Truth

On machine epistemology and the importance of dependencies

Leon Towns-von Stauber
Cascadia IT, March 2013
http://www.occam.com/sa/

Contents

Legal Notices

- This presentation Copyright © 2013 Leon Towns-von Stauber. All rights reserved.
- All trademarks are the property of their respective owners.

Introduction

- If you're involved in managing more than about 40 systems (physical and/or virtual), and you had access to a computer on your network, could you quickly:
  - –Answer *precisely* how many systems you manage?
  - –For each of those systems, provide (for example):
    - OS version
    - Hardware model
    - Amount of installed/allocated RAM
    - Physical location, to rack and U
    - Serial number
  - –Determine the number of active login sessions on each system?
  - –Determine whether configuration management is operating correctly on each system?

Introduction

- I've rarely worked in shops where anyone could claim to answer such questions with 100% accuracy
- In the analogy of IT to manufacturing (see: Lean IT, theory of constraints, etc.), our systems are like the machines on the factory floor
  - Do you think that manufacturing companies don't have reliable knowledge about every significant piece of equipment in a factory: where it is, what it's doing, whether it's working as it should?

The Repository of Truth

The Repository of Truth

- You need a single *repository* of truth about machines
- *Sources* of truth are multiple
  - –Machine (RAM, CPU, MAC address, serial number, etc.)
  - –Software (OS and version, installed software, configurations, etc.)
  - –Physical environment (location, asset tag, network/power connections, temperature, etc.)
  - –Organization (responsible persons, role of system, etc.)
  - –The repository may (should) also be the source of some truths
- There should be exactly one source of truth for any datum
  - –All other instances of that datum should be determined (programmatically if possible) from the designated source of truth
  - –This is an ideal, which may not be completely possible in practice

The Repository of Truth

- Most organizations have multiple independent repositories of data about machines, few of which are authoritative
  - –For example, my current shop has over 20 that I've counted, including:
    - RackTables (2 different instances)
    - The hosts themselves
    - Virtual machine management systems
    - DNS zone files, Active Directory
    - Network device configurations (switches, load balancers, firewalls)
    - Software configurations (CFEngine, Nagios, Cacti, backup software)
    - Lists used by `dsh`, database account management tools, cache flush systems, and other automation

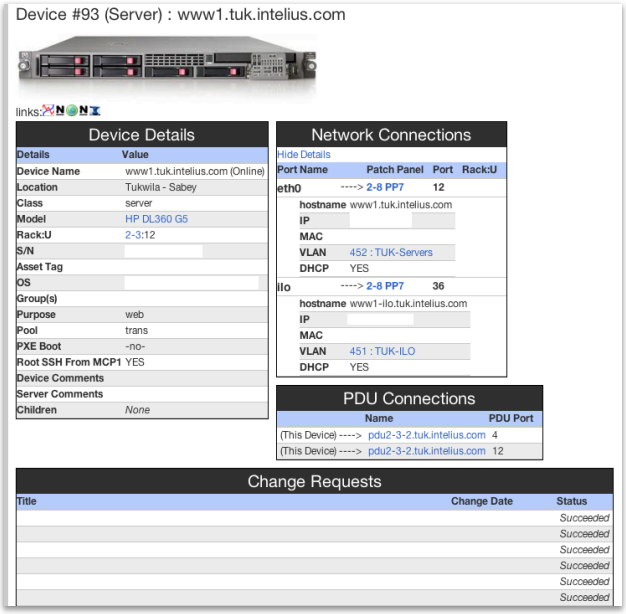
The Repository of Truth

- Implementation
  - –The repository is most likely a database, with a web front end
  - –Features it should have (incomplete list)
    - CRUD functionality (Create/Read/Update/Delete)
    - Authentication
    - Authorization (differentiated access to functions and content)
    - Accounting: what was done and who did it (not necessary, but very useful)
  - –A plain database can do all of these, but SQL isn't the best user interface

The Repository of Truth

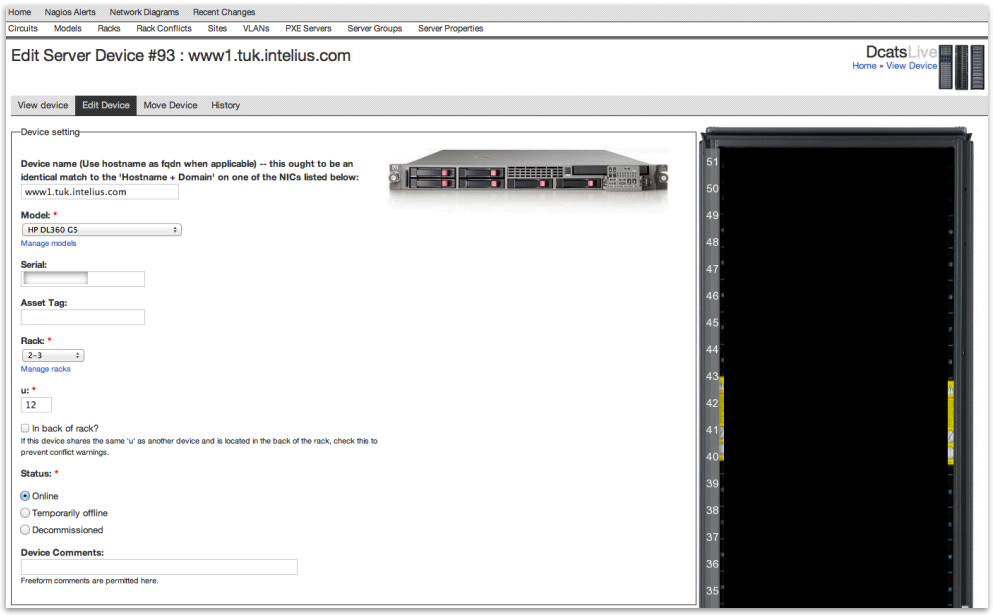
- A repository of truth may be referred to as, or be part of, a:
  - –Machine database
  - –Configuration management database (CMDB) in ITIL-speak
  - –Data center infrastructure management (DCIM) tool
- Examples
  - –RackTables
  - –dcTrack® (from Raritan)
  - –DCATS (in-house application at previous shop)

The Repository of Truth



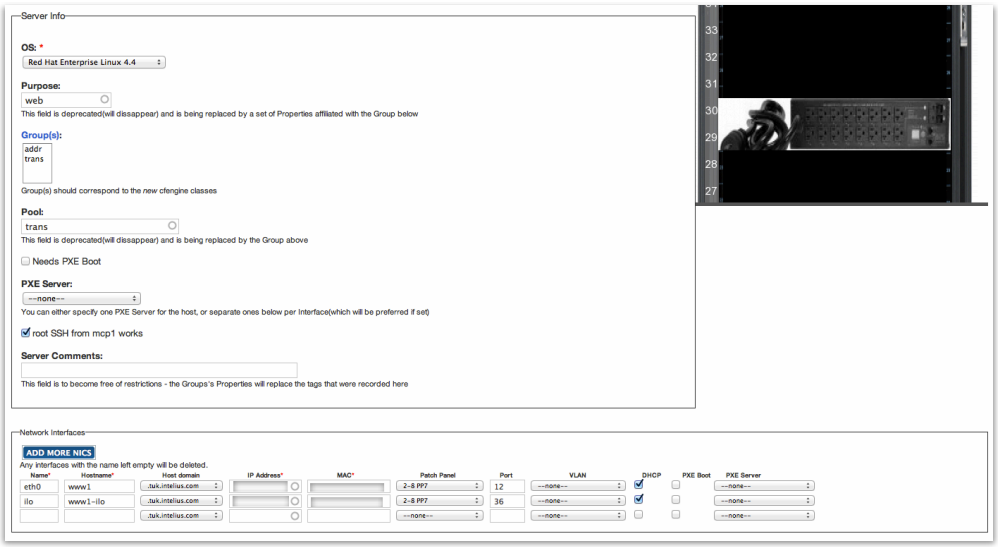
DCATS Device Details

The Repository of Truth



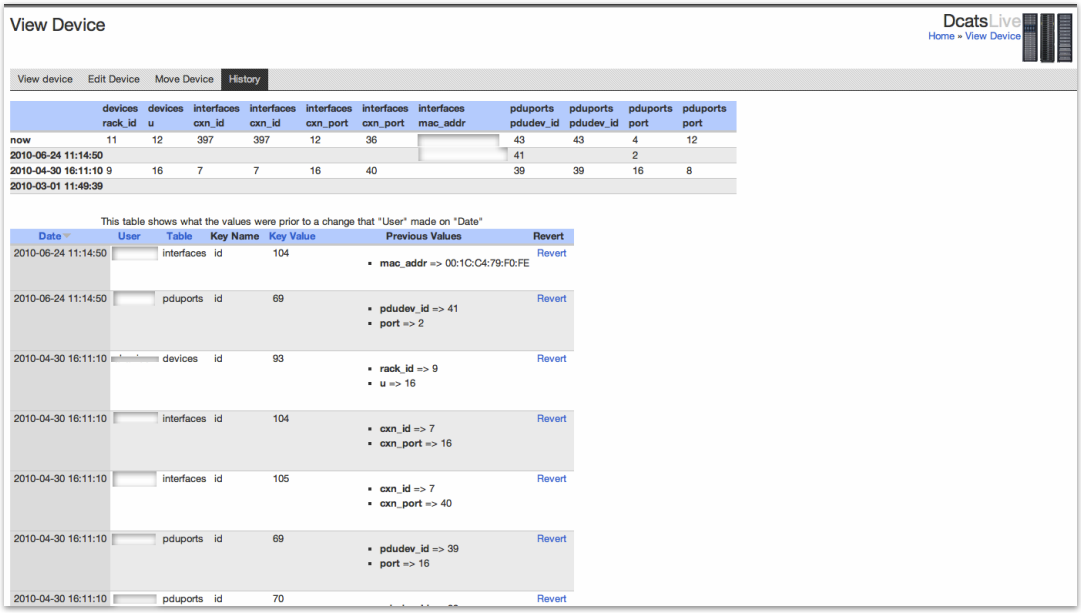
DCATS Device Editing

The Repository of Truth



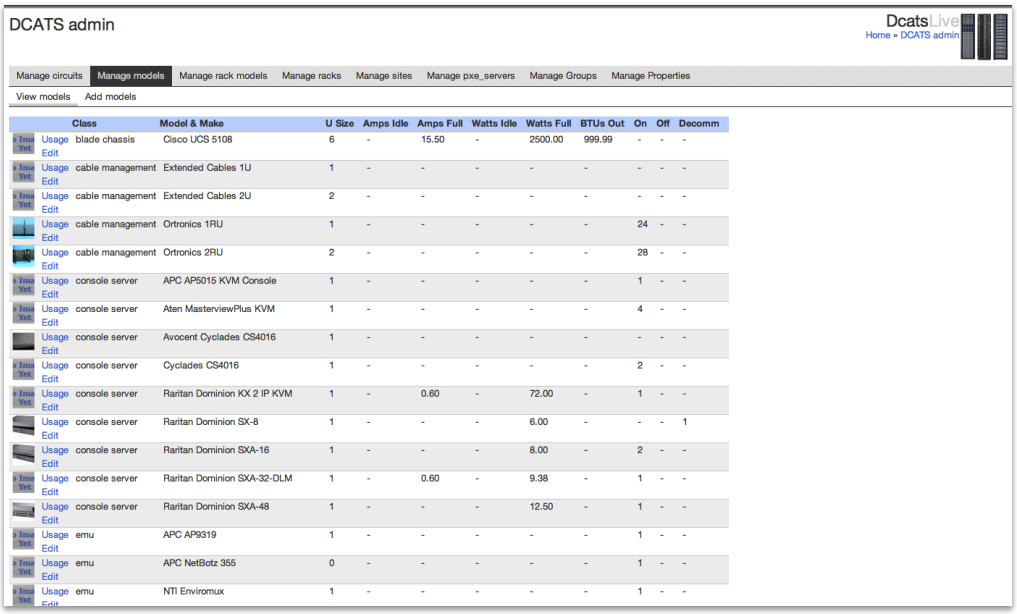
DCATS Device Editing

The Repository of Truth



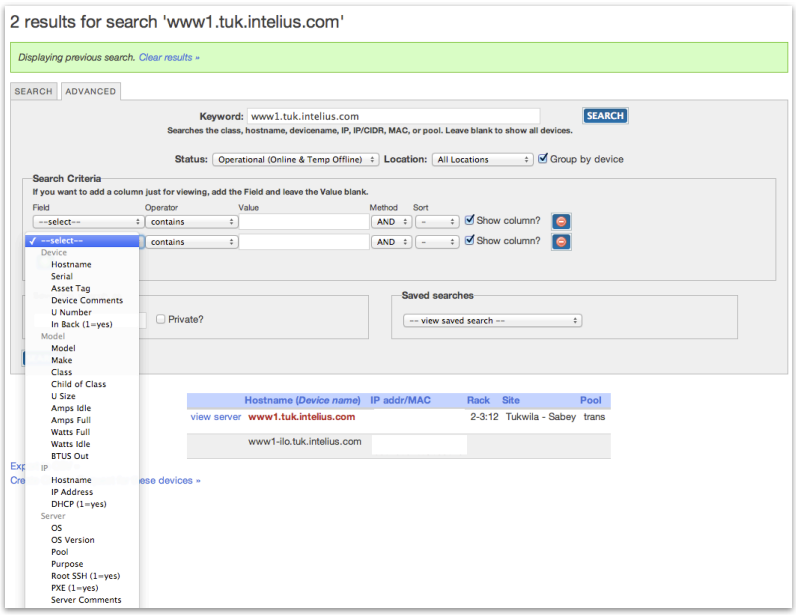
DCATS Device History

The Repository of Truth



DCATS Administration

The Repository of Truth



DCATS Advanced Search

The Repository of Truth

- Quote from 2009 Forrester Research report, "Knocking the NOC: Enter the New Operations Center" (p. 4)

> In all cases, however, it is best to make sure that all tools, regardless of their domains of control, cooperate with other related tools. A good case in point is the unifying role of the configuration management database (CMDB) or configuration management systems (CMS). In the design phases of a new or modifying service, SMEs and design tools must consult a common CMS from the very beginning. This presents a snapshot of reality on which one can build. This snapshot is modified in the tools to reflect the newly proposed situation. The new design then passes through the necessary change and release processes to move into production. When the changes are actually implemented, the CMS must be synchronized and reconciled in real time to maintain an accurate perspective of "the truth" within the CMS. If "the truth" is not true, then success is false.

Dependencies

Dependencies

- Epistemology
  - How do we know what we think we know?
  - How do we know if it's true?
- How do you ensure that the data in the truth repository is complete and accurate?
  - I can guarantee that if you're depending on human beings keeping this information up-to-date, you'll be sorely disappointed
- The answer: Dependencies
  - Dependencies are good, and you should actively create them

Dependencies

- Now I'm going to qualify that statement to make it less crazy
- "Hard" dependencies, that have real-time impact on functioning services, should be avoided
  - Loss of availability or correctness on the part of the truth repository shouldn't break a working service
- "Soft" dependencies are encouraged
  - Inaccurate or incomplete data in the truth repository *should* break processes
    - System builds, DNS additions, code releases, etc.

Dependencies

- Dependencies are good as long as you can detect and remediate root causes of problems before they get serious
- However, the dependencies must cause some pain; that's their purpose
  - –The pain alerts you to untruth in the repository, thus enforcing accuracy
    - How do we know that what we know is true? This is the key
  - –Without pain, you quickly accumulate untruth in the repository, and then lose trust in it
    - Without pain, people skip steps, forget or delay updates, etc.
- Dependencies lead to management by *fact* rather than hope

Automation

Automation

- Use automation to help enforce accuracy and completeness
  - –Build automation over time
- Three categories of automation
  - –Dependencies
  - –Data gatherers
  - –Consistency checks

Automation

- Dependencies
  - –Automatically build configurations, or implement processes based on data in the truth repository
  - –Examples:
    - Scripts to generate DNS zone files and DHCP configs
      - –Make the repository the source of truth for IP addresses (host IP addresses, at least)
      - –Include checks for duplicate IP assignments, malformed addresses, properly qualified domain names, valid subnets, etc.
        - »You may want additional checks for changes to existing records beyond those for creation of new ones
      - –If IP address data in the repository is incorrect, the script will fail, or the host won't get on the network, or something else will break
        - »Make failures obvious
    - Monitoring applications
      - –Generate host lists and groups from the truth repository for Nagios, Ganglia, etc.

Automation

- Dependencies (cont'd.)
  - –Examples:
    - Host management tools
      - –Data used by tools should come from the repository
      - –`rshall` is a tool used to manage UNIX/Linux hosts in parallel
        - »Build or modify an `rshall` extension to use the repository as its host data source
      - –If the data is incorrect, it'll probably be noticed in a frequently-used tool
    - Code release tools
      - –Determine hosts that need code by checking the repository for roles and status
    - Change management
      - –Tie CM systems into the truth repository
      - –Change-detection systems can reference lists of machines from the truth repository associated with CM requests
      - –CM requests can feed comments into the truth repository

Automation

- Data gatherers
  - –If the repository isn't the source of truth for particular data, gather that data from the source automatically whenever you can
  - –Examples:
    - Scripts to gather hardware models, installed RAM and CPU, serial numbers, OS revisions, etc., and update the truth repository
      - –Optionally, alert you to updates of incorrect information in the repository

Automation

- Consistency checks
  - –If the repository isn't the source of truth, and the truth can't be gathered automatically, at least check whether the data in the repository is plausible
  - –Examples:
    - Check racked device locations (rack and U)
      - –Check for valid formatting and values
      - –Two devices can't be in the same location
        - »Include consideration of device height
    - Check network switch or patch panel ports, or PDU sockets
      - –Check for valid formatting and values
      - –Two devices can't share the same port
      - –Single-port hosts can't multiple network/power connections

Review

Review

- You need a single *repository* of truth about machines
  - *Sources* of truth are multiple, but there should be a single source for each datum
  - Your truth repository may be called a machine database, a CMDB, or be part of a DCIM tool
- Dependencies help ensure truth
  - Actively create them, and make them painful (but not fatal!)
  - The goal is to manage by fact rather than hope
- Use automation to ensure accuracy and completeness
  - Consider automation of dependencies, data gathering, and consistency checks

Truth

On machine epistemology and the importance of dependencies

Leon Towns-von Stauber
Cascadia IT, March 2013
http://www.occam.com/sa/