



Matahari and FMCI

Remote and local APIs for Systems Management and Configuration

Red Hat

Jaroslav Reznik, Perry Myers and Andrew Beekhoff, based on original Matahari slides

February 10, 2011

Copyright © 2011 Jaroslav Reznik, Perry Myers and Andrew Beekhoff, based on original Matahari slides, Red Hat.

This work is licensed under a Creative Commons Attribution 3.0 Unported License (CC-BY).



Part I

Introduction



Section 1

Welcome and introduction

Agenda

1 Welcome and introduction

About me

- Software Engineer @Red Hat Czech, Brno, responsible for:
 - Qt and KDE maintenance and development,
 - System configuration tools.
- Active in several open source communities.
 - Fedora
 - KDE
 - Linux v Brně
 - Openmobility

- Matahari stands for Remote and local APIs for Systems Management and Configuration
- FMCI stands for Fedora Management and Configuration Infrastructure
- Same goal, different approaches led to merge

Goals

- Collection of generically useful APIs accessible over a remote and local interfaces via a collection of Agents
- Framework for adding small set of Agents and APIs
- Provide easy extensibility for new Agents and APIs
- Cross Platform: Linux (Fedora) / Windows
- Can be extended to additional OSes and Distros
- Can be used on virtualized guests and bare metal hosts

Use cases

- Provide Guest Introspection / Control for
 - Cloud Deployments
 - Virtual Machine Management Environments
 - High Availability of Virtual Machines
- General OS Management
 - General purpose local / remote systems management (FMCI / system-config-* tools)
 - Integration with tools like kickstart / Puppet for post boot configuration (appliance / cloud models)

General usage

Agents

- Agent is a
 - QMF agent, which is a daemon that runs and exposes a QMF model to a qpid bus
 - DBus agent, which is a daemon that runs or activated on demand and exposes it's interface on DBus system bus
- Matahari agents provide
 - methods invocation
 - properties and statistics (QMF)
 - events
- Any agent using QMF, DBus or both could be a Matahari agent
- Agents are dumb, policies are driven via external management infrastructure
- Agents may run as a standalone agents, no need for the whole Matahari stack and can be packaged and shipped separately

Functional areas

- **Host** – Hardware identification
- **Net** – add, remove, start, stop and query network interfaces
- **Services** – start, stop, monitor, query services
- **Logging** – retrieve application and system logs by data and filter
- **Configuration** – read, write, query configuration files
- **User** – add, remove, update and query users and groups
- **Storage** – mount / unmount filesystems, reporting
- **Applications** / Packages – add, remove, query installed and available packages
- any other future agents???

Limited scope!

- yes but we don't want agents for agents to be in Matakari, functionality driven by concrete needs instead of imagined future functionality
- For example:
 - Matakari will provide APIs for installing packages and files, but...
 - It is up to server-side tools like Spacewalk and Puppet, to aggregate these capabilities into a centralized management or provisioning solution
 - Matakari will provide APIs for starting / stopping / querying services, but...
 - Matakari contains no logic for when those operations should occur
 - This is the role of a centralized management server (eg. a cluster or cloud management engine)
- This is where Matakari meets FMCI, the high level API for the local management

Let's Not Reinvent the Wheel

- API reuse, not development
- Whenever possible we simply expose an existing stable API via a QMF Model and DBus interface
- Examples:
 - **sigar** – Library of cross platform host management APIs
 - **augeas** – Granular configuration management
 - **libvirt** – virtualization management for Hosts
 - libvirt-qpid is effectively a Matahari Agent, though separate from the core Matahari packages



Part II

Architecture

Transports

- QMF is an object modeling framework that layers on top of AMQP / Qpid
- DBus is a message bus system for interprocess communication
- Qpid default transport is over TCP
- AMQP / virtio-serial used for guest to host comms
- Additional transport types could be integrated into Qpid project
- Remote management servers would be written as QMF Consoles or REST clients

Independent agents

- Functionality is split up into multiple agents
 - SELinux
 - Simplifies bug-fixing and testing
 - Allows for pluggable architecture
 - Agents are written to be deployed / used independently
- Each agent contains a set API functions
- actual implementation in shared C library, then exposed via QMF Model (C++) and DBus (C)

Cross platform

- Sigar libraries used to provide cross-platform and cross-distribution APIs
- mingw32 used to cross-compile windows binaries from Linux
 - Integrates into existing distribution build chains
 - No need for proprietary build tools
 - Avoids most licensing issues for shipping code compiled by Microsoft native tools

Authentication and Policies

- For QMF (Qpid broker)
 - Access to bus is restricted, policies are solved on QMF level
- DBus
 - System bus is open, PolKit 1 is used to authorize actions

Why Qpid and DBus?

- For QMF (Qpid broker)
 - Open and Standards based bus (AMQP)
 - Cross platform
 - Reliable, Fast and Secure
- DBus
 - De facto standard messaging bus on Linux
 - DBus and PolKit 1 solution preferred on desktops for system configuration tools
 - Easy to use, running on nearly all Linux based systems, not need to setup broker



Part III

Matahari in use

Status

- We are ready to merge DBus branch to master
- Host, Network and Services APIs working on Fedora and Windows
- Windows installer built in Fedora

Roadmap

- Inclusion into Fedora 15
- Initial Use Cases
 - Cloud Guest Agent for Aeolus (Cloud Engine) Project
 - LRM (Local Resource Manager) replacement for Pacemaker
 - Ricci replacement for Fedora Cluster Stack
- Future Agents: storage, user management, packaging and generic configuration interface (Augeas - Radek Novacek)

Links

- Qpid/QMF – <http://qpid.apache.org>
- FMCI – <http://www.fedora.redhat.com/wiki/Features/FMCI>
- Hyperic Sigar – <http://sourceforge.net/projects/sigar/>



The end.

Thanks for listening.