Open Standard CIM
Management for Xen

Dr. Gareth S. Bestor <bestor@us.ibm.com>
Jm Fehlig <jfehlig@novell.com>
What is the Common Information Model (CIM)?

- System management is hard!
  - Huge vendor investment to maintain custom narrow vertical management stacks for their products
  - Minimal interoperability with other vendors' management tools

- The CIM advantage:
  - Functional model-based management infrastructure, independent of vendor’s implementation
  - More expressive than SNMP; e.g. object relationships, inheritance
  - Insulates management front-end from vendor implementation
  - Enables vendors to participate in heterogeneous Data Centers

- CIM is increasingly being adopted across the IT industry
  - Widely used in storage management (SNIA)
  - Growing interest and support by Linux distros (Novell/ SuSE, RH)
  - CIM models are the basis of Microsoft WMI!
Distributed Management Task Force (DMTF) Standards Organization

- DMTF oversees the standardization of CIM models, protocols, usage and certification
  - Membership includes most major IT companies: IBM, Novell, Dell, HP, Intel, AMD, Unisys, XenSource, EMC/VMWare, Microsoft, Sun...
  - Working Groups focus on modeling Virtualization, Clusters, Databases, Networks, Security, Desktop/Mobile, ...
- Publishes CIM protocol standards documents:
  - DSP0004 – CIM Infrastructure Specification
  - DSP0201 – Representing CIM in XML
  - DSP0134 – SMBIOS Specification
- Publishes regular *CIM Schema* models
  - 4 month revision cycle; changes go in tagged as “experimental”
The CIM Management Stack

- **CIM client**
  - translate management commands to CIM-XML

- **CIMOM (CIM Object Manager)**
  - listens for incoming CIM requests
  - invokes the CIM provider for the requested CIM class(es)
  - unmarshals incoming CIM-XML into CIM objects
  - marshals provider results into CIM-XML to stream back to client

- **CIM providers**
  - implements CIM ops against the managed resources/services

- **Local APIs, CLIs and daemons**
  - manage the actual host resources
CIM Models - CIM_CompilerSystem

- **CIM instances** represent various system components:
  - physical devices, I/O controllers...
  - operating system objects (e.g. processes, daemons)
  - other runtime mgmt services
- **CIM associations** show relationships between objects
  - describe topology of entire sub-systems and their components
- **CIM class hierarchy**
  - subclass instances and associations for a specific implementation
  - inherit and override parent class properties
CIM Class Definition (MOF) - CIM_Fan

// Copyright (c) 2005 DMTF. All rights reserved.
// ==================================================================
// CIM_Fan
// ==================================================================
[Version ("2.6.0"), Description ("Capabilities and management of a Fan CoolingDevice.")]
class CIM_Fan : CIM_CoolingDevice {

    [Description ("Indication of whether the fan supports variable speeds.")]
    boolean VariableSpeed;

    [Description ("DesiredSpeed is the currently requested fan speed, defined "
        "in Revolutions per Minute, when a variable speed fan is "
        "supported (VariableSpeed boolean = TRUE). The current speed "
        "is determined via a sensor (CIM_Tachometer) that is "
        "associated with the Fan using the CIM_AssociatedSensor "
        "relationship.").
        Units ("Revolutions per Minute")]
    uint64 DesiredSpeed;

    [Description ("Requests that the Fan speed be set to the value specified "
        "in the method's input parameter. The return value should be "
        "0 if the request was successfully executed, 1 if the "
        "request is not supported and some other value if an error "
        "occurred. In a subclass, the set of possible return codes "
        "could be specified, using a ValueMap qualifier on the "
        "method. The strings to which the ValueMap contents are "
        "translated' may also be specified in the subclass as a "
        "Values array qualifier.").]
    uint32 SetSpeed (
        [IN, Description ("The desired speed for the fan.")]
        uint64 DesiredSpeed);

};
The Open Source Xen-CIM Project

- Collaboration established at January '06 Xen Summit between XenSource, IBM, Novell, Unisys and others to develop CIM management instrumentation for Xen.

- Open Source implementation of the DMTF System Virtualization Model for Xen
  - General model for managing virtual systems being developed by IBM, EMC/VMWare, Microsoft, Sun, HP and others
  - Work in progress; will first appear in CIM Schema v2.14 (Dec '06)
  - Driving new Xen API requirements; e.g. managing inactive DomUs

- Xen-CIM is the first working (partial) implementation of the Open Standard DMTF System Virtualization model!
**Xen-CIM Model of Dom0 and DomUs**

- **ComputerSystem (CS)**
  - hosting system Dom0
  - guest DomUs
- **LogicalDevice**
  - host CPU, RAM, disks, NICs
  - DomU CPU, RAM, disks, NICs
- **SystemDevice** (association)
  - match LogicalDevice with its ComputerSystem
- **ResourceAllocationSettingData**
  - Xen-specific settings for the virtualized LogicalDevices
- **DomU CS model mirrors Host CS**
  - Recursive! e.g. DomU running container-based virtualization
  - Nested hypervisors (zVM)
DomU Creation and Lifecycle Management

- VirtualizationManagementService:
  - DefineVirtualSystem() registers a new DomU config
    • pass in array of CPU, memory, disk & NIC settings
  - DestroyVirtualSystem() removes DomU config
- Lifecycle operations are initiated via CIM’s existing ComputerSystem RequestStateChange() method:
  - \texttt{RSC(Enabled)} = start/resume the DomU
  - \texttt{RSC(Disabled)} = stop the DomU (but retain its config)
  - \texttt{RSC(Quiesce)} = pause
  - \texttt{RSC(Enabled but Offline)} = suspend
- CIM provider invokes appropriate APIs to initiate DomU creation/destruction or perform lifecycle operation
DomU State Transition Diagram
DomU Resource Allocation & Resource Pools

- Host resources aggregated into **pools**
  - CPU’s, memory pool, virtual NIC pool...

- DomU creation allocates resources from the appropriate pool(s)

- Resource allocation for virtual device is distinct from virtual device itself
  - Xen_Processor vs Xen_ProcessorRASD
  - Allows non-virtual aware CIM client to manage virtual systems exactly like physical systems
  - Flexibility in how virtual systems' CIM objects are exposed
    - out-of-band vs in-band provider
CIM Instance Diagram - Active DomU
The Plumbing: Xend API, libvirt & xm

- Original IBM LTC CIM providers written to a custom API
  - DomU config stored in xenstore
  - No support for reading & writing xm config files
  - No support for DomUs created via xm!

- Xen- CIM providers ported to libvirt via temporary 'shim'
  - Stub to read & write DomU config as XML file
  - Custom code to list inactive domains (unknown to xen!)
  - Libvirt backend to xend, xenstore & hypervisor as needed

- Replacing shim with direct calls to Ewan's new Xen API C bindings, giving Xen tool/xm interoperability

- Future use of libvirt for hypervisor- agnostic CIM providers
Xen-CIM Status and Roadmap

- **Sept 2006**: Partial implementation of the draft DMTF System Virtualization Model using libvirt
- **Dec 2006**: Full implementation of the published DMTF System Virtualization Model using the new Xen API
- **2007**: Shipped in SLES10 SP1 & user-addon to RHEL5

Remaining and ongoing tasks:
- Expose more Xen resource types
- Expose DomU scheduling functions
- Support resource reallocation; e.g. memory ballooning, CPU pinning
- Migration & cluster support
- Keep up with DMTF Model changes...
**Helpful Links**

- **Xen- CIM Wiki (under construction)**
- **Standards Based Linux Instrumentation for Manageability (SBLIM)**
  - [http://www.sblim.org/](http://www.sblim.org/)
- **OpenWBEM CIMOM**
- **OpenPegasus CIMOM**
  - [http://www.openpegasus.org/](http://www.openpegasus.org/)
- **Open Management with CIM Project**
- **CIM Tutorials**