Gaining control of complexity

The standard for the data center

technical white paper

This document describes the data center markup language (DCML), a structured XML-based format for describing the contents of data centers and the policies governing the management of those contents. DCML describes blueprints for constructing managed environments, the knowledge necessary to turn those blueprints into an environment instance and the resulting environment.

dcml Data Center Markup Language

Table of Contents

Authors:
Tim Howes
Chief Technology Officer
Opsware Inc.

Darrel Thomas
Chief Technologist
EDS Hosting Services

Introduction

Over the past five to 10 years, dramatic changes have occurred in the data center landscape. The proliferation of the Internet and the World Wide Web made it dramatically easier to develop and deploy applications. The introduction of multitiered Web-based application architectures caused a substantial shift of computing power from client back to server. The continuing trend toward smaller, cheaper servers resulted in a dramatic increase in the number of servers. The result of these trends has been an explosion of complexity and scale in the data center. Today's data centers often contain thousands or tens of thousands of servers, networking devices, storage devices and other special-purpose equipment, running an even greater array of operating systems, software, configurations and data.

How well are we handling data center complexity?

New methods and approaches

The information technology (IT) approach to managing the data center has not kept pace with these changes. IT departments are still using methods developed when data centers were significantly less complex. These methods rely on armies of experts to make manual changes to the environment and react to problems after they occur. The result for IT has been a staggering increase in cost and decrease in quality. Traditional data center management tools such as monitoring have not kept pace with these changes either, employing the same passive, manual frameworks implemented a decade ago. Advances among these tools have resulted in solutions that amplify the bad results of the manual approach.

This situation led to new approaches to data center management. Although these new approaches go by different names and are designed to solve different parts of the problem, they all share common characteristics and goals. Data center automation, utility computing, adaptive enterprise computing, on-demand computing and other efforts focus on replacing the traditional manual, passive approach to data center management with an automated, active approach. The active approach is designed to reduce costs through increased labor and equipment utilization, and to increase quality by eliminating human error and enforcing best practices. These new approaches do not replace the traditional monitoring, ticketing and other operational systems. Rather, they typically integrate with and build on those systems, making them more effective in the new data center environment.

These new approaches brought with them new requirements, on behalf of customers, for vendor independence and interoperability. Just as traditional management approaches led to the development of standards such as Simple Network Management Protocol (SNMP) and Common Information Model (CIM) for interoperable network monitoring and management applications, today's new automated management approaches require standards that support interoperability among solutions. The data center markup language (DCML) is a proposal for one such standard.

Introduction to DCML

In the automated data center, there is a need for a standard format to describe the contents of the data center and the policies governing the construction and management of that content. This need is evident in a number of scenarios:

- Construction and change – Customers often need to build or rebuild the full or partial contents of data centers. The build case occurs when new applications or infrastructure are deployed, during technology refresh cycles and when increasing the capacity of existing applications. The rebuild case occurs during disaster recovery, data center consolidation and migration, and during simple system recovery.

  Automated implementation of these scenarios requires two things. First is a comprehensive blueprint of the environment to be built. Hardware requirements, software components and configurations, networking and storage – all must be specified to facilitate automation. Furthermore, this description must be in a form adaptable to a variety of environmental situations. Second is the set of policies governing the building of the environment. Approved and banned versions and combinations of various technologies, the order in which software is installed and started, the blessed configurations to be used, the required support from underlying hardware – all must be specified to ensure the resulting environment conforms to standard best practices. Third is a description of the individual components that make up the environment itself. A DCML description is composed of these three components, as depicted in Figure 1.

  After it is constructed, the data center environment endures constant change. Patches are applied and removed, new versions of applications are rolled out and rolled back, configurations are tweaked, capacity is added and removed, and a host of other changes happen on a daily basis to support changing requirements.

  In an automated data center, these changes must be described in detail so they can be applied without human intervention. Changes are governed by strict policies aimed at minimizing downtime caused by disruptive changes and maximizing the agility with which good changes can be made. Furthermore, the result of these changes must be documented and communicated to monitoring and other systems.

- Visibility – Many useful and illuminating conclusions can be drawn from an accurate and up-to-date representation of a data center environment. Reporting on hardware and software usage in the data center is an obvious example. But such information can also inform data center planning decisions, procurement decisions and other business processes.

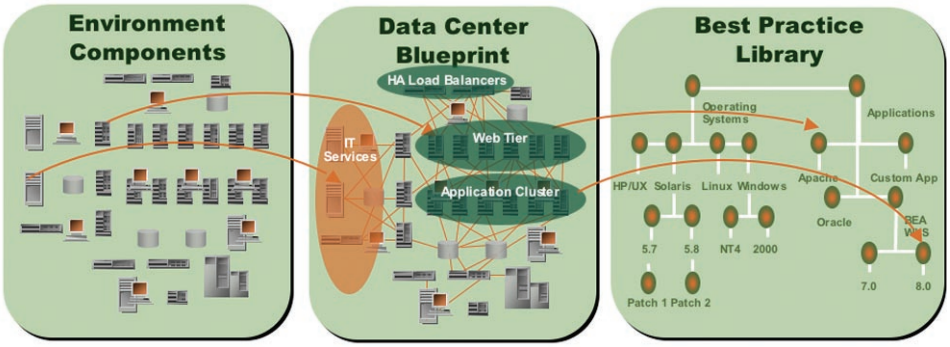

Figure 1: DCML scope includes environment components, data center blueprints, and best practice policies

For example, having a description of the requirements of applications slated to inhabit a given data center, one can infer the rack space, power and cooling requirements necessary to house the applications. Knowing the technology requirements represented in several data centers to be consolidated or migrated, one can infer the hardware and software that must be procured to facilitate the transition.

- Management – As with traditional data centers, the automated data center must be monitored to detect faults and track performance, track down and fix problems, and perform day-to-day management. Although the needs are similar to those of the traditional data center, the opportunity for improvement is huge. Traditional management systems suffer from not having an accurate and up-to-date view of the environment – something automated systems provide.

In all of these scenarios, the common thread is the need for knowledge of the configuration of a data center, application, network component, storage component or server. In some scenarios, the knowledge is generic, used to reproduce instances of an object. The first step in being able to reproduce something is to describe it. DCML provides a language for describing these data center elements, including applications, servers, software components, configurations, network and storage infrastructure, and hardware and data center requirements. Often, such descriptions may be exchanged offline in their entirety between data center automation tools or with traditional management systems.

In other scenarios, the knowledge is specific, used to keep other systems in sync with the current state of the environment as changes are made – for example, incremental changes to the list of servers that should be monitored. This use may imply a more incremental description exchanged with systems in a real-time environment. DCML supports both types of uses.

A DCML description provides the blueprint that enables an appropriate data center automation tool (or set of tools) to reproduce the infrastructure described or extract useful information from the description. Like an architectural blueprint, a DCML blueprint can be easily adapted to slightly different environments, such as hardware, networking and data center variations. DCML can also describe the key building blocks (software packages, configurations, etc.) and policies necessary to construct an actual environment based on the blueprint. Finally, DCML can describe the resulting environment itself. All of these descriptions can be used to extract information useful to traditional management systems, and to drive procurement and other planning processes.

Why a standard?

A standards-based approach to these problems provides a number of benefits. For the types of applications described above, multiple components from different vendors are almost always involved and must be described. A standard format enables all components to be described using a single format. Similarly, multiple automation tools may be required to reproduce the components being described (for example, an OS-provisioning tool, an application-provisioning tool and a network-provisioning tool). A standard format enables these tools to interoperate more easily. Furthermore, some scenarios – such as a move between hosting providers – imply transfer of control between autonomous organizational entities, a situation for which standards are tailor-made.

Comparison to traditional approaches

Traditionally, tasks such as those described above were accomplished by a variety of means:

- *Manually* – System administrators, database administrators, application experts and others use their memory, expertise and knowledge of the existing systems to reproduce them and communicate the resulting changes to other systems. There are numerous problems with this approach, including slow execution time, inconsistency and poor quality resulting from human error, out-of-date documentation and faulty memories of people.
- Ad hoc *and platform-specific standards* – Some platforms define *de facto* standards for representing certain types of information. For example, on Red Hat Linux, the RedHat Package Manager (RPM) package format is a *de facto* standard for representing software packages. Similar *de facto* standards exist on other platforms. Problems with this approach include a lack of cross-platform support and inability to represent the comprehensive set of information required across components. Even within platforms, often there is no standardization.
- *Imaging* – Disk images of the machines and their contents are created and used to reproduce the configuration when needed. Problems with this approach include the fact that images are tied closely to specific hardware, network and data center configurations. This makes it difficult to change any of those environmental variables when rebuilding. The storage management problem of organizing and retaining significant numbers of large, unwieldy images is daunting. In addition, imaging generally only applies to server elements, leaving networking and other critical infrastructure components unable to participate.

- *Backups* – The contents of disks are backed up to tape and often stored offsite for disaster-recovery purposes. If a disaster occurs, the tapes are retrieved and restored. Problems with this approach include the time required for backup and restore, and the difficulty getting a consistent, usable backup image that will restore to a useful state. Backup systems are generally better at selective file retrieval than wholesale system restoration.

These solutions also share the problem of not having a standard for representing the information they need to succeed. In the case of imaging, there is no standard image format across platforms, and sometimes no standard format even on a single platform. In the case of manual recreation, there is no standard for representing the information needed – only the knowledge contained in the heads of people and whatever they may or may not have captured on paper, in files, and other places. These problems make it impossible to reliably extract information needed for data center planning, inventory management and other functions among the key needs identified above. Clearly, given the diverse of software, hardware, vendors and applications involved, a standard format is needed.

Related standards

There are several standards either defined or in process to address related problems, but none of them can be used in place of DCML. These related standards are described briefly below.

- *CIM* – The Common Information Model is a Distributed Management Task Force (DMTF) standard for describing overall management information in a network/enterprise environment. CIM defines an abstract data model, method for instantiation in XML, and mappings to other management and information standards such as SNMP and Lightweight Directory Access Protocol (LDAP). CIM's focus is on providing end-to-end consistent management abstractions to be used by monitoring and other traditional management systems. Although CIM is quite comprehensive in this area and overlaps somewhat with DCML, CIM is not well suited to the data center automation problem, which is DCML's much narrower focus. Many of CIM's concepts and data elements can be mapped onto DCML for use with data center automation tools. Where DCML overlaps with definitions found in CIM, DCML references CIM to avoid duplicating work.
- *SDM* – Microsoft's Systems Definition Model (SDM) is a recently announced initiative complementary to DCML. SDM establishes a technical contract between development and operations. By providing a standard format for encoding Windows application component requirements, SDM can help automate the creation of a production server for Windows-based applications. However, while SDM provides operational requirements for an application component, DCML provides the blueprint for constructing and managing the entire environment in which that application is running. Ultimately, SDM component requirements will feed directly into the DCML-defined constraints for Windows applications.
- *ITIL* – The IT Infrastructure Library is a collection of standards defining best-practice IT processes. ITIL standardizes processes such as service desk, change management, service level management and others. DCML is an important implementation component of these ITIL processes in automated environments, making them more reliable, consistent and vendor-independent.
- *OSS/J* – The Operational Support System through Java initiative is defining a set of standard application program interfaces (API) used to communicate with and between IT OSS and business support systems. OSS/J defined APIs for service activation, trouble ticketing, billing and quality of service, with other APIs in process. When and if OSS/J defines interfaces to data center automation systems, DCML will provide important content communicated through those interfaces.

DCML concepts

A data center can be thought of as a complex entity composed of simpler lower-level entities. For example, a server's composed of an operating system, patches, various software components, configurations, and associated access rights and policies governing the use of the server. To instantiate such a server in a data center environment requires certain underlying support from the hardware, network and physical environment in which it lives. Included in the DCML description of a server are instructions for performing this instantiation. Note the difference between the DCML concept of a server (a description of a generic class that can be instantiated in a variety of environments) and the traditional concept of a server as designed by CIM and other standards (a description of a specific instance of a server). This concept of adaptability is central to DCML.

The concept essentially means that a DCML description is like a blueprint. The blueprint can be used to build the entity described in a number of different locations and environments. During the build process, blueprint variables (such as host name and networking information) are filled in and configured with values that make sense for the environment. In the case of reproductions on different hardware, DCML must describe the basic hardware requirements that cannot be changed and those that can. For example, Intel-compatible hardware is required to reproduce a Microsoft Windows server, but there may be flexibility in the exact manufacturer, model or amount of memory.

DCML accomplishes this adaptability through the use of requirement attributes and "marked-up" configuration files. A DCML-capable data center automation system uses the requirement attributes to determine hardware and other compatibilities. When configuring systems, the data center automation system uses the marked-up configuration file to substitute configuration values appropriate to the new environment.

The need for adaptability is clear in disaster recovery situations, data center migrations and consolidations, even when using DCML to describe additional server capacity. In all these scenarios, the entities described by DCML often need to be reproduced in environments requiring different hardware, different network configurations, different clustering configurations, different connecting components, and simple differences such as host name and IP address.

Another key DCML concept is the ability to represent best practices, standards and policies used in the management of a data center environment. Without the ability to capture and represent this kind of information, a data center automation or utility

computing solution is simply an "amplifier" for the bad (or possibly good) practices knowledge possessed by each individual. DCML allows application-, organization- and technology-wide best practices to be described and, therefore, interpreted by data center automation systems. These best practices relate primarily to software and their configurations, and are captured by the best-practices library portion of a DCML description. The DCML blueprint describes how to combine these best-practice standard configurations with the physical components to create a best-practices data center environment.

Another key DCML requirement is the ability to represent the wide variety of diverse technologies in today's (and tomorrow's) data centers without imposing requirements that those technologies be changed. This requirement is a practical one. Imposing new requirements on technologies to be managed has many advantages in theory, but actually getting those technologies changed poses serious challenges to the deployment of DCML. Because it imposes no requirements on the technologies it describes, DCML can be useful today, without the long adoption cycles that have accompanied other management standards such as SNMP and CIM.

Security

Although DCML is neither a security protocol or concerned directly with security itself, there often may be a need to represent sensitive information in DCML. For example, to reproduce a server, one might need to reproduce the password file contained on that server. To reproduce the state of an application, one might need to reproduce sensitive data maintained by that application.

DCML provides the mechanisms by which sensitive information can be separated from nonsensitive information and secured using appropriate means. For example, DCML enables information components to be signed to prevent tampering and/or encrypted to prevent unauthorized access. DCML does not define the mechanisms by which this occurs – it merely provides a framework in which it can be done.

DCML defines two constructs to enable the security of the information conveyed in a DCML document:

- *Encrypted* – The encrypted construct is used to encrypt the payload it carries to avoid unauthorized viewing. Any encryption algorithm may be used, provided it follows a few simple rules.
- *Signed* – The signed construct is used to sign the payload it carries to avoid unauthorized tampering. Any signature algorithm may be used, provided it follows a few simple rules.

The encrypted and signed constructs may appear at any level of a DCML document and may be nested within one another, enabling maximum flexibility.

Overview of the DCML specifications

Describing the entire contents of a data center in enough detail to be able to reproduce those contents is a daunting task, given the diversity of technologies represented in a typical data center. Yet, you do not have to solve the entire problem to start doing useful things. For example, you can solve the capacity management problem for many applications simply by describing servers, software components and configurations.

Therefore, we have taken an incremental approach to defining DCML, which is composed of the following set of specifications:

- *Framework specification* – This describes the overall DCML approach, formats and conventions used in subsidiary standards, and how DCML components are combined and related to one another. It also describes how DCML can be extended in the documents below and in future documents.
- *Hardware specification* – This specification describes how to represent server, network, storage and security hardware requirements in DCML. Because this type of information overlaps with work already done by CIM, this specification describes how DCML references CIM to represent the information.
- *Network specification* – It describes how to represent network topology, software and configuration information in DCML.
- *Storage specification* – This specification describes how to represent storage topology, software and configuration information in DCML.
- *Software specification* – It describes how to represent software components in DCML, their configurations and associated data. Software components include operating systems, patches, packaged software applications and custom software applications.

XML representation of DCML

There are many different ways to represent the information described above. We chose XML Schema as the language to describe DCML. XML has become the lingua franca of information exchange on the Internet. It is widely known, widely implemented, and well-understood technology. XML parsers and generators are readily available from a number of sources, making implementation cost relatively low. XML Schema is a convenient method of defining standard XML documents.

DCML components are defined using XML Schema, a standard way to represent the structure, content and semantics of XML documents. XML Schema is defined by the W3C.

Future DCML work

DCML is in its infancy. Much work is required to make it into a suitable standard. Nevertheless, early work on the specification and prototype implementation is promising. Our intention is to form a consortium of representative like-minded organizations willing to commit time and resources to the further development of DCML as an industrywide standard. That organization will submit the DCML specification to a standards body such as DMTF, IETF, OASIS, or SNIA. The specific organization will be chosen at the appropriate time.

About the Authors

Tim Howes

Tim Howes is co-founder, chief technology officer and executive vice president of development at Opsware Inc. (formerly Loudcloud), where he is responsible for product development and strategy. Prior to co-founding Opsware Inc., Tim held a number of senior technical positions at America Online, Inc. and Netscape Communications Corp. Before joining Netscape, he was a researcher and National Science Foundation project director at the University of Michigan, where he co-invented the Lightweight Directory Access Protocol (LDAP). Tim holds a Ph.D. in computer science from the University of Michigan.

Darrel Thomas

Darrel Thomas is chief technologist of the EDS Automated Hosting Services division, where he leads architecture, implementation and business-to-technology strategy. His responsibilities include acquisition strategy, and integration and automation evangelism across EDS. Previously, Darrel was the chief architect, designer and implementation lead for EDS' original Internet IT Hosting offering as well as the WebVault™, EDS' original leading-edge hosting environment. He was also chief technologist of Persona, Inc. – the first identity management and privacy services provisioning entity – which developed and provided online personal information (PII) and identity management services. Darrel holds a bachelor of science degree in computer science from Millsaps College in Jackson, Mississippi.

Contacts
Tim Howes, Chief Technology Officer
Opsware Inc.
599 N. Mathilda Avenue
Sunnyvale, CA 94085
phone: 408 744 7300
e-mail: howes@opsware.com

Darrel Thomas, Chief Technologist
EDS Hosting Services
5400 Legacy Drive
Plano, Texas 75024-3199
phone: 1 800 566 9337
e-mail: darrel.thomas@eds.com

About the DCML Organization

The DCML Organization is an open, independent, vendor-neutral, non-profit corporation being formed to create an open, freely licensed specification, Data Center Markup Language (DCML), and to encourage its broad adoption. DCML is the first standard that provides a structured model and encoding to describe, construct, replicate and recover data center environments and elements. DCML is designed to provide a mechanism to enable data center automation, utility computing and system management solutions to exchange information about the environment to make utility computing a reality. In addition to developing specifications, the organization intends to work with formal standards bodies, enable and administer certification and compliance programs, and perform user and market education. For more information about how to join the DCML Organization, or to learn more about planned activities and DCML, visit www.dcml.org.

dcml Data Center Markup Language