Embedded SMI-S Lessons Learned

Art Colvig & Mike Lamb
IBM
Overview

IBM SMI-S Providers
Why did we embed?
CIMOM Selection
Device 1
Device 2
Device 3
Device 4
Lessons Learned
Outcomes
- For external devices (storage arrays, tape libraries, …)
  - Providers could run on a *proxy* server

![Diagram showing a proxy server with managed devices connected to it]
Providers and CIMOM could be *embedded* in devices either acting as an integral component or as a management server possibly supporting multiple devices.
IBM SMI-S Providers

TS3500
  Large tape library

DS8xxx
  High end block storage

SAN Volume Controller
  High end block storage virtualizer

XIV
  High end block storage
Why did we embed

Support Costs
~50% of field issues tied to proxy CIM to device or client to proxy connectivity issues

Data Collection

Simplify Client Infrastructure
It just works
Common authentication
Reduce data center footprint
CIMOM only manages a single device firmware version

Performance
Scalability
Native interfaces
CIMOM Selection

SNIA CIMOM
- Legacy
- Java – No JVM on some platforms

Pegasus
- Good support
- Intended for Server usage
- Poor memory and thread controls
- C++

SFCB – Small Footprint CIM Broker
- Intended for embedded use
- Good thread Controls
- C99
- Good support
Device 1 - Constraints

Can't interfere with core device services
Soft memory cap
Large legacy install base
Device I - Decisions

Open Pegasus
Loose device coupling
Preserved Proxy
Preserved Compatibility
Device 2-Constraints

Can't interfere with core device services
Restricted Memory – 32 Meg
No Swap
Restricted mass storage
Old C++ implementation
Restricted processor
RTOS
  
  Thread Management
  Minimal OS Services

Thread priority
Device 2 - Decisions

Pre-SFCB CIMOM
Tight device coupling
Sunset Proxy
Preserved Compatibility
Device 3 - Constraints

Can't interfere with core device services

No Swap

Hard Limits on open files
- Clients are naughty – Many clients will leak connections
  - Add netstat to your log collection to help identify naughty clients
- Clients like to send parallel requests even though the device processes them serially
  - No good solution identified other than documentation and defensive coding.

Hard Limits on memory consumption

Hard limits on disk consumption

No more than 5% processor
Device 3 - Decisions

OpenPegasus CIMOM

Tight device coupling

One release overlap with Proxy

Preserved Compatibility
Device 4-Constraints

Can't interfere with core device services
Plenty of Memory
Plenty of Disk
Device 4 - Decisions

OpenPegasus CIMOM

Tight device coupling

No proxy release

Preserved Compatibility
Lessons Learned

OpenPegasus or SFCB … It depends

Ulimit is good and bad

- Hard Limits prevent the CIMOM from interfering with core device function

- New services on a device are the first one to be blamed for problems.... Ulimits help prevent the long nights shooting a problem that is not in your area.

- Out of memory or file handles are not handled gracefully by the CIMOM

Watchdogs are your friends

- Memory
- Responsiveness
- File Handles
Lessons Learned - 2

Client Software is not embedded friendly
- Parallel Connections
- Failure to close connections
- Nasty Queries
- Use protocol analyzers – ex. Ecutre
- Use network analyzers – ex. Wireshark

Indications are not embedded friendly
- They don't expire
- Clients don't clean them up
- There is no upper limit
- Add mechanisms for third party cleanup
- Indications don't cluster well
Lessons Learned - 3

Tight Coupling is good

- Common user names
- Common Log collection
- Common firmware updates
  - This is a design trade-off depending on your customer base and device update strategies.

Enable/disable, Service Reset

SMI-S is not a separate service it is part of the device

Get device team buy in and work with them closely.

Log Management

- Log rotate
- Add mechanisms for managing log levels
- Tie in with system log collection
Outcomes

Significantly reduced connection related problems
Significantly reduced authentication problems
Simplified Deployments – it just works
Simplified infrastructure
Discovered odd CIMOM usage
  Firewall Proxy
  Clustering
Simplified data collection and improved first time data capture
Performance – Flat to slight improvement with a single device
Scaling – Multiple devices saw significant performance improvements.
Exposed and allowed fixing of known but hard to re-create issues.
Native interfaces enabled feature support earlier in feature development.
Questions?