           111,616 activeds.tlb
01/18/2008  11:33 PM           326,656 actxprxy.dll
01/18/2008  11:33 PM            81,408 ACW.exe
09/18/2006  05:31 PM           107,620 acwizard.ico
```

See Appendix C Emser in Detail: What is EMSER?

What is EMSER?

Since 2004

The emser service released in late 2004 on all major console/terminal servers This service/daemon runs on all major linux based console servers, all versions of Linux, major UNIX, Solaris, HPUX, AIX and Windows. The software provides command line access via serial ports or provides a command window directly on the host device, i.e., console server, desktop, notebook or server. The emser deployment allows multiple servers running MO management applications to gain access to emser enabled hosts and hosts connected to emser enabled console servers. The emser may run via redirected 8689 IP port access allowing access using the Intel Active Management Technology to gain access to a system that is not connected to the network or in single user or diagnostic mode.

MO and emser team to provide single software solution that spans major systems and console servers. This teaming means companies are now able to provide their user community with a consistent, uniform, extremely efficient, highly secure, one time login, and time saving access. The user authenticates via the MO applications and may gain direct access to any emser enabled serial port via line card or consoles server as well as any server, desktop, or notebook without having to login. Intel's AMT enabled servers, notebooks and desktops allow access as well.

EMSER enabled Console Servers

The emser service release on major console servers greatly improves all terminal server access performance, presents a standard uniform access interface across all console servers, allows data and operations centers to purchase the most effective solution and protects investment in existing access infrastructure.

The emser enabled console servers benefit and in many ways including:

- a more efficient use of network, memory and CPU by maintaining a single connection to MO service, climd, that also results in greatly improved performance
- faster access over a single connection pipe and thus not subject multiple sshd connections; one connection over which 4, 8, 16,32 or 48 ports are multiplexed rather than 4,8,16,32,48 or more SSHD connections. the connections to devices attached to the console server is as fast as one connection coming up rather than waiting for single connections to each port to come up; when recovering from network outages re- establishing access quickly is extremely important.
- Centralized configuration is quickly deployed that automatically configures the console server upon the console server's power on; this makes provisioning a new or replacement console server very quickly in a straight forward and simple manner that saves time.
- A single aggregated metric of console activity across all console server ports that allows administrators to monitor system performance by high and low water mark settings of

console activity. Unusual activity in lower or much higher activity may also indicate a security threat.
- the state of the device attached to a console server's serial port may be determined by the EMSER monitored UART Signal Status; the server that is disconnect from the console server and/or powered off is quickly identified.
- Less memory is required on emser enabled console servers as emser takes less memory than the SSHD daemon(s)
- Emser once deployed updates automatically. The code has been stable for 2 years.
- Emser has local access to each serial port via emstty application; means administrators may access the port from the console server while logged in or via the MO application that access the port remotely; both connections may be maintained simultaneously.

MO applications and Services Team with EMSER

The MO command line access service, climd, logs all data; the MO log file manager and viewer scans and views all data that passes between EMSER connected sessions and the climd service. The climd service connection between the emser enabled console server and MO server is 1024 bit encrypted; the message exchange is proprietary and therefore highly secure.

The MO console manager quickly configures the console server with two inputs: the console server name and IP address. The emser deployed to multiple console servers from multiple vendors means the user has a single tool with a single, uniform management interface to any vendors' linux based console servers. The UART signal monitor and other tools described above provide the means to quickly debug or deploy replacement console server connections.

The MO target manager configures the console server port and maps target name to a serial port. Multiple MO servers and the MO access applications may each connect to emser on the console server. The emser on the console server maintains the connection to the serial port. The MO climd service maintains the availability status of a console server attached device/target.

Multiple failures may be tolerated when MO service daemons and emser team. Redundant MO servers and applications may be in place making access highly available in cases of multiple failures; the MO and emser team and greatly reduce the complexity of redundant systems and failover.

The MO log file Services and applications allow scanning of log files using Extended Regular Expression to find important patterns as well as the ability to quickly view important portions of the console log created and maintained by the climd service. The MO log file applications make it easy to email or print relevant portions of the console log quickly and easily. These console log files may be replicated to secure servers in a real time fashion; this ensures finding tampering and allowing other security programs access to the log files without impacting the MO server.