CHAPTER

THE CASE FOR STANDARDS 16

As cloud computing models emerge and evolve in a significant way to deliver reliable, automated services across private, hosted and public environments, standards provide a base of consistent, interoperable management across different cloud service implementations.
Brad Anderson, Microsoft General Manager, Management and Services Division

As every market matures, so evolves the need for standards. HP sees that the right balance between industry standards and proprietary technologies propels the industry forward, fostering collaboration and innovation.
James Mouton, Hewlett–Packard, Chief Technology Officer, Technology Solutions Group

It is not only about open standards, but it is about how these standards all play together for the benefit of the customer.
Angel Diaz, VP IBM Standards, Open Source and Cloud Labs

INTRODUCTION

One of the priorities of **application management** is monitoring and controlling application performance, security, and **total cost of ownership (TCO)**. To address the rising costs associated with these tasks and to further facilitate the management of applications, a number of industry, vendor, and neutral organizations have focused their efforts on creating standards and **management information bases (MIBs)**. Some of the most prolific groups to lead these efforts include the **Institute of Electrical and Electronics Engineers (IEEE)**, the **Internet Engineering Task Force (IETF)**, the **Desktop Management Task Force (DMTF)**, and the **International Organization of Standardization and International Electro-Technical Commission (ISO/IEC)**.

Standards establish consistent protocols that can be widely understood and adopted to facilitate performance, security, compatibility, and interoperability. Standards also simplify application development by speeding time-to-market and expediting the comparison of competing products. Standards are essential to ensure interconnectivity and interoperability requirements and verify new products and new markets. Without standards, application development, implementation, and management would be far less consistent and much more complex.

Over the years, numerous standards relevant to application management were developed. As changes occur in the nature of the applications themselves, the types of infrastructure on which they exist and the way in which they are managed (automated vs. manual) formed committees and task forces to address the ongoing need for new or updated standards. The following sections introduce the composition and purpose of each of the standards' organizations, and discusses the management standards they developed and their relevance to various aspects of application management.

Application Performance Management (APM) in the Digital Enterprise. http://dx.doi.org/10.1016/B978-0-12-804018-8.00016-4

MANAGING WITH INTERNET ENGINEERING TASK FORCE STANDARDS

The IETF is a large, open community of network designers, operators, vendors, users, and researchers that produce high-quality, relevant technical documents that influence the way people design, use, and manage the Internet. The IETF's standards development efforts are organized into several areas, which include applications and real-time, general, Internet, operations and management, routing, security, and transport. These working groups develop and review specifications for Internet standards in their area of expertise. The IETF standards process includes developing a specification that undergoes several iterations of review by the Internet community, making appropriate revisions, adopting the specification as a standard, and publishing the standard. According to the IEFT,[1] the goals of the IETF Internet standards process are:

- Technical excellence
- Prior implementation and testing
- Clear, concise, and easily understood documentation
- Openness and fairness
- Timeliness

More information on IETF membership requirements, processes, structure, and procedures can be found on their website at ietf.org.

Each IETF standard consists of a document that contains the standard's specifications. All documents published by IETF are designated as requests for comments (RFC) and allocated a unique identifier. The most popular and widely used IETF application management standards are **simple network management protocol (SNMP), system application MIB (sysApplMIB)**, and **application management MIB (ApplMIB)**.

SNMP NETWORK MANAGEMENT PROTOCOL (SNMP V1, V2, V2C, V3)

SNMP was first conceived as an application-layer protocol for exchanging management information between network devices. Since its inception in 1988, the various iterations of SNMP have gained widespread acceptance and proved to be a highly successful standard. SNMP v1 was initially created to provide network device monitoring for TCP/IP networks and defined in RFC 1155 and 1157. Subsequently, users became aware of functional deficiencies in SNMP v1, such as the inability to easily specify the transfer of bulk data and security deficiencies that included the lack of authentication and privacy mechanisms.

In 1993 SNMP v2, defined in RFCs 1441–1452, was released to address both the functional and security deficiencies in SNMP v1. Unfortunately, the security facility in SNMP v2 was criticized for deficiencies in the definition. Consequently, SNMP v2 was revised and released as SNMP v2c in 1996, defined in RFCs 1901, 1905, 1906, and 2578. The new version retained the functional enhancements of SNMP2 but eliminated the security facility. Instead, SNMP v2c contained a simple and unsecure password-based authentication feature, referred to as the community feature. Security was afforded a much greater emphasis in SNMP v3, defined in RFCs 2271–2275 and published in 1998. SNMP v3 defines a framework for integrating security features into the overarching capabilities of SNMP v1 or SNMP v3 and specifies a set of capabilities for network security and access control.

[1]The IETF Standards Process. Available from: ietf.org/about/standards-process-html.

The primary goal of the SNMP is to reduce the amount and complexity of management functions. SNMP achieves this by decreasing development costs for the management agent software required to support the protocol, which increases the degree of management functions remotely to allow best use of internet resources and impose the fewest possible restrictions on the form and sophistication of the management tools. It also simplifies the set of management functions to make it easier for developers to understanding and use. Another goal of SNMP is that the functional paradigm for monitoring and control be sufficiently extensible to accommodate additional, possibly unanticipated, aspects of network operation and management. Finally, it seeks to be independent of the architecture and mechanisms of particular hosts or particular gateways.[2]

Two key components of SNMP are the **object identifier (OID)** and **MIB**. The OID is structured and follows a hierarchical tree format, and the MIB describes the managed device parameters. Typically, the MIB contains a standard set of statistical and control values defined for hardware nodes on a network that forms the basis for various management queries. As the management community developed additional MIBs evolved, in 1998 the first application management MIB was published as part of the SNMP v2 Network Management Framework under the direction of the Application MIB Working Group.

The new working group bifurcated its efforts to first develop the sysApplMIB, defined in RFC 2287 and published in September 1998, followed by the development of the ApplMIB, defined as RFC 2564 and published in May 1999.

SYSTEMS APPLICATION MIB

The primary purpose of the sysApplMIB was to define the nature of the applications to be managed. In the sysApplMIB, an application is defined as, "One or more units of executable code and other resources, installed on a single host system."[3] Thus, by definition, the Application MIB Working Group did *not* limit the term "application" to a particular category of applications but instead made it generic to *any* type of application installed or running on a single system. The group also specified that an application be modeled both as a whole and as its individual elements, such as files and executables. In this way, the sysApplMIB models information regarding installed applications and their elements that are running or have previously run on the host and provides the necessary link for associating applications with their executing processes.

The purpose of the sysApplMIB is to provide a basic systems-level view of the applications and their components on a single-host system. The sysApplMIB also models activity information on applications and their components that are running or have previously run on the host system to link associating executing processes with the corresponding part of the application. The objects in the sysApplMIB are arranged into three groups (Systems Application Installed Group, Systems Application Group, and Systems Application Map Group). Each group has one or more information tables and/or scalars (see Table 16.1).

The sysApplMIB restricts the managed objects to information that can be determined from the system itself and does not require instrumentation within the applications to make the information available.

[2]A Simple Network Management Protocol, 1990. Available from: https://tools.ietf.org/html/rfc1157.
[3]MIB for Applications. Accessed from: http://www.ietf.org/rfc/rfc2287.txt.

Table 16.1 SysApplMIB Group Tables

Group	Table	Purpose
System application installed group	**1.** sysApplInstallPkgTable	List application installed on a particular host.
	2. sysApplInstallElmtTable	Provide information about executables and nonexecutable files or elements that collectively compose an application.
System application run group	**1.** sysApplRunTable	Contain application instances currently running on the host. Each time application is invoked, a new entry is created to provide information about that particular invocation of the application. Entries remain until the application instance terminates, then entry will be deleted from sysApplRunTable and placed in the sysApplPastRunTable.
	2. sysApplPastRunTable	Maintain history of instances of applications previously executed on the host. Entries are made when an invoked application from sysApplRunTable terminates. Size of this table is controlled by two scalars—sysApplPastRunMaxRows specifies maximum number of entries in the table, sysApplPastRunTblTimeLimit specifies the maximum age of table entries. Oldest entries are removed first.
	3. sysApplElmtRunTable	Contain an entry for EVERY process currently running on the host, not just those processes that are running as part of an identified application. Entry is created for each process at the time it is started and remains in the table until the process terminates. When processes terminate, only information from entries corresponding to elements of an identified application are moved to the sysApplElmtPastRunTable. If the process cannot be associated with any "parent" application, it is simply removed from the sysApplElemtRunTable.
	4. sysApplElmtPastRunTable	Maintain a history of processes previously executed on the host as part of an application. To control the size of this table, two scalars are defined—sysApplElmtPastRunMaxRows specifies maximum number of entries and sysApplElmtPastRunTblLimit specifies maximum age of entry of the table. Oldest entries are removed first.
	Scalars	Control size of each of the past run tables by number of entries and age of entries.
System application map group	**1.** sysApplMapTable	Provide a backward mapping to determine the invoked application, installed element, and installed application package given a known process ID number.

APPLICATION MANAGEMENT MIB

In May 1999, the Application MIB Working Group published the Application Management MIB (applMIB), defined in RFC 2564, to provide a more granular level of detail about the management objects and expand the sysApplMIB to include attributes that typically require instrumentation within the managed resource. Its purpose is to describe a basic set of management objects for fault, configuration, and performance management of applications from a systems perspective.

To ensure ease and speed of implementation while allowing room for growth, the applMIB defines a model for application information resident on a host computer that can be determined from the system itself and not from the individual applications. This system-level view of applications is designed to provide information about software applications installed and running on the host system without requiring modifications and code additions to the applications themselves. To support configuration,

fault, and performance management, the information described by the objects in the applMIB represent some of the basic attributes of application software from a nonapplication specific perspective. In this way, the applications are described as collections of executables and files installed and executing on a host computer.[4] The applMIB was last updated in March 2013.

To use SNMP to enhance application management, it is necessary to create an MIB to map the application attributes for monitoring and corrective action. With an MIB in place, SNMP can be used to query and set the attribute variables. When the state of a variable changes during normal operations, SNMP will trigger automated or manual management actions. In this way, the MIB becomes a base that is accessible to the entire management system and customized to support a specific application.

Numerous software products evolved using SNMP to troubleshoot network problems and distributed applications and ensure that critical systems, applications, and services are always available. These include Microsoft Network Monitor, Nagios, OpenNMS, Capsa Free, and Fiddler.

MANAGING WITH THE INSTITUTE OF ELECTRONIC AND ELECTRICAL ENGINEERS STANDARDS

The IEEE is the world's largest technical professional society with approximately 400,000 members worldwide. The IEEE is designed to serve professionals involved in all aspects of the electrical, electronic, and computing fields and related areas of science and technology. The main objective of the IEEE is to foster technological innovation and excellence through the educational and technical advancement of electrical and electronic engineering, telecommunications, computing, and associated disciplines. To achieve this aim, the IEEE has established itself as one of the leading standards-making organizations in the world and has developed more than 900 active standards with another 500 under development in a wide range of industries, including healthcare, power and energy, robotics, telecommunications, transportation, information assurance, and information technology.

Two of the most important IEEE standards that apply to application management are the **IEEE 1220 Standard for Application and Management of the Systems Engineering Process** and the **Portable Operating System Interface (POSIX) 1387.2**.

IEEE 1220—APPLICATION AND MANAGEMENT OF THE SYSTEMS ENGINEERING PROCESS

The stated purpose of the IEEE 1220 is to provide a standard for managing a system from initialization through development, operations, and disposal. The IEEE 1220 defines the systems engineering course as a generic problem-solving process that provides mechanisms for identifying and evolving the product and process definitions of a system.

The IEEE 1220 was first released for trial use in 1995. The first fully fledged **IEEE 1220** was originally developed in 1998, superseded by IEEE 1220:2005 in 2005, and reaffirmed in 2011.

IEEE 1220 defines the interdisciplinary tasks required through a system's lifecycle to change customer needs, requirements, and constraints into a system solution. In addition, the IEEE 1220 specifies the

[4]Application Management MIB. Accessible from: tools.ietf.org/html/rfc2564.

requirements for the systems engineering process and its applications throughout the product lifecycle. The six processes of the Systems Engineering Process model outlined in IEEE 1220 are:

1. Requirements analysis
2. Requirements validation
3. Functional analysis
4. Functional verification
5. Synthesis
6. Physical verification

These six systems engineering processes are connected by five control processes: data management, configuration management, interface management, risk management, and performance-based progress measurements. The focus of IEEE 1220 is on engineering activities required to guide product development while ensuring proper design to make it affordable to produce, own, operate, maintain, and dispose of without undue risk to health and environment. IEEE is widely used in the public and private sectors and has not changed significantly since its trial issue.

POSIX 1387.2 SOFTWARE ADMINISTRATION STANDARD

POSIX 1387.2 provides the basis for standardized software administration. Developed in 1995, POSIX 1387.2 is part of the IEEE POSIX series of standards for applications and user interfaces to open systems. POSIX 1387.2 consists of three main components:

1. **Standard application structures and application packaging layout**
   a. Defines a hierarchical set of structures that enable application developers to organize software files into management components
   b. Defines layout of distribution media containing packaged applications to provide portability of installation media including serial and directory access formats
2. **Standard utilities and application catalog information**
   a. Provides a standard interface for administering applications to avoid retraining of system administrators and simplify installation documentation
   b. Provides utilities to package application files into distributions using software packaging layouts, including utilities to install applications from distribution onto target systems, manage applications on the system, and manage distributions and intersystem distributed applications
   c. Creates an application catalog to drive the management utilities including listing, verifying, or removing an application throughout its lifecycle
3. **Distributed application administration**
   a. Defines concepts and utility syntax for application management in a distributed environment

Once an application is created, it must be logically organized and packaged for distribution. Packing, distributing, and controlling applications is a critical and time-consuming task for application developers and administrators. POSIX 1387.2 is a set of processes that guides an application from one state into another, for example, from development to distribution or distribution to postimplementation. To achieve this, POSIX 1387.2 addresses distributed software administration, standard software structures and software packaging layout, standard utilities and software catalog information and defines the roles relevant to application development, operation, and use. This structured set of definitions of processes and roles helps managers to consistently assign tasks and package and distribute any application, with a minimum of effort and expense.

Initially, the IEEE working group that created POSIX 1387.2 focused their efforts on Unix systems, but over the years they have widened its scope to apply to other operating systems, including those that support personal computers. Interestingly, while POSIX 1387.2 does not address the issue of interoperability, it does define the distributed environment by defining six distributed roles. To fully understand the distributed aspect of POSIX 1387.2, it helps to present these distributed roles[5] as provided in the POSIX 1387.2 rationale (see Fig. 16.1). In the distributed environment, each of these roles can be on the same system for local operations and a separate system for distributed operations.

A complete description of POSIX 1387.2 software structures, configuration scripts, software packaging layout, standard utilities, software catalog information, and distributed software administration can be found in "Foundations of Application Management"[6] or "Go Solo 2."[7]

MANAGING WITH THE TIVOLI SYSTEMS APPLICATION MANAGEMENT SPECIFICATION

The **Tivoli Application Management Specification (AMS)** is a roadmap for application developers to make applications management-ready. It does so by capturing information about an application in a central repository to enable an AMS-compliant management tool to manage the application. The AMS describes the following application management tasks:

- Structure and topology
- Deployment and distribution
- Installation
- Dependency checking
- Monitoring and events
- Verification
- Operational control

The core of the AMS is a definition of a set of **application description files (ADFs)** that contains all the information needed to effectively carry out the management tasks listed above and include the following information:

- **Topology**—description of each component of the distributed application and the relationships between the components
- **Distribution**—list of source files and directories
- **Installation**—custom scripts that run before or after application distribution
- **Dependency checking**—custom scripts that check hardware and software configurations of target systems to ensure application requirements are met
- **Monitoring**—specification of metrics and events accessible from the application and how to access them
- **Operational control**—custom scripts to perform an arbitrary set of operational control tasks

[5]POSIX Software Administration. Available from http://www.unix.org/version2/whatsnew/softadmin.html.
[6]Sturm, R., Bumpus, W., 1999. Foundations of Application Management. John Wiley and Sons, Ltd., New York, NY.
[7]Open Group, 1997. Go Solo 2: The Authorized Guide to Version two of the Single UNIX Specification. Josey, A. (Ed.).

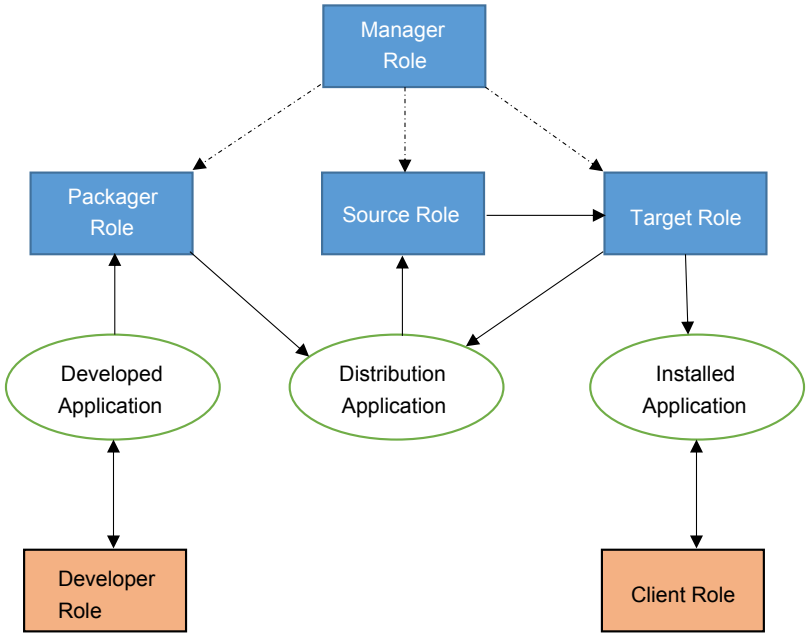

FIGURE 16.1

POSIX 1387.2 distributed roles.

- **Developer Role**—specifically outside the scope of the standard, in the developer role the application is constructed (developed software) into the form that can be accepted by the packager role. Activities performed by the developer role include compilation, source control, etc.
- **Manager Role—**provides for distributed control of the application administration process and tasks are initiated. The manager role is the one in which the administrator directly interacts through the command line interface provided in the standard or through a management application. The manager role directs the other roles to actually package, install, or manage the application.
- **Packager Role**—transforms application from locations where it exists after being developed into the standard packaging layout suitable for distribution. The packager role can package directly to media or to file system-based distributions that can later be copied to media, or from where installation can be performed directly.
- **Source Role**—serves the files in the application distribution source to the target role for actual installation onto a target system or for copying to a second distribution. The source role also makes logical software structures available to the manager role for user selection of software to be operated on.
- **Target Role**—actually installs the application and places it into a form in which it will eventually be used. Normally the application is installed on the system where the application is run, in which case the application is also configured. However, while this is usually the case, the target role is not necessarily on the system that will run the application but instead may be on a file server that serves the installed application to other client systems.
- **Client Role**—actually configures the application so that it is in a state where it can be executed. Configuration typically takes place on the system and architecture on which the application will be run. In this case, the standard supports configuration independent of installation. Installed applications may be subject to being configured several times. How the installed application is made available to the client role is also outside the scope of POXIS 1387.2. In addition to making the installed application files available to the client, an implementation must also make the installed application catalog information available so that clients can be configured by running configure scripts.

The ADF format is based on the Distributed Management Task Force's **Management Information Format (MIF)**, which is described in the next section. While the format of each ADF is common for all applications, the content of each set of ADFs is different since each set describes a different application with a unique set of characteristics. A typical ADF would include a single **global description file (GDF)** and multiple **component description files (CDFs).**

Information in the GDF includes:

- Application make
- Version identifier
- Manufacturer/author
- A free-form description of the application
- A reference to each application component
- Other miscellaneous fields of information that are global to the application.

Information included in each CDF includes:

- Component name
- Textual description
- Version identifier

Each CDF includes a reference back to the GDF to allow a management application to treat a distributed application as a cohesive whole. A stand-alone, single-system application supported by AMS has a single component (Fig. 16.2).

Administrators managing a distributed application need to understand both the type of application components and the relationship(s) among various components. AMS enables the application provider to define relationship specifications that include name, type, and related component type. With this relationship information, a management tool can provide an interface through which the administrator can view and modify existing application topology.

Distribution and installation are simplified by AMS support for automated software distribution, which breaks a distributed application into platform-specific components. In this way, the administrator can easily choose which parts of the application should be distributed and where the application should be installed. Component definitions include executables, data, scripts, etc. organized according to the target directory in which they reside on the target system.

In instrumented (i.e., management-ready) applications, in the component definition the administrator can include the types of programs required during the distribution or removal process, depending on

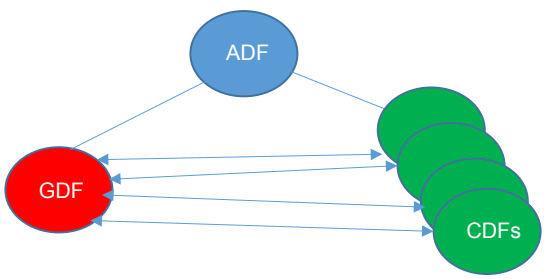

FIGURE 16.2

ADF structure for a distributed system.

whether additional programs need to run before, after, and during the process. Application types include:

- Before
- After
- Before removal
- After removal
- Commit
- On error

Other features of AMS allow the administrator to define application dependencies (both component-dependency instances and predefined dependency types), monitoring, and events and operational control tasks.

The AMS defines the following monitor types:

- Synchronous monitor (general)
- Asynchronous monitor (general)
- Log file monitor
- SNMP monitor
- Custom monitor definitions

Operational tasks include start, stop, backup or restore, and sending a message to a particular application user. Application provides can specify application-specific tasks, such as supporting a control task, by providing a program or script that, when executed, performs the desired function. The following information must be included in the CDF for each task:

- Name of task
- Description of task
- Command that, when executed on the system where the component resides, performs the control function
- Description of arguments required by the command

The AMS encourages creation of software element management information when the software is created and several vendors integrated AMS plug-ins into their products to automatically generate management data as part of software development. Tivoli submitted their work on AMS to DMTF to be used as input in the development of the Common Information Model. A full description of AMS can be found in *Foundations of Application Management.*[8]

MANAGING WITH DISTRIBUTED MANAGEMENT TASK FORCE, INC. STANDARDS

The **Distributed Management Task Force, Inc. (DMTF)** is an industry standards organization that works to simplify the manageability of mobile and cloud systems through open and collaborative efforts of leading technology companies that include Broadcom, CA, Dell, Emerson Network

[8]Sturm, R., Bumpus, W., 1999. Foundations of Application Management. John Wiley and Sons, Ltd., New York, NY.

Power, Hitachi, Hewlett Packard, Intel, Lenovo, Microsoft, NetApp, Software AG, TIM, and VmWare. The group has 160 member companies and more than 4000 active participants in 43 countries. DMTF develops and guides worldwide adoption of its interoperable management standards to enable the management of diverse traditional and emerging technologies. An interesting animation that shows the relationship between the various management technologies and DMTF standards can be found at: http://www.dmtf.org/managementtechnologiesdiagram. The three DMTF standards most relevant to application management are the **Common Information Model (CIM), Cloud Auditing Data Federation (CADF)**, and the **Web-Services Management (WS-MAN)** specification.

COMMON INFORMATION MODEL (CIM, CIM V2)

The Common Information Model (CIM) is a DMTF computer industry standard that methodically defines device and application characteristics to enable systems administrators and managers to control devices and applications across different vendors. In this way, products can be controlled based on the same kind of information (e.g., device name, model, serial number, network location, capacity, etc.). The CIM infrastructure approaches systems and network management by applying basic object-oriented structuring and conceptualization techniques. Using several defined Extensible Markup Language (XML) schemas, hardware and software vendors can supply CIM information about their product. CIM builds on the SNMP to provide relationship information to help track the source and status of application concerns. Originally released in 2005, CIM was most recently updated as CIM v2.45 in January 2016. The complete and most current CIM specification can be found at http://www.dmtf.org/standards/cim.

To use CIM to enhance application management, it is necessary to choose the CIM schema that reflects organizational needs, including the core schema. This CIM would reflect organizational ways of thinking that are applicable to all management areas, plus any common schemas that reflect on particular management areas of interest (e.g., applications, databases, security). If desired, extension schemas can be developed to represent organizational extensions of the common schema. Next, tools need to be selected to use this application management information. Finally, logical management systems that are understood and used by IT personnel operating the system on which the application resides would be constructed.

CLOUD AUDITING DATA FEDERATION

The Cloud Auditing Data Federation (CADF) is an open standard published by DMTF that was primarily designed to assure organizations, with cloud deployments, that their required security policies are as consistently managed and enforced in the cloud as they would be in the enterprise. Additionally, the CADF can provide accurate measurement data that can be used for SLA monitoring, real-time analytics, and problem diagnosis in cloud infrastructures.

CADF defines an event model for use in filling the essential data needed to certify, self-manage, and self-audit application security in the cloud and enables cross-vendor information sharing via its data format and information definitions. In this way, CADF delivers increased levels of insight into a provider's hardware, software, and network infrastructure used to run specific consumer applications in a multivendor environment in a private, public, or hybrid cloud.

CADF has a robust query interface that can be modified to reflect the unique resources of each provider. This standard also defines:

- Attachment of domain-specific identifiers
- Event classification values
- Tags to dynamically generate customized logs and report for cloud subscribers and customers

To enhance application management, CADF can be used to perform log-based periodic audits and meter, and monitor real-time performance to ensure quality of customer service. In doing so, it is important to demonstrate the value of CADF to the user. Show them that what they get is practical and that using a standard such as CADF is not merely an academic exercise using another new technology.

WEB-SERVICES MANAGEMENT SPECIFICATION

The Web Services Management (WS-Man) specification is a simple object access protocol for management-specific domains such as PCs, servers, smart devices, Web services, and other applications. It helps systems and network-based services collaborate seamlessly, provides interoperability between management applications and managed resources, and identifies a core set of web service specifications and usage requirements.

WS-Man addresses the cost and complexity of information technology (IT) management by providing a common way for systems to access and exchange management information across the entire IT infrastructure. It is used as a network access protocol by CIM-based management solutions, including **Desktop and Mobile Architecture for System Hardware (DASH)** and **System Management Architecture for Server Management (SMASH)**. WS-Man has the following capabilities:

- Get, put (update), create, and delete individual resource instances, such as settings and dynamic values
- Enumerate the contents of containers and collections, such as large tables and logs
- Subscribe to events emitted by managed resources
- Execute specific management methods with strongly typed input and output parameters

The WS-Man specification provides a standard for constructing XML messages using different web service standards such as WS-Addressing and WS-Transfer. These standards define XML schemas for web service messages. The messages refer to a resource using a resource URL. WS-Man adds a set of definitions for management operations and values. For example, WS-Transfer defines the Get, Put, Create, and Delete operations for a resource. WS-Man adds Rename, Partial Get, and Partial Put.

Version one of WS-Man was released in 2008, and WS-Man v1.2 was published in 2014. The objectives of WS-Man v1.2 include:

- Constrain web services protocols and formats to enable web services to be implemented with a small footprint in hardware and software management services
- Define minimum requirements for compliance with containing richer implementations
- Ensure backward compatibility and interoperability with MS-Man version 1.0 and 1.1
- Ensure composability with other web services specifications

In 2013, WS-Man was confirmed and published as an international standard by the ISO, designated as **ISO/IEC 17963:2013**. WS-MAN assists in application management by promoting interoperability between managed applications and managed resources. It also enables you to:

- Discover the presence of management resources, as well as provide navigation among them
- View and write to individual management resources, such as settings and dynamic values
- Obtain a list for contents of containers and collections, such as system components and log entries
- Run management methods

More information on implementing and deploying a WS-Man environment to remotely manage IT infrastructure can be found at http://www.dmtf.org/standards/wsman.

OPEN VIRTUALIZATION FORMAT

The OVF 2.0.0 standard published in August 2014[9] was formally known as DSP0243 **Open Virtualization Format (OVF)** V1.0.0. OVF 2.0.0 describes an open, secure, and portable format for the packaging and distribution of software run in virtual machines. Originally developed by DMTF, the OVF has been formalized as ISO/IEC as **ISO/IEC 17203:2011**. OVF provides a platform independent, efficient, open, and extensible packaging and distribution format that facilitates portability and deployment of virtual appliances and provides customers with platform independence. Once installed, an OVF package adds to the user's infrastructure a self-contained, self-consistent software application that provides a particular service or services. For example, an OVF package might contain a fully functional and tested web-server, database, and OS combination, such as a LAMP stack (Linux + Apache + MySQL + PHP), or it may contain a virus checker, including its update software, spyware detector, etc.

The OVF specification describes a hypervisor-neutral, efficient, extensible, and open format for the packaging and distribution of virtual appliances composed of one or more virtual systems. It aims to facilitate the automated and secure management not only of individual virtual systems but also of the virtual appliance as a functional unit. To be successful, OVF was developed and endorsed by a wide range of vendors. The OVF specification promotes customer confidence through the collaborative development of common standards for portability and interchange of virtual systems between different vendors' virtualization platforms. OVF is intended to be immediately useful, to solve an immediate business need, and to facilitate the rapid adoption of a common, backward compatible, yet rich format for packaging virtual appliances.

Within the OVF remit is the concept of the certification and integrity of a packaged virtual appliance. This concept allows the platform to determine the provenance of the appliance and permits the end-user to make the appropriate trust decisions. The OVF specification was constructed so that the appliance is responsible for its own configuration and modification. In particular, this means that the virtualization platform does not need to be able to read from the appliance's file systems. This decoupling of platform from the appliance means that OVF packages may be implemented by using any operating system and installed on any virtualization platform that supports the OVF format. A specific mechanism is provided for appliances to detect and react to the platform on which they are installed. This mechanism allows platforms to extend this specification in unique ways without breaking compatibility of appliances across the industry.

[9]DSP2017 – Open Virtualization Format White Paper. Available from: http://www.dmtf.org/standards/ovf.

The OVF format has several specific features that are designed for complex, multitier services and their associated distribution, installation, configuration, and execution:

- Provides direct support for the configuration of multitier applications and the composition of virtual systems to deliver composed services
- Permits the specification of both virtual system and application-level configuration
- Has robust mechanisms for validation of the contents of the OVF and full support for unattended installation to ease the burden of deployment for users and thereby enhance the user's experience
- Uses commercially accepted procedures for integrity checking of the OVF contents through the use of signatures and trusted third parties. This serves to reassure the consumer that an appliance was not modified because it is signed by the creator of the appliance. This assurance is seen as critical to the success of the virtual appliance market and to the viability of independent creation and online download of appliances
- Allows commercial interests of the appliance vendor and user to be respected, by providing a basic method for presentation and acknowledgment of licensing terms associated with the appliance

A detailed description of OVF 2.0.0 is available from http://www.dmtf.org/standards/ovf.

OVF can be used in many situations, for example, by software vendors to produce an application, data center operators to transport an application from one data center to another, and customers to archive an application.

MANAGING WITH ASL-BISL FOUNDATION STANDARDS

The **ASL-BiSL Foundation** is a meeting place (http://aslbislfoundation.org) for professionals with a common interest to encourage improvement in working methods and promote the exchange of best practices. The ASL-BiSL Foundation contributes to international standards for application management and information management organizations, in collaboration with ISO/IEC and NEN, the Dutch Standardization Institute.

Sound guidance for application management and development can be found in the Application Services Library (ASL) that is currently managed by the ASL-BiSL Foundation.

APPLICATION SERVICES LIBRARY

The ASL consists of a framework of processes and a library of best practices in the area of application management. Closely related to the Information Technology Infrastructure Library (ITIL) that supports service management, the ASL is a framework whose main objective is to professionalize the field of application management by supporting application management best practices.

The vendor-independent library consists of a set of books that describe best practices in the IT industry. ASL includes all processes and activities required to keep current the functionality of an application for the lifetime of the business process it supports. Originally developed in the Netherlands in

2002, the current version, ASL2, was published in 2009 in the Netherlands and released in English in 2012. ASL2 consists of:

- Standard terminology
- Generic descriptions of all ASL processes including process inputs and outputs, activities within the processes, relationships between process, and the roles of those involved in executing the processes
- Templates for important documents, such as, annual plans, management plans, SLAs, agreements and procedures, etc.

The structure of the ASL framework has not changed significantly since its first release. The main differences between the two versions relate to the positioning of internal and external IT vendors that affect how application management processes are implemented currently and in the future due to fluctuations in supply and demand. ASL2 consists of 26 processes, contained in six clusters across three levels of application management (Fig. 16.3).

The operational level consists of eleven processes that fall into three clusters to provide guidance in the operation processes for services and applications:

1. Application Support—four processes (use support, continuity management, configuration management, and IT operations management) support the day-to-day use of IT to ensure that applications perform as expected
2. Connecting Processes, Operational Level—two processes (change management and software control/distribution) focus on synchronizing service organization/operations and development and maintenance with an emphasis on transferring from day-to-day operations to maintenance and vice versa
3. Application Maintenance and Renewal—five processes (impact analysis, design, realization, testing, and implementation) guide application design, programming, and testing to ensure optimal availability of applications currently being used to support business processes with a minimum of resources and disruption and defines and discusses changing an application in response to disruptions and new requirements from defining consequences of a change request to supporting the customer in acceptance testing

The managing level has a total of five processes that fall into one cluster: management processes. The five processes of the management processes cluster (contract management, planning and control, quality management, financial management, and supplier management) are used to manage the activities that occur within the three operational level clusters. In ASL2's management processes, all areas of application management are controlled and managed and agreements with customers and suppliers are defined.

The strategic level consists of 10 processes that fall into two clusters:

1. Applications Strategy—five processes (IT development strategy, customer organizations strategy, customer environment strategy, application lifecycle management, and application portfolio management) focus on future application demands of the application portfolio of customers
2. Application Management Organization Strategy—five processes (account and market definition, capabilities definition, technology definition, supplier definition, and service delivery definition) deal with the creation of their own organization's management strategies for maintaining and enhancing necessary skills, capabilities, markets, and customers

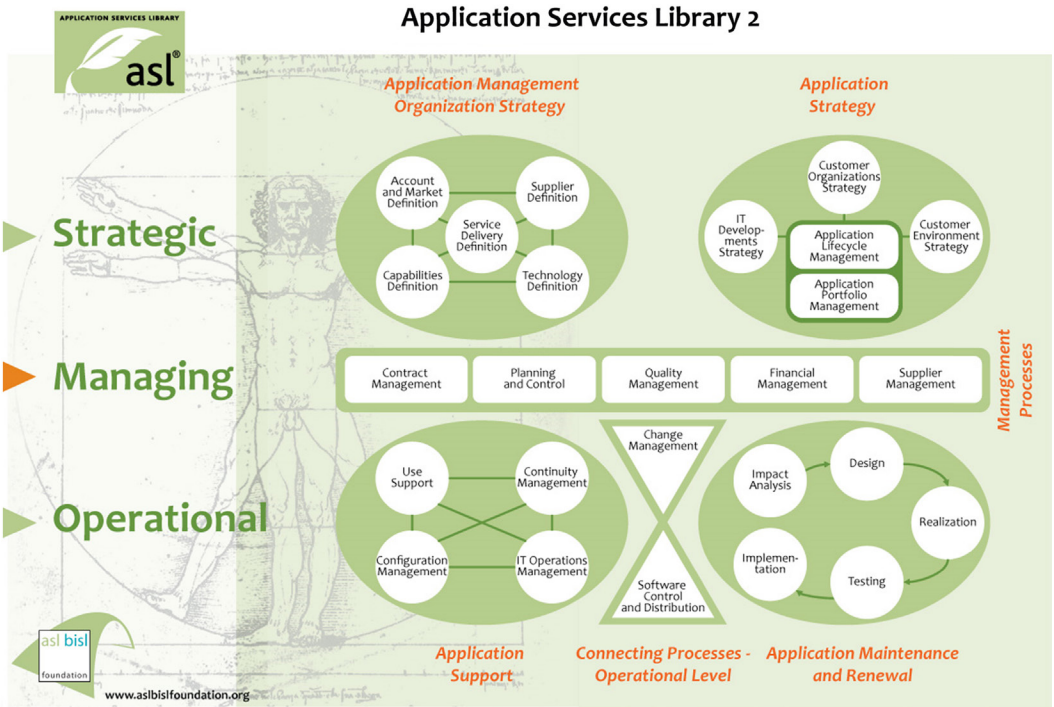

FIGURE 16.3

ASL 2.

www.aslbislfoundation.org.

For use in conjunction with ASL2, the ASL Foundation has developed an ASL2 Maturity Model (Fig. 16.4) to enable organizations to measure their progress in adopting and using the 26 processes defined in the ASL2 framework.

Similar in purpose and scope to the capability maturity model (CMMI) used to evaluate the current state of business processes, the ASL2 maturity model is a useful tool for application managers charged with making process improvements across a project or division or an entire organization.

The ASL framework helps IT professionals work and gain greater credibility for their profession. It can be used for all forms of application management such as outsourced and internal IT services. Based on best practices, the framework serves as a guide for the division of tasks and defines processes to direct the execution of those tasks. ASL can also be used to provide guidance to help structure and visualize activities that take place within application management and identify gaps between what should be done and what is done. In addition, ASL is a useful communication tool. For example, by providing a common set of terms and a clear definition of concepts and activities, ASL can facilitate communication between hardware and application management staff.

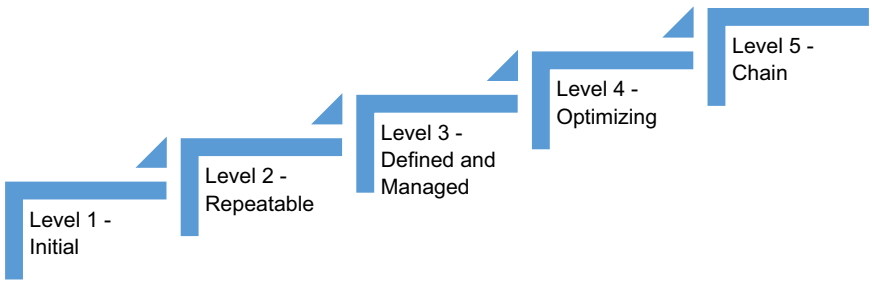

FIGURE 16.4

ASL2 maturity model.

MANAGING WITH ISO/IEC STANDARDS

The ISO and IEC are two international organizations that form a specialized system for worldwide standardization (ISO/IEC). National bodies that are members of the ISO or IEC develop international standards for particular fields of technical activity through technical committees established by the respective organization. These technical committees work together in fields of mutual interest and collaborate with other international government and nongovernment organizations to develop ISO/IEC standards. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. The ISO/IEC standards that are of greatest interest for application management are **ISO/IEC 16350** for Application Management, ISO/IEC 17203:2011 (formerly known as the Open Virtualization Format (OVF) developed by DMTF), and ISO/IEC 17963:2013 that was described earlier as DMTF's WS-MAN.

ISO/IEC 16350 FOR APPLICATION MANAGEMENT

A closer look at the standards discussed so far reveals an emphasis on the initial stages of application development. Few of these frameworks and standards address the operation, maintenance, and use of applications where the majority of TCO is spent. ISO/IEC 16350 was developed to address this shortfall in application management standards. In 2007, the Dutch Standardization Institute, NEN, introduced NEN 3434. This Dutch Standard for Application Management was developed in cooperation with the ASL BiSL Foundation and is heavily based on ASL. With the help of an international study group, the Dutch standard NEN3434 evolved into ISO/IEC 16350 for Application for Management, the international standard for application management published in August 2015.

ISO 16350:2015 establishes a common framework for application management processes using well-defined terminology referenceable by the software industry. ISO/IEC 16350:2015 contains processes, activities, and tasks that apply during the stage of operation and use. It is written from the point of view of the supplier organization that enhances, maintains, and renews the application software and the software-related products such as data-structures, architecture, designs, and other documentation. It applies to supply, maintenance, and renewal of applications, whether performed internally or externally with respect to the organization that uses the applications.

The structure of ISO/IEC 16350 is almost identical to that of ASL2. ISO/IEC 16350 has 26 processes, six clusters across three levels of application management, and essentially elevates the concepts of ASL from those of a guiding framework to the status of an international standard.

ISO/IEC 17203: 2011

Originally released by DMTF as the Open Virtualization Format (OVF) in 2009, ISO/IEC 17203 offers a standard packaging format for virtual machines to address a critical business need of software vendors and cloud service providers. ISO/IEC 17203:2011 describes an open, secure, portable, efficient, and extensible format for the packaging and distribution of applications to be run in virtual machines. The focus of this standard is on the software engineering activities that are involved in software development. It also ensures proper design to make the application affordable to produce, own, operate, maintain, and retire.

OPEN STANDARDS

In addition to the standards organizations discussed, a number of other groups have developed some interesting open standards that are important to enhancing application management. Described next are Open Group's Applications Response Measurement (ARM) and the **Cloud Application Management for Platforms (CAMP)** initially developed by an industry consortium and confirmed by the **Organization for Advancing Open Standards for the Information Society (OASIS)**.

APPLICATIONS RESPONSE MEASUREMENT

Application Response Measurement (ARM) is an open standard initially developed by Tivoli and Hewlett–Packard in 1996 and published by The Open Group later that same year as a specification entitled, "Systems Management: Application Response Measurement (ARM) API" v1.0 (API = application programming interface). The Open Group is an international vendor-neutral consortium upon which buyers and suppliers of technology can rely to lead the development of IT standards and certifications. The objective of The Open Group is to ensure interoperability and vendor neutrality, provide guidance in an open environment, and provide organizations with access to key industry peers, suppliers, and best practices. Founded in 1996 by X/Open Company Ltd., and the Open Software Foundation, The Open Group is supported by most of the world's largest user organizations, information systems vendors, and software suppliers. The Open Group operates in all phases of open systems technology lifecycle including innovation, market adoption, product development, and proliferation. The Open Group published a diverse assortment of technical documentation focused on technical standards and product documentation.

Subsequently, the ARM v2 (API) specification was issued in 1997 and finally approved for publication as an Open Group Technical Standard in March 1998. ARM v4 was released in 2003 and revised in 2004, followed by ARM v4.1. The current version of the ARM standard is ARM 4v2. An ARM 4v2 Software Development Kit (SDK) using Java and C bindings and a library of convenience function API calls used to manipulate ARM four structures are available for download from opengroup.org.

The purpose of ARM is to monitor and diagnose performance bottlenecks of loosely coupled or service-oriented enterprise applications in a single system or distributed environment. The ARM

standard describes a universal method for integrating applications as manageable entities by allowing users to engage in end-to-end transaction and application response time monitoring and performance management. ARM monitors user-visible business transactions and those visible only within the IT infrastructure by embedding simple calls into an application that are captured by an agent supporting the ARM API in either C or Java. Timing information from each step in a processing transaction is logged to a remote server for later analysis. This captured data allows the application to be monitored and analyzed for availability, service levels, and capacity.

To help organizations comply with the ARM standard, many applications have already instrumented some of their offerings with ARM calls, including:

- Apache HTTP Server
- Baan 5
- IBM WebSphere Application Server
- IBM HTTP Server
- IBM DB2 Database Server
- IBM Tivoli Access Manager
- Mozilla Firefox
- SAS

ARM is easy to use. It uses a small number of calls and parameters and can be called from many different programming languages. To use ARM, the sequence of steps shown in Fig. 16.5 and described next should be followed.

Identify and Select Transactions—before an application in a production environment can be monitored and analyzed, the units of work must be identified and selected. In ARM, a unit of work is referred to as an ARM transaction. Typical examples are transactions initiated by a user and transactions with servers. Transactions selected should be ones that need to be measured and monitored and for which corrective action can be initiated if performance is subpar. Next, the application must be instrumented with calls to the ARM interface.

Instrument the Application—when the application is initialized, define transaction names and register definitions with the ARM implementation using the appropriate API to instrument the client and server applications. Once registered, the names can be used to measure transactions by judiciously inserting Start and Stop calls that define the start and end of important business transactions to the ARM

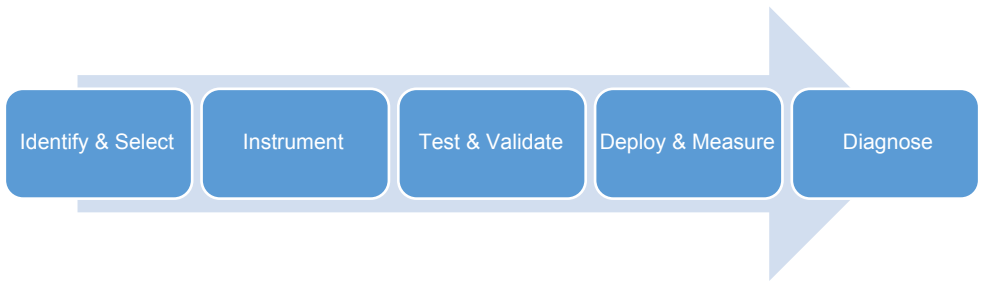

FIGURE 16.5

Application Response Measurement (ARM) v4.1

API. To build the correlation between the client and server measurements, the ARM start call creates an ARM correlator token to pass to the server within the SOAP-request measurement.

Test and Validate Measurements—provide a meaningful transaction name to test. Confirm that measured responses times are within an expected range of values. Provide the status of the transaction (i.e., success or fail). Pass the appropriate parent correlator to the child(ren) correlator(s).

Deploy and Measure Transactions—deploy instrumented applications with an ARM interface. Measure selected business or technical ARM transactions. Measurements are processed and forwarded to a configured database.

Diagnose Measurement Data—analyze specific transaction measurements and review summary of all measured transactions. Identify users on a per-transaction basis. Implement appropriate corrective action.

Many of the standards discussed so far were created by developers as an engineering exercise. The ARM API provides unique capabilities that other solutions cannot provide because it focuses heavily on the business. As a result, customers have shown a strong interest in ARM, which translates into the need for vendors to integrate an ARM API into their products to remain competitive.

CLOUD APPLICATION MANAGEMENT FOR PLATFORMS

CAMP is a platform-as-a-service management API specification designed to facilitate the management of applications in public and private clouds throughout their lifecycle. It is the first attempt to standardize the PaaS management interface by providing a common basis for developing multicloud management tools together with a representational state transfer-based approach to application management for vendors and consumers.

CAMP specification v1.0 was developed in 2012 by a consortium of vendors consisting of CloudBees, Cloudsoft, Huawei, Oracle, Rackspace, Red Hat, and Software AG.

One of the major pain points for PaaS consumers is portability, that is, the need to use a different plug-in for applications produced by different vendors due to the diverse proprietary APIs that individual providers use. Existing PaaS offerings use different languages (Java, Python, Ruby, etc.), frameworks (Springs, Rails, etc.), and APIs.

CAMP provides common development terminology and an API that can operate across multiple cloud environments without excessive adaptation. Using CAMP organizations can redeploy applications across cloud platforms from different vendors, thus reducing the effort required to move applications and provide service assurance through interoperability.

It was widely recognized that in order to enable the PaaS space to evolve, the problem of divergent application management APIs needed to be addressed. Since the features of APIs are not viewed as differentiators of PaaS products, the consortium saw the opportunity to achieve industry consensus on a generic language, framework, and platform neutral application and platform management API to effectively address the growing concerns around portability and interoperability in mobile and cloud systems. CAMP address these concerns in the following ways:

- **Portability**—CAMP portably migrates applications between platforms by taking application artifacts and putting some metadata around them to construct a package and deploy it. CAMP can also export a package from one platform and deploy it on another platform.
- **Interoperability**—CAMP supports management of applications and their use of the platform by defining basic application lifecycle operations (upload, deploy, configure/customize, start/stop, suspend, restart, and delete an application) and enables monitoring of the applications.

In August 2012, the consortium submitted the CAMP v1.0 specification to the Organization for Advancing Open Standards for the Information Society (OASIS), a nonprofit consortium that drives the development, convergence, and adoption of open standards for the global information society. OASIS develops international standards focused on security, Internet of Things, cloud computing, energy, content technologies, emergency management, and other IT areas.

OASIS established the OASIS CAMP Technical Committee to refine CAMP v1.0 and produce an OASIS Standard specification. The OASIS CAMP Technical Committee consisting of Cloudsoft, Fujitsu, NetApp, Oracle, Rackspace, Red Hat, and Vnomic came together to refine the CAMP v1.0 specification by further leveraging the similarities in current PaaS APIs. The OASIS CAMP TC also solicited input from the public at large through two public reviews in August 2013 and February 2014. Feedback received from the reviews led to a simplified resource model as shown in Fig. 16.6, a single-step deployment process and resource type inheritance. Subsequently, CAMP v1.1 was published in November 2014, and nCAMP, a proof-of-concept implementation of CAMP v1.1, was developed by the OASIS CAMP Technical committee to test the concepts and constructs of CAMP v1.1.

The CAMP specification consists of three components:

1. **Domain Specific Language**—defines application artifacts, services needed to execute them, and relationships between artifacts and services
2. **Resource Model**—represents the applications and their components, supports interoperability between diverse PaaS offerings, HTTP-based, REST-based protocol using JSON, highly extensible for future evolution
3. **Packaging Format**—moves applications between clouds, supports ZIP, TAR, or TGZ, YAM metadata, and is highly extensible

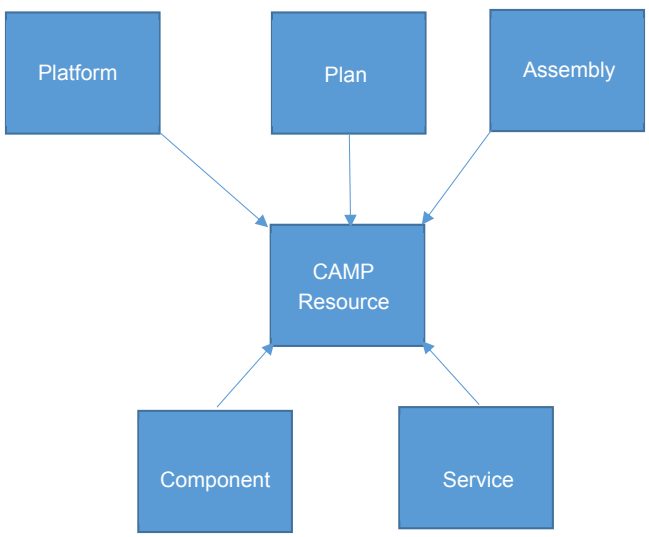

FIGURE 16.6

CAMP v1.1 resource model.

Although CAMP is not yet an official OASIS standard, it shows great potential for enhancing application management by addressing interoperability and portability issues in the cloud and as such is worthy of inclusion in this chapter. When addressing these two issues, creating a complex specification/standard is time consuming and labor intensive. Interoperability in particular has a large number of issues that need to be addressed. The OASIS CAMP TC is attacking the small problems first to gain "quick wins" by currently limiting CAMP v1.1's scope to a minimal set of known problems such as extensibility and versioning in the hope that those providers who adopt CAMP will extend it to fit their own applications. In this way, the OASIS CAMP TC will be able to integrate those extensions into upcoming versions of CAMP to improve the specification even further.

SUMMARY

The importance of standards in application management is reflected by the large number of standards that are evolving and the commercial and open-source applications that already use them. These applications benefit from the interoperability and portability the standards provide, along with shared functionality provided by numerous vendors that are developing tools to capitalize on the power of these application management standards.

Increasingly, software applications are the result of the integration of a diverse collection of executable and nonexecutable components dispersed over a network. Different components are often provided by different vendors that simultaneously compose different systems. These components can change rapidly and independently, making it difficult to deploy and manage a system in a consistent manner. Two key factors for the applicability and interoperability of the system are the completeness and standardization of the information structures that support the system. Standards are one way to effectively and efficiently address these issues.

The standards discussed here differ primarily in their depth and breadth of coverage. For example, early standards focused on the initial stages of application development where they proved useful in establishing a common set of terminology, processes, and roles associated with application management. Viewed over the past 25-plus years, it can be seen that standards as a whole provide a consistent and holistic approach to managing the entire lifecycle of diverse application portfolios (see Table 16.2).

As mobile and cloud computing continues to mature, even more standards will emerge. For example, VMWare recently voiced an interest in becoming a new entrant into Application Management Standards in the mobile environment. As greater numbers of standards created by industry task groups such as IETF, DMFT, and ASL-BiSL are recognized by international organizations such as ISO/IEC, it is expected that the use of standards for application management will gain even greater favor among the following groups who are charged with increasingly complex application development, operation, and management responsibilities:

- Business application owners
- Individuals responsible for application-dependent business processes
- Systems integrators
- Technical staff, including members of an application support team
- Heads of systems development functions
- System developers
- IT auditors

Table 16.2 Application Management Standards at a Glance

Year	Name of Standard	Developer	Objectives
2015	ISO/IEC 16350	ISO/IEC	Establish common framework for application management processes using well-defined referenceable terminology
	CIM v2	DMTF	Update CIM v1
2014	CAMP v1.1	OASIS	Refine resource model, create a single-step deployment process, provide resource-type inheritance
	CADF	DMTF	Address need to certify, self-manage, and self-audit application security in the cloud and enables cross-vendor information sharing via its data format and information definitions
	WS-Man v1.2	DMTF	Constrain web services protocols and formats to enable implementation of web services with a small hardware and software management services footprint
2013	ISO/IEC 17963:2013	ISO/IEC	Confirm WS-Man as an international standard
2012	ASL2 (English version)	ASL-BiSL	English version of ASL2
	CAMP v1	CAMP TC	
2011	ISO/IEC 17203	ISO/IEC	Formalize OVF
	IEEE1220-2011	IEEE	Reaffirm IEEE 1220–2005
2009	ASL2 (Dutch version)	ASL-BiSL	Extend ASL to address positioning of internal and external IT vendors and how they impact implementation application management processes now and in the future due to fluctuations in supply and demand
	OVF	DMTF	Offer standard packaging format for virtual machines to address critical business needs of vendors and providers
2008	WS-Man	DMTF	Provide interoperability between management applications and managed resources and identify a core set of web service specifications and usage requirements
2007	ARM v4.1	Open group	Add new capabilities (e.g., instrumentation control interface to query an implementation) to determine amount of useful transaction information, interfaces to improve ARM's usefulness in messaging and workflow
2005	CIM v1	DMTF	Methodologically define device and application characteristics to enable systems administrations and managers to control devices and applications across different vendors
	IEEE 1220–2005	IEEE	Reaffirm IEEE 1220–1998
2004	ARM v4	Open group	Richer and more flexible than ARM v3 (e.g., specify identity of application and transaction), reports changes in transaction attributes on a per-instance basis, bind a transaction to a thread
2002	ASL (Dutch version)	ASL-BiSL	Professionalize field of application management and provide a vendor-independent library of a set of books that describes application management best practices

Continued

Table 16.2 Application Management Standards at a Glance—cont'd

Year	Name of Standard	Developer	Objectives
2001	ARM v3	Open group	Increase flexibility of ARM v2, e.g., add ability to identify a user on a per-transaction basis, add Java bindings
1998	SNMP v3	IETF	Define a framework for integrating security features into overarching capabilities of SNMP v1 and SNMP v2 and define specific set of capabilities for network security and access control
	IEEE 1220–1998	IEEE	Describe systems engineering activities and process required throughout a system's lifecycle to develop systems meeting customer needs, requirements, and constraints
1997	ARM v2	Open group	Add ability to correlate parent and child transactions and collect other transaction measurements using C bindings
1996	SNMP v2c	IETF	Added a simple and unsecured password-based authentication "community" feature to SNMP v2
	ARM v1	Tivoli/HP	Measured response time and status of business transactions using C bindings
1995	Tivoli AMS	Tivoli	Provide central repository of information about an application to enable a management tool to manage the application
	POSIX 1387.2	IEEE	Define application layout, information about application and set of utility programs to manipulate the application and its information in stand-alone and distributed computing environments
1995	IEEE 1220 trial use	IEEE	Provide a standard for managing a system from initial concept through development, operations, and disposal
1993	SNMP v2	IETF	Improve SNMP v1 by addressing functional and security deficiencies of SNMP v1
1988	SNMP v1	IETF	Provide network device monitoring for TCP/IP networks

- Vendors
- Policy administrators
- Chief information officer (CIO)
- Chief finance officer (CFO)
- Chief compliance officer (CCO)
- Chief information security officer (CISO)

It is important to remember that when managing with standards, it is not necessarily a question of choosing just one standard. Depending on the specific needs of a given organization, several standards may be used together to gain most improvement in consistency and quality of management across the application lifecycle. According to the ISO, standards "facilitate trade, spread knowledge, disseminate innovative advances in technology, and share good management and conformity assessment practices."[10]

[10]ISO Standards – What's the Bottom Line? Accessible from: http://www.iso.org/iso/home/standards/benefitsofstandards.htm.

In application management, standards enable IT professionals to benefit from the expert opinion without having to call on those experts directly. Standardization also begins to simplify the increasingly complex application management environment, particularly the issues of interoperability, security, and portability. As the number of mobile and cloud applications increases, compliance with standards in product development will provide application managers and other IT professionals with insights for making better decisions about product choice and the management of their diverse application portfolios.

KEY TAKEAWAYS

- Key focus of cloud standards is on interoperability, portability, and security.
- Standards represent the knowledge and skills of a large number of industry leaders resulting in best practices.
- Standards facilitate application instrumentation. An application that is instrumented for management reduces its TCO.
- Early standards focused on the initial stages of the application lifecycle, but the existing suite of standards now covers the entire lifecycle.
- As mobile and cloud computing evolve further and the proliferation of available applications to support this complex environment increases, standards-based management solutions become even more critical to facilitate a seamless user experience.
- Since standards are very much an engineering exercise, it is important to keep the user perspective and requirements in mind.