NEW IN VERSION 4 – SLAs

Introducing:
Service Level Agreements

Cloud Computing Use Cases White Paper

Version 4.0

A white paper produced by the

Cloud Computing Use Case Discussion Group

Cloud Computing Use Cases

A white paper produced by the Cloud Computing Use Case Discussion Group

Version 4.0

2 July 2010

Contributors: Miha Ahronovitz, Dustin Amrhein, Patrick Anderson, Andrew de Andrade, Joe Armstrong, Ezhil Arasan B, James Bartlett, Richard Bruklis, Ken Cameron, Mark Carlson, Reuven Cohen, Tim M. Crawford, Vikas Deolaliker, Pete Downing, Andrew Easton, Rodrigo Flores, Gaston Fourcade, Thomas Freund, Tom Hanan, Valery Herrington, Babak Hosseinzadeh, Steve Hughes, William Jay Huie, Nguyen Quang Hung, Pam Isom, Shobha Rani J, Sam Johnston, Ravi Kulkarni, Anil Kunjunny, Edmond Lau, Thomas Lukasik, Bob Marcus, Gary Mazzaferro, Craig McClanahan, Meredith Medley, Walt Melo, Andres Monroy-Hernandez, Ayman Nassar, Dirk Nicol, Lisa Noon, Santosh Padhy, Gilad Parann-Nissany, Greg Pfister, Thomas Plunkett, Ling Qian, Balu Ramachandran, Jason Reed, German Retana, Bhaskar Prasad Rimal, Dave Russell, Matt F. Rutkowski, Clark Sanford, Krishna Sankar, Alfonso Olias Sanz, Mark B. Sigler, Wil Sinclair, Erik Sliman, Patrick Stingley, Phillip Straton, Robert Syputa, Robert J. Taylor, Doug Tidwell, Kris Walker, Kurt Williams, John M Willis, Yutaka Sasaki, Michael Vesace, Eric Windisch, Pavan Yara and Fred Zappert.

Public comments on this document are welcomed and encouraged via the discussion groups referenced at http://cloudusecases.org.

Table of Contents

About Version 4: New to Version 4 is Section 8, Service Level Agreements (SLAs). See the Summary of Changes on page 67 for complete details.

1 Introduction

The Cloud Computing Use Case group brought together cloud consumers and cloud vendors to define common use case scenarios for cloud computing. The use case scenarios demonstrate the performance and economic benefits of cloud computing and are based on the needs of the widest possible range of consumers.

The goal of this white paper is to highlight the capabilities and requirements that need to be standardized in a cloud environment to ensure interoperability, ease of integration and portability. It must be possible to implement all of the use cases described in this paper without using closed, proprietary technologies. Cloud computing must evolve as an open environment, minimizing vendor lock-in and increasing customer choice.

The use cases:

- Provide a practical, customer-experience-based context for discussions on interoperability and standards.
- Make it clear where existing standards should be used.
- Focus the industry's attention on the importance of Open Cloud Computing.
- Make it clear where there is standards work to be done. If a particular use case can't be built today, or if it can only be built with proprietary APIs and products, the industry needs to define standards to make that use case possible.

A use case that clearly describes a common task and outlines the difficulties in accomplishing it is the best possible justification for any standards effort.

The Open Cloud Manifesto (opencloudmanifesto.org) is a statement of the principles for maintaining openness in cloud computing. Within two months of its announcement, 250 organizations signed on as supporters. This group's activity is done in light of the six principles of the Open Cloud Manifesto:

- Cloud providers must work together to ensure that the challenges to cloud adoption are addressed through open collaboration and the appropriate use of standards.
- Cloud providers must use and adopt existing standards wherever appropriate. The IT industry has invested heavily in existing standards and standards organizations; there is no need to duplicate or reinvent them.

- When new standards (or adjustments to existing standards) are needed, we must be judicious and pragmatic to avoid creating too many standards. We must ensure that standards promote innovation and do not inhibit it.

- Any community effort around the open cloud should be driven by customer needs, not merely the technical needs of cloud providers, and should be tested or verified against real customer requirements.

- Cloud computing standards organizations, advocacy groups, and communities should work together and stay coordinated, making sure that efforts do not conflict or overlap.

- Cloud providers must not use their market position to lock customers into their particular platforms and limiting their choice of providers.

This paper is part of the ongoing effort to make these principles a reality.

2 Definitions and Taxonomy

The following definitions and taxonomy are included to provide an overview of cloud computing concepts. However, the focus of this white paper is defining cloud scenarios and use cases based on real-world applications and requirements, not defining cloud computing itself. Our goal is to provide use case scenarios that are clear, interesting and useful, regardless of how those scenarios might be defined or placed into a taxonomy.

2.1 Definitions of Cloud Computing Concepts

Cloud Computing: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. (This definition is from the latest draft of the NIST Working Definition of Cloud Computing published by the U.S. Government's National Institute of Standards and Technology.[1])

2.1.1 Delivery Models

The NIST definition of cloud computing defines three delivery models:

- **Software as a Service (SaaS):** The consumer uses an application, but does not control the operating system, hardware or network infrastructure on which it's running.
- **Platform as a Service (PaaS):** The consumer uses a hosting environment for their applications. The consumer controls the applications that run in the environment (and possibly has some control over the hosting environment), but does not control the operating system, hardware or network infrastructure on which they are running. The platform is typically an application framework.
- **Infrastructure as a Service (IaaS):** The consumer uses "fundamental computing resources" such as processing power, storage, networking components or middleware. The consumer can control the operating system, storage, deployed applications and possibly networking components such as firewalls and load balancers, but not the cloud infrastructure beneath them.

[1] You can find the full document on the NIST Cloud Computing page at http://csrc.nist.gov/groups/SNS/cloud-computing/. The document states, "This material is public domain although attribution to NIST is requested. It may be freely duplicated and translated." The essential characteristics, delivery models and deployment models discussed in this paper are based on Version 15 of the document, dated 8-19-09.

2.1.2 Deployment Models

The NIST definition defines four deployment models:

- **Public Cloud:** In simple terms, public cloud services are characterized as being available to clients from a third party service provider via the Internet. The term “public” does not always mean free, even though it can be free or fairly inexpensive to use. A public cloud does not mean that a user’s data is publically visible; public cloud vendors typically provide an access control mechanism for their users. Public clouds provide an elastic, cost effective means to deploy solutions.

- **Private Cloud:** A private cloud offers many of the benefits of a public cloud computing environment, such as being elastic and service based. The difference between a private cloud and a public cloud is that in a private cloud-based service, data and processes are managed within the organization without the restrictions of network bandwidth, security exposures and legal requirements that using public cloud services might entail. In addition, private cloud services offer the provider and the user greater control of the cloud infrastructure, improving security and resiliency because user access and the networks used are restricted and designated.[2]

- **Community Cloud:** A community cloud is controlled and used by a group of organizations that have shared interests, such as specific security requirements or a common mission. The members of the community share access to the data and applications in the cloud.

- **Hybrid Cloud:** A hybrid cloud is a combination of a public and private cloud that interoperates. In this model users typically outsource non-business-critical information and processing to the public cloud, while keeping business-critical services and data in their control.[3]

2.1.3 Essential Characteristics

The NIST definition describes five essential characteristics of cloud computing.

- **Rapid Elasticity:** Elasticity is defined as the ability to scale resources both up and down as needed. To the consumer, the cloud appears to be infinite, and the consumer can purchase as much or as little computing power as they need. This is one of the essential characteristics of cloud computing in the NIST definition.

[2] A private cloud can be managed by a third party and can be physically located off premises. It is not necessarily managed and hosted by the organization that uses it.

[3] A Hybrid Cloud is a superset of the technology used in a Community Cloud. For that reason, the requirements for the two deployment models are discussed together under the heading "Hybrid Cloud" in Section 3.7.

- **Measured Service:** In a measured service, aspects of the cloud service are controlled and monitored by the cloud provider. This is crucial for billing, access control, resource optimization, capacity planning and other tasks.

- **On-Demand Self-Service:** The on-demand and self-service aspects of cloud computing mean that a consumer can use cloud services as needed without any human interaction with the cloud provider.

- **Ubiquitous Network Access:** Ubiquitous network access means that the cloud provider's capabilities are available over the network and can be accessed through standard mechanisms by both thick and thin clients.[4]

- **Resource Pooling:** Resource pooling allows a cloud provider to serve its consumers via a multi-tenant model. Physical and virtual resources are assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or datacenter).[5]

2.1.4 Other Terms

Interoperability: Interoperability is concerned with the ability of systems to communicate. It requires that the communicated information is understood by the receiving system. In the world of cloud computing, this means the ability to write code that works with more than one cloud provider simultaneously, regardless of the differences between the providers.[6]

Portability: Portability is the ability to run components or systems written for one environment in another environment. In the world of cloud computing, this includes software and hardware environments (both physical and virtual).

Integration: Integration is the process of combining components or systems into an overall system. Integration among cloud-based components and systems can be complicated by issues such as multi-tenancy, federation and government regulations.

Service Level Agreement (SLA): An SLA is contract between a provider and a consumer that specifies consumer requirements and the provider's commitment

[4] This does not necessarily mean Internet access. By definition, a private cloud is accessible only behind a firewall. Regardless of the type of network, access to the cloud is typically not limited to a particular type of client.

[5] In many cases privacy laws and other regulations require the cloud provider's resources to be in a particular location. The cloud provider and cloud consumer must work together to adhere to those regulations.

[6] The definitions of interoperability, portability and integration are based on the work at http://www.testingstandards.co.uk/interop_et_al.htm.

to them. Typically an SLA includes items such as uptime, privacy, security and backup procedures.

Federation: Federation is the act of combining data or identities across multiple systems. Federation can be done by a cloud provider or by a cloud broker.

Broker: A broker has no cloud resources of its own, but matches consumers and providers based on the SLA required by the consumer. The consumer has no knowledge that the broker does not control the resources.

Multi-Tenancy: Multi-tenancy is the property of multiple systems, applications or data from different enterprises hosted on the same physical hardware. Multi-tenancy is common to most cloud-based systems.

Cloud bursting: Cloud bursting is a technique used by hybrid clouds to provide additional resources to private clouds on an as-needed basis. If the private cloud has the processing power to handle its workloads, the hybrid cloud is not used. When workloads exceed the private cloud's capacity, the hybrid cloud automatically allocates additional resources to the private cloud.

Policy: A policy is a general term for an operating procedure. For example, a security policy might specify that all requests to a particular cloud service must be encrypted.

Governance: Governance refers to the controls and processes that make sure policies are enforced.

Virtual Machine (VM): A file (typically called an image) that, when executed, looks to the user like an actual machine. Infrastructure as a Service is often provided as a VM image that can be started or stopped as needed. Changes made to the VM while it is running can be stored to disk to make them persistent.

Application Programming Interface (API): An application programming interface is a contract that tells a developer how to write code to interact with some kind of system. The API describes the syntax of the operations supported by the system. For each operation, the API specifies the information that should be sent to the system, the information that the system will send back, and any error conditions that might occur.

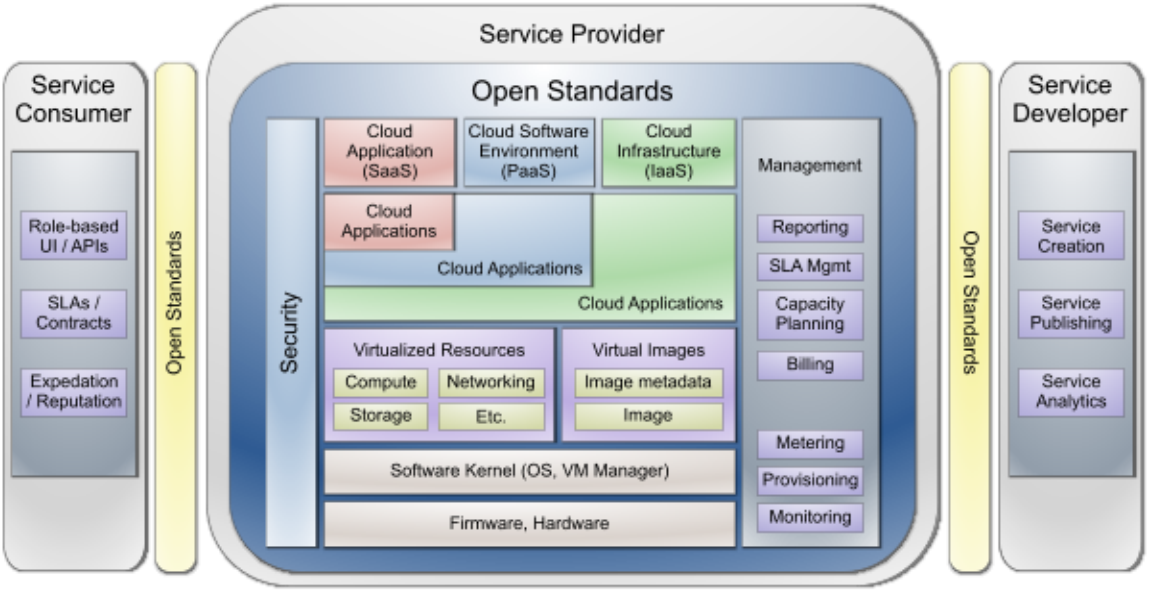
- APIs can be defined in specific programming languages or in more neutral formats such as WSDL or IDL. REST specifications typically don't have a machine-readable language, but they define an API nonetheless.

- An API can also include the details of protocols (such as HTTP) and data formats (such as JSON or an XML Schema).

- An API requires human intelligence to understand the semantics of the data and operations. A machine can discover that method x requires two integers as its parameters, but a developer, a human being, has to figure out which of infinity's two integers should be used.

2.2 Taxonomy

This diagram defines a taxonomy for cloud computing:



In this diagram, Service Consumers use the services provided through the cloud, Service Providers manage the cloud infrastructure and Service Developers create the services themselves. (Notice that open standards are needed for the interactions between these roles.) Each role is discussed in more detail in the following sections.

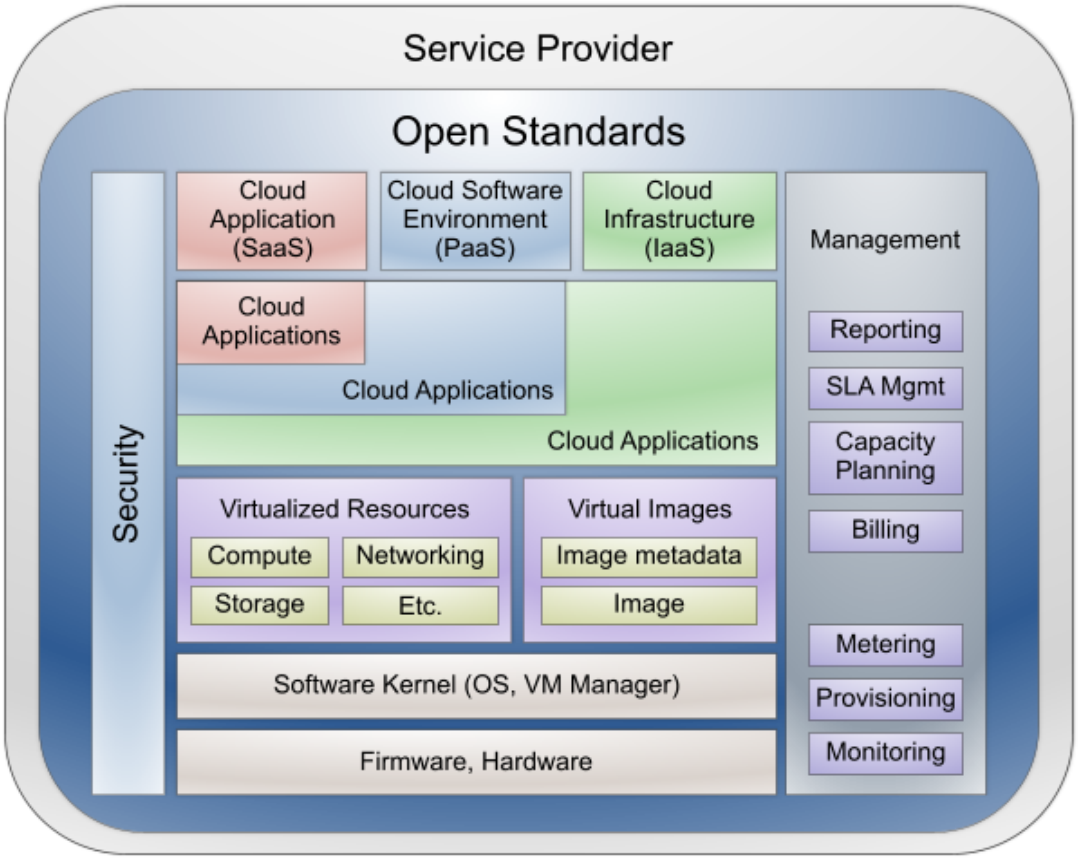
2.2.1 Service Consumer

The service consumer is the end user or enterprise that actually uses the service, whether it is Software, Platform or Infrastructure as a Service.

Depending on the type of service and their role, the consumer works with different user interfaces and programming interfaces. Some user interfaces look like any other application; the consumer does not need to know about cloud computing as they use the application. Other user interfaces provide administrative functions such as starting and stopping virtual machines or managing cloud storage. Consumers writing application code use different programming interfaces depending on the application they are writing.

Consumers work with SLAs and contracts as well. Typically these are negotiated via human intervention between the consumer and the provider. The expectations of the consumer and the reputation of the provider are a key part of those negotiations.

2.2.2 Service Provider



The service provider delivers the service to the consumer. The actual task of the provider varies depending on the type of service:

- For Software as a Service, the provider installs, manages and maintains the software. The provider does not necessarily own the physical infrastructure in which the software is running. Regardless, the consumer does not have access to the infrastructure; they can access only the application.

- For Platform as a Service, the provider manages the cloud infrastructure for the platform, typically a framework for a particular type of application. The consumer's application cannot access the infrastructure underneath the platform.

- For Infrastructure as a Service, the provider maintains the storage, database, message queue or other middleware, or the hosting environment for virtual machines. The consumer uses that service as if it were a disk drive, database, message queue, or machine, but they cannot access the infrastructure that hosts it.

In the service provider diagram, the lowest layer of the stack is the firmware and hardware on which everything else is based. Above that is the software kernel, either the operating system or virtual machine manager that hosts the infrastructure beneath the cloud. The virtualized resources and images include the basic cloud computing services such as processing power, storage and middleware. The virtual images controlled by the VM manager include both the images themselves and the metadata required to manage them.

Crucial to the service provider's operations is the management layer. At a low level, management requires metering to determine who uses the services and to what extent, provisioning to determine how resources are allocated to consumers, and monitoring to track the status of the system and its resources.

At a higher level, management involves billing to recover costs, capacity planning to ensure that consumer demands will be met, SLA management to ensure that the terms of service agreed to by the provider and consumer are adhered to, and reporting for administrators.

Security applies to all aspects of the service provider's operations. (The many levels of security requirements are beyond the scope of this paper.) Open standards apply to the provider's operations as well. A well-rounded set of standards simplify operations within the provider and interoperability with other providers.

2.2.3 Service Developer

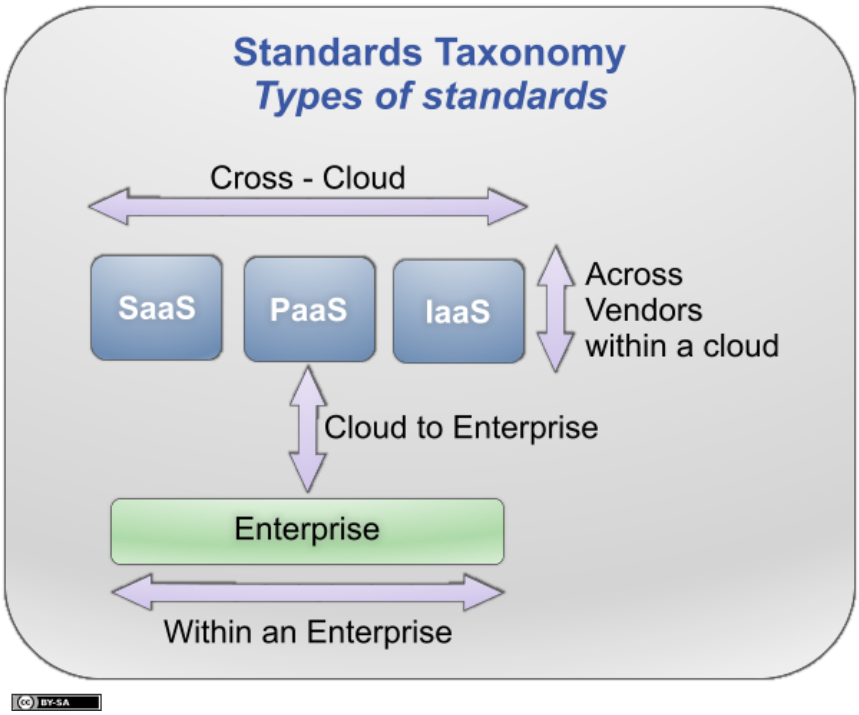
The service developer creates, publishes and monitors the cloud service. These are typically "line-of-business" applications that are delivered directly to end users via the SaaS model. Applications written at the IaaS and PaaS levels will subsequently be used by SaaS developers and cloud providers.

Development environments for service creation vary. If developers are creating a SaaS application, they are most likely writing code for an environment hosted by a cloud provider. In this case, publishing the service is deploying it to the cloud provider's infrastructure.

During service creation, analytics involve remote debugging to test the service before it is published to consumers. Once the service is published, analytics allow developers to monitor the performance of their service and make changes as necessary.

2.3 Relationships Between Standards and Taxonomies

There are four different ways standards will affect cloud use case scenarios. Standards will have an impact within each type of cloud service, across the different types of cloud services, between the enterprise and the cloud, and within the private cloud of an enterprise.



2.3.1 Standards Across Cloud Service Types

As cloud computing becomes more common, applications will likely use different types of cloud services. An application might use a cloud storage service, a cloud message queue, and manage (start/stop/monitor) virtual machines running in the cloud. Standards to define how these different services work together should provide value.

2.3.2 Standards Within Cloud Service Types

Within each type of cloud service (IaaS, PaaS or SaaS), open standards make it possible to avoid vendor lock-in.

For Infrastructure as a Service, a standard set of APIs to work with cloud databases would allow applications to work with data from multiple vendors. That common API would give users the freedom to move to another cloud database provider without major changes, and it would make it much easier to integrate new data sources with existing applications. Common APIs for other cloud infrastructure services such as storage, message queues or MapReduce would provide similar benefits, as would common formats for data and data interchange. In the case of virtual machines, a common virtual machine format is crucial. Users should be able to take a VM built and deployed with one cloud provider and deploy it to another cloud provider without changes.

For Platform as a Service, many of the platforms provided in the cloud are application frameworks. Those frameworks typically provide common services such as user interfaces, storage and databases, but they are accessible only through the APIs of the framework.

For Software as a Service, open standards apply at the application level. Very little of the standards work here is cloud-specific, so those standards are beyond the scope of this paper. For example, a cloud-based word processing application should support standards for document portability; the requirement for standards support in a word processing application has nothing to do with whether the application is running in the cloud.

2.3.3 Standards Between the Cloud and the Enterprise

Even as cloud computing emerges, enterprise architectures such as Java EE are not going away. Standards that define how an enterprise application communicates with resources such as a cloud database or a cloud message queue would enable those applications to use cloud services with little or no changes. Figuring out how to integrate cloud computing with existing architectures and development paradigms will be a major challenge for this group.

2.3.4 Standards Within an Enterprise

Standards within an enterprise will be determined by requirements such as interoperability, auditability, security and management, and will build upon the standards that apply between enterprises and the cloud. The enterprise will interact with some combination of private, public and hybrid clouds.

2.4 Application Programming Interfaces (APIs)

The primary mechanism for building cloud computing solutions is the APIs provided by cloud providers. Cloud APIs work at four different levels, and they fall into five basic categories.

2.4.1 Levels of APIs

There are four different levels of APIs. Each level requires the developer to focus on different tasks and data structures.

Level 1 – The Wire: At this level, the developer writes directly to the wire format of the request. If the service is REST-based, the developer creates the appropriate HTTP headers, creates the payload for the request, and opens an HTTP connection to the service. The REST service returns data with an accompanying HTTP response code. Because of the straightforward nature of many REST services, it is possible to be relatively efficient while writing code at this level.

If the service is SOAP-based, the developer creates the SOAP envelope, adds the appropriate SOAP headers, and fills the body of the SOAP envelope with the data payload. The SOAP service responds with a SOAP envelope that contains the results of the request. Working with SOAP services requires parsing the XML content of the envelopes; for that reason, most SOAP services are invoked with a higher-level API.

Level 2 – Language-Specific Toolkits: Developers at this level use a language-specific toolkit to work with SOAP or REST requests. Although developers are still focused on the format and structure of the data going across the wire, many of the details (handling response codes and calculating signatures, for example) are handled by the toolkit.

Level 3 – Service-Specific Toolkits: The developer uses a higher-level toolkit to work with a particular service. Working at this level, the developer is able to focus on business objects and business processes. A developer can be far more productive when focusing on the data and processes that matter to the organization instead of focusing on the wire protocol.

Level 4 – Service-Neutral Toolkits: This is the highest level of API. A developer working at this level uses a common interface to multiple cloud computing

providers. As with Level 3, the developer focuses on business objects and business processes. Unlike Level 3, a developer working at Level 4 does not have to worry about which cloud service they are working with. An application written with a service-neutral toolkit should be able to use a different cloud vendor (see Changing Cloud Vendors on page 27) with minimal changes to the code, if any.

2.4.2 Categories of APIs

Programming interfaces can be divided into five categories:

Category 1 – Ordinary Programming: The usual application programming interfaces in C#, PHP, Java, etc. There is nothing cloud-specific in this category.

Category 2 – Deployment: Programming interfaces to deploy applications to the cloud. In addition to any cloud-specific packaging technologies, this includes traditional packaging mechanisms such as .Net assemblies and EAR/WAR files.

Category 3 – Cloud Services: Programming interfaces that work with cloud services. As discussed in the previous section, cloud service APIs can be either service-specific or service-neutral. These APIs are divided into subcategories for cloud storage services, cloud databases, cloud message queues, and other cloud services. A developer writing code using cloud services APIs is aware that they are using the cloud.

Category 4 – Image and Infrastructure Management: Programming interfaces to manage virtual machine images and infrastructure details. APIs for images support uploading, deploying starting, stopping, restarting, and deleting images. Infrastructure management APIs control details such as firewalls, node management, network management and load balancing.

Category 5 – Internal Interfaces: Programming interfaces for the internal interfaces between the different parts of a cloud infrastructure. These are the APIs you would use if you wanted to change vendors for the storage layer in your cloud architecture.

2.4.3 Developer Roles

To discuss requirements for developers, it is helpful to point out the different roles that developers play. Requirements for APIs and cloud services vary from one role to the next. Here are the roles we will discuss in this document, along with the categories of APIs they use:

Client Application developer: Writes cloud-based client applications for end users. These developers use APIs for cloud services (category 3).

Application developer: Writes traditional applications that use the cloud. These developers use ordinary APIs (category 1) as well as APIs for cloud services (category 3).

Deployers: Package, deploy and maintain applications that use the cloud. Lifecycle management is a concern here as well. These developers use APIs for deployment, cloud services, and image management (categories 2, 3 and 4).

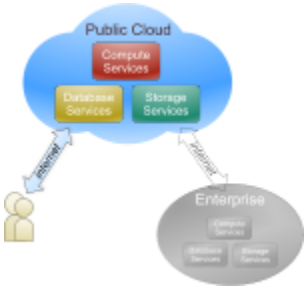
Administrators: Work with applications at multiple levels, including deployment and infrastructure management. These developers use APIs from categories 2, 3, and 4.

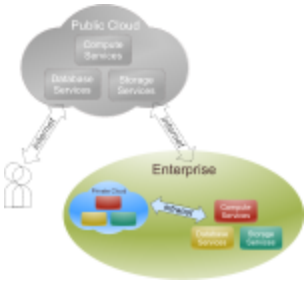
Cloud Providers: Work with the infrastructure beneath their cloud offerings. These developers use APIs from category 5.

3 Use Case Scenarios

The Enterprise Cloud Usage scenarios are intended to illustrate the most typical cloud use cases and are not meant to be an exhaustive list of realizations within a cloud environment.

The graphics in this section have common elements throughout. If a given element does not apply to a particular use case, it is grayed out or drawn with a dashed line. As an example, the Private Cloud use case does not involve the End User or the Public Cloud, so only the Enterprise appears in color.

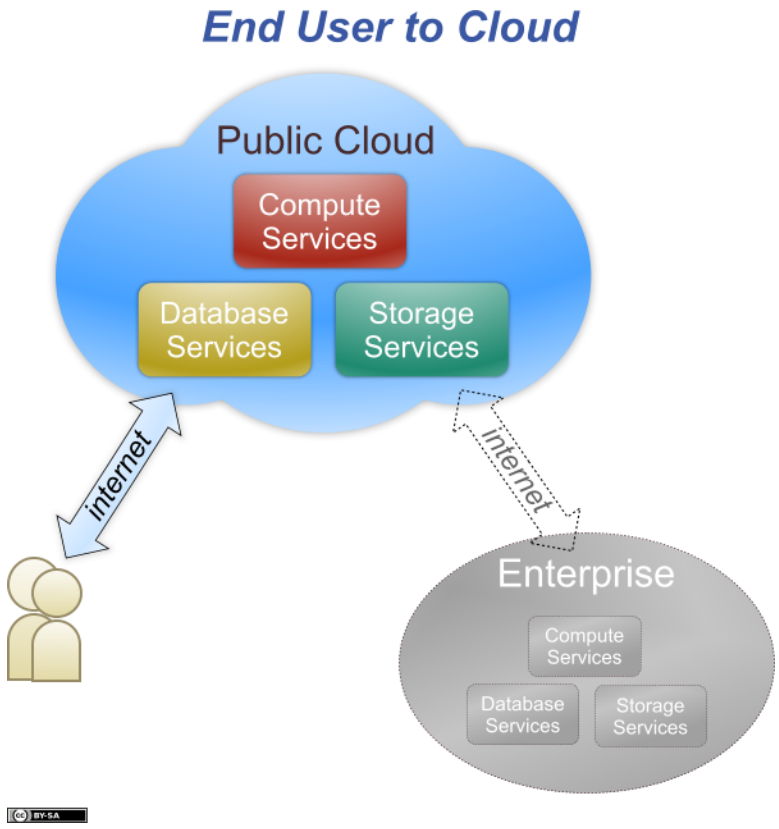
End User to Cloud	Applications running on the cloud and accessed by end users	
Enterprise to Cloud to End User	Applications running in the public cloud and accessed by employees and customers	
Enterprise to Cloud	Cloud applications integrated with internal IT capabilities	

Enterprise to Cloud to Enterprise	Cloud applications running in the public cloud and interoperating with partner applications (supply chain)	
Private Cloud	A cloud hosted by an organization inside that organization's firewall.	
Changing Cloud Vendors	An organization using cloud services decides to switch cloud providers or work with additional providers.	
Hybrid Cloud	Multiple clouds work together, coordinated by a cloud broker that federates data, applications, user identity, security and other details.	

3.1 End User to Cloud

In this scenario, an end user is accessing data or applications in the cloud. Common applications of this type include email hosting and social networking sites. A user of Gmail, Facebook or LinkedIn accesses the application and their data through any browser on any device. The user doesn't want to keep up with anything more than a password; their data is stored and managed in the cloud.

Most importantly, the user has no idea how the underlying architecture works. If they can get to the Internet, they can get to their data.



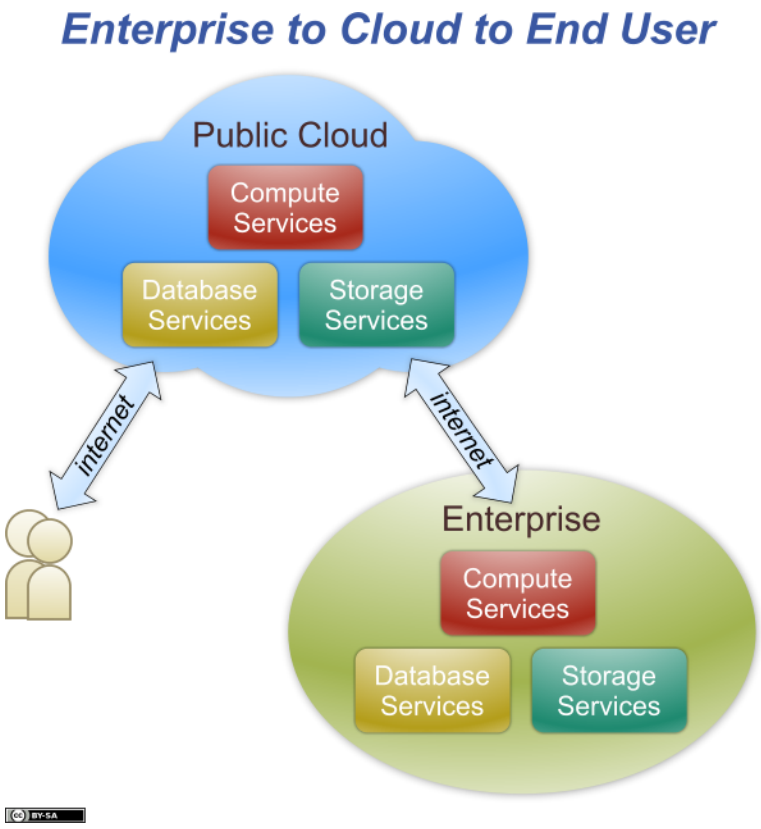
3.1.1 Requirements

- **Identity**: The cloud service must authenticate the end user.
- An **open client**: Access to the cloud service should not require a particular platform or technology.
- **Security**: Security (including privacy) is a common requirement to all use cases, although the details of those requirements will vary widely from one use case to the next. A full discussion of security in cloud computing is beyond the scope of this paper.
- **SLAs**: Although service level agreements for end users will usually be much simpler than those for enterprises, cloud vendors must be clear about what guarantees of service they provide.

3.2 Enterprise to Cloud to End User

In this scenario, an enterprise is using the cloud to deliver data and services to the end user. When the end user interacts with the enterprise, the enterprise accesses the cloud to retrieve data and / or manipulate it, sending the results to

the end user. The end user can be someone within the enterprise or an external customer.



3.2.1 Requirements

- **Identity**: The cloud service must authenticate the end user.
- An **open client**: Access to the cloud service should not require a particular platform or technology.
- **Federated identity**: In addition to the basic identity needed by an end user, an enterprise user is likely to have an identity with the enterprise. The ideal is that the enterprise user manages a single ID, with an infrastructure federating other identities that might be required by cloud services.
- **Location awareness**: Depending on the kind of data the enterprise is managing on the user's behalf, there might be legal restrictions on the location of the physical server where the data is stored. Although this violates the cloud computing ideal that the user should not have to know details of the physical infrastructure, this requirement is essential. Many applications cannot be moved to the cloud until cloud vendors provide an API for determining the location of the physical hardware that delivers the cloud service.

- **Metering and monitoring**: All cloud services must be metered and monitored for cost control, chargebacks and provisioning.

- **Management and Governance**: Public cloud providers make it very easy to open an account and begin using cloud services; that ease of use creates the risk that individuals in an enterprise will use cloud services on their own initiative. Management of VMs and of cloud services such as storage, databases and message queues is needed to track what services are used.

  Governance is crucial to ensure that policies and government regulations are followed wherever cloud computing is used. Other governance requirements will be industry- and geography-specific.

- **Security**: Any use case involving an enterprise will have more sophisticated security requirements than one involving a single end user. Similarly, the more advanced enterprise use cases to follow will have equally more advanced security requirements.

- A **Common File Format for VMs**: A VM created for one cloud vendor's platform should be portable to another vendor's platform. Any solution to this requirement must account for differences in the ways cloud vendors attach storage to virtual machines.

- **Common APIs for Cloud Storage and Middleware**: The enterprise use cases require common APIs for access to cloud storage services, cloud databases, and other cloud middleware services such as message queues.

  Writing custom code that works only for a particular vendor's cloud service locks the enterprise into that vendor's system and eliminates some of the financial benefits and flexibility that cloud computing provides.

- **Data and Application Federation**: Enterprise applications need to combine data from multiple cloud-based sources, and they need to coordinate the activities of applications running in different clouds.

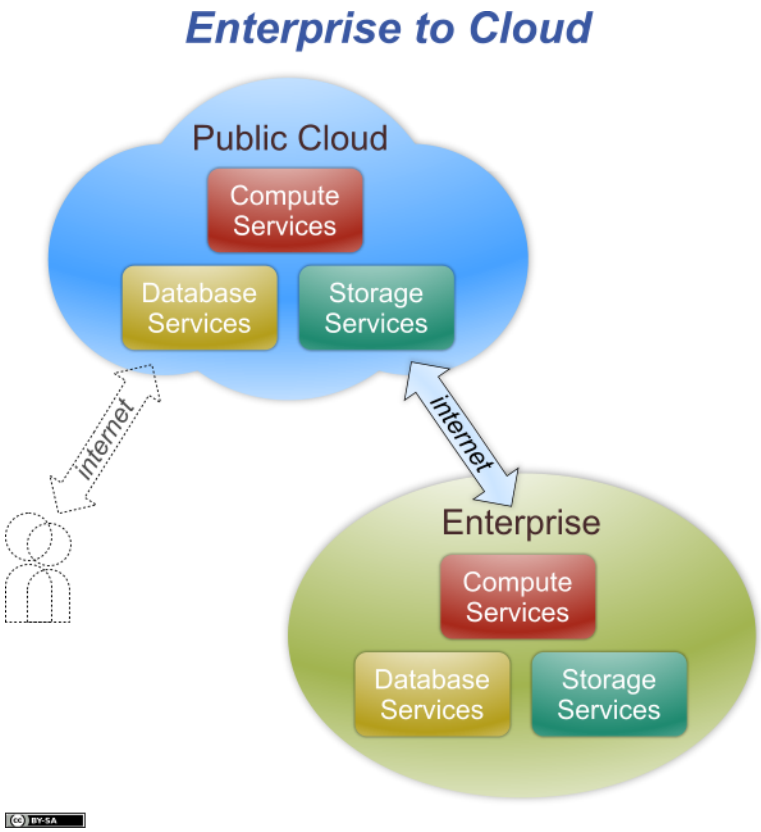
- **SLAs and Benchmarks**: In addition to the basic SLAs required by end users, enterprises who sign contracts based on SLAs will need a standard way of benchmarking performance. There must be an unambiguous way of defining what a cloud provider will deliver, and there must be an unambiguous way of measuring what was actually delivered. (SLAs have additional implications for cloud security; see the discussions of SLAs, auditing and monitoring in Section 6 for more information.)

- **Lifecycle Management**: Enterprises must be able to manage the lifecycle of applications and documents. This requirement includes versioning of

applications and the retention and destruction of data. Discovery is a major issue for many organizations. There are substantial legal liabilities if certain data is no longer available. In addition to data retention, in some cases an enterprise will want to make sure data is destroyed at some point.

3.3 Enterprise to Cloud

This use case involves an enterprise using cloud services for its internal processes. This might be the most common use case in the early stages of cloud computing because it gives the enterprise the most control.



In this scenario, the enterprise uses cloud services to supplement the resources it needs:

- Using cloud storage for backups or storage of seldom-used data
- Using virtual machines in the cloud to bring additional processors online to handle peak loads (and, of course, shutting down those VMs when they're not needed anymore)
- Using applications in the cloud (SaaS) for certain enterprise functions (email, calendaring, CRM, etc.).

- Using cloud databases as part of an application's processing. This could be extremely useful for sharing that database with partners, government agencies, etc.

3.3.1 Requirements

The basic requirements of the Enterprise to Cloud use case are much the same as those for the Enterprise to Cloud to End User use case. An **open client**, **federated identity**, **location awareness**, **metering and monitoring**, **management and governance**, **security**, a **common file format for VMs**, **common APIs for cloud storage and middleware**, **data and application federation**, **SLAs** and **lifecycle management** all apply.

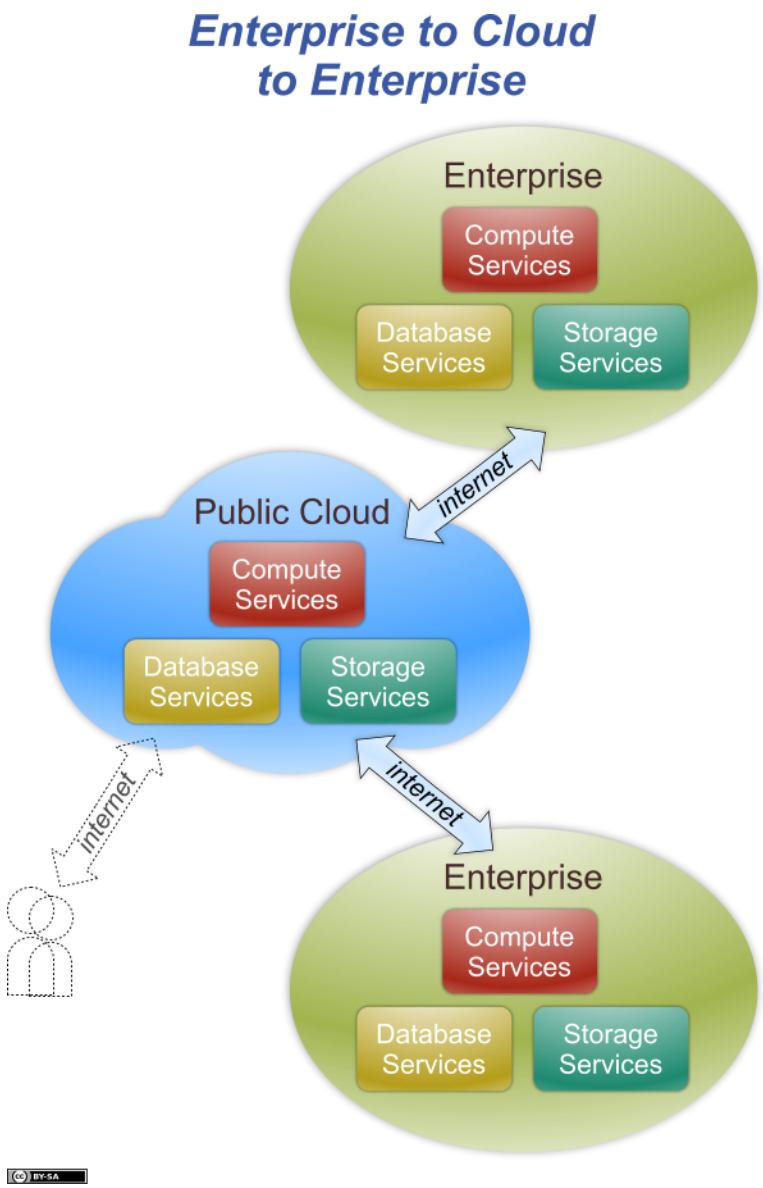
Other requirements for this use case are:

- **Deployment**: It should be simple to build a VM image and deploy it to the cloud as necessary. When that VM image is built, it should be possible to move that image from one cloud provider to another, compensating for the different mechanisms vendors have for attaching storage to VMs. Deployment of applications to the cloud should be straightforward as well.

- **Industry-specific standards and protocols**: Many cloud computing solutions between enterprises will use existing standards such as RosettaNet or OAGIS. The applicable standards will vary from one application to the next and from one industry to the next.

3.4 Enterprise to Cloud to Enterprise

This use case involves two enterprises using the same cloud. The focus here is hosting resources in the cloud so that applications from the enterprises can interoperate. A supply chain is the most obvious example for this use case.



3.4.1 Requirements

The basic requirements of the Enterprise to Cloud to Enterprise use case are much the same as those for the Enterprise to Cloud use case. **Identity**, an **open client**, **federated identity**, **location awareness**, **metering and monitoring**, **management and governance**, **security**, **industry-specific standards**, **common APIs for storage and middleware**, **data and application federation**, **SLAs** and **lifecycle management** all apply.

Other requirements for this use case are:

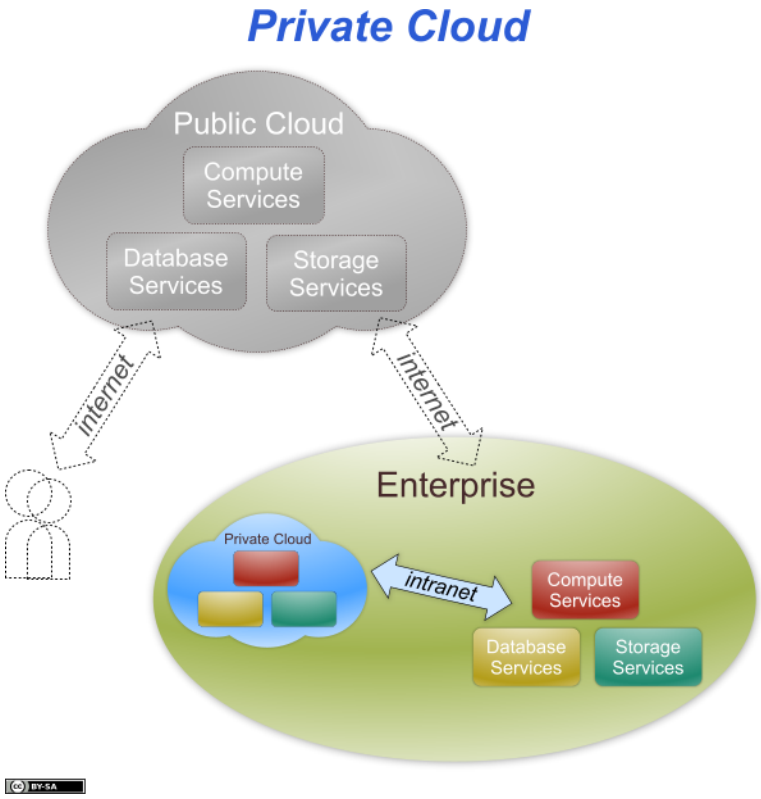
- **Transactions and concurrency**: For applications and data shared by different enterprises, transactions and concurrency are vital. If two enterprises are using the same cloud-hosted application, VM, middleware

or storage, it's important that any changes made by either enterprise are done reliably.

- **Interoperability**: Because more than one enterprise is involved, interoperability between the enterprises is essential.

3.5 Private Cloud

The Private Cloud use case is different from the others in that the cloud is contained *within* the enterprise. This is useful for larger enterprises. For example, if the payroll department has a surge in workload on the 15th and 30th of each month, they need enough computing power to handle the maximum workload, even though their everyday workload for the rest of the month is much lower. With a private cloud, computing power is spread across the enterprise. The payroll department gets extra cycles when they need it and other departments get extra cycles when they need it. This can deliver significant savings across the enterprise.



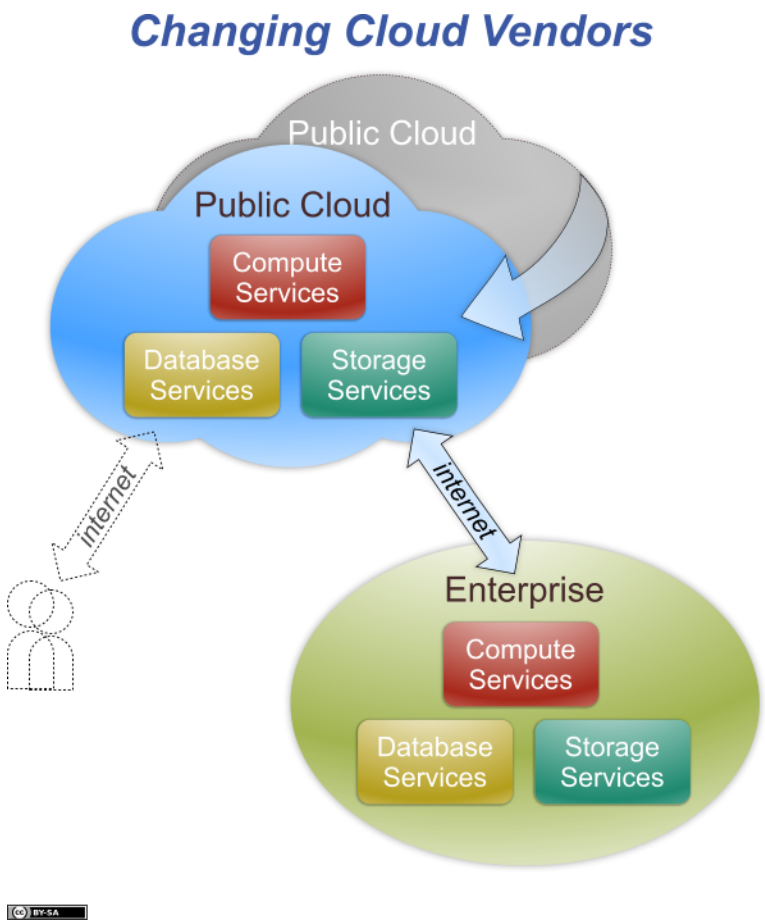
3.5.1 Requirements

The basic requirements of the Private Cloud use case are an **open client**, **metering and monitoring**, **management and governance**, **security**, **deployment**, **interoperability**, a **common VM format**, and **SLAs**.

Note that a private cloud does not require identity, federated identity, location awareness, transactions, industry standards, common APIs for cloud middleware and lifecycle management. In many cases, consumers have to use a private cloud so that location awareness will no longer be an issue. Keeping the cloud inside the enterprise removes many of the requirements for identity management, standards and common APIs.

3.6 Changing Cloud Vendors

This use case involves working with a different cloud vendor, either adding an additional vendor or replacing an existing one. It applies to all of the other use cases discussed in this paper. Being able to work with other vendors without major changes is one of the main benefits of openness and standardization.



There are four different scenarios here, each of which has slightly different requirements. In general, changing cloud vendors requires an **open client**, **location awareness**, **security**, **SLAs**, a **common file format for VMs** and **common APIs for cloud storage and middleware**. The details of those requirements are discussed in each of the following subsections.

3.6.1 Scenario 1: Changing SaaS vendors

In this scenario a cloud customer changes SaaS vendors. Both SaaS vendors provide the same application (CRM, accounting, word processing, etc.). Documents and data created with one vendor's software should be importable by the second vendor's software. In some cases, the customer might need to use the two vendors interchangeably.

3.6.1.1 Requirements

- **Industry-specific standards**: Moving documents and data from one vendor's application to another requires both applications to support common formats. The formats involved will depend on the type of application.

  In some cases, standard APIs for different application types will also be required.

It is important to note that there is nothing cloud-specific to these requirements. The standards for moving a document from Zoho to Google Docs are the same standards for moving a document from Microsoft Office to OpenOffice.

3.6.2 Scenario 2: Changing middleware vendors

In this scenario a cloud customer changes cloud middleware vendors. Existing data, queries, message queues and applications must be exportable from one vendor and importable by the other.[7]

3.6.2.1 Requirements

- **Industry-specific standards**: Moving documents and data from one vendor's middleware to another requires both applications to support common formats. The formats involved will depend on the type of application.

- **Common APIs for Cloud Middleware**: This includes all of the operations supported by today's cloud services, including cloud databases, cloud message queues and other middleware. APIs for connecting to, creating and dropping databases and tables.

  Cloud database vendors have enforced certain restrictions to make their products more elastic and to limit the possibility of queries against large data sets taking significant resources to process. For example, some cloud databases don't allow joins across tables, and some don't support a true

[7] Because of the popularity of cloud storage, cloud middleware (databases, message queues, Map Reduce) and cloud storage are considered separate scenarios, even though both are classified as PaaS.

database schema. Those restrictions are a major challenge to moving between cloud database vendors, especially for applications built on a true relational model.

Other middleware services such as message queues are more similar, so finding common ground among them should be simpler.

3.6.3 Scenario 3: Changing cloud storage vendors

In this scenario a cloud customer changes cloud storage vendors.

3.6.3.1 Requirements

- A **common API for Cloud Storage**: Code that reads or writes data in one cloud storage system should work with a different system with as few changes as possible; those changes should be confined to configuration code. In a JDBC application, as an example, the format of the URL and the driver name are different for different database vendors, but the code to interact with the database is identical.

3.6.4 Scenario 4: Changing VM hosts

In this scenario a cloud customer wants to take virtual machines built on one cloud vendor's system and run it on another cloud vendor's system.

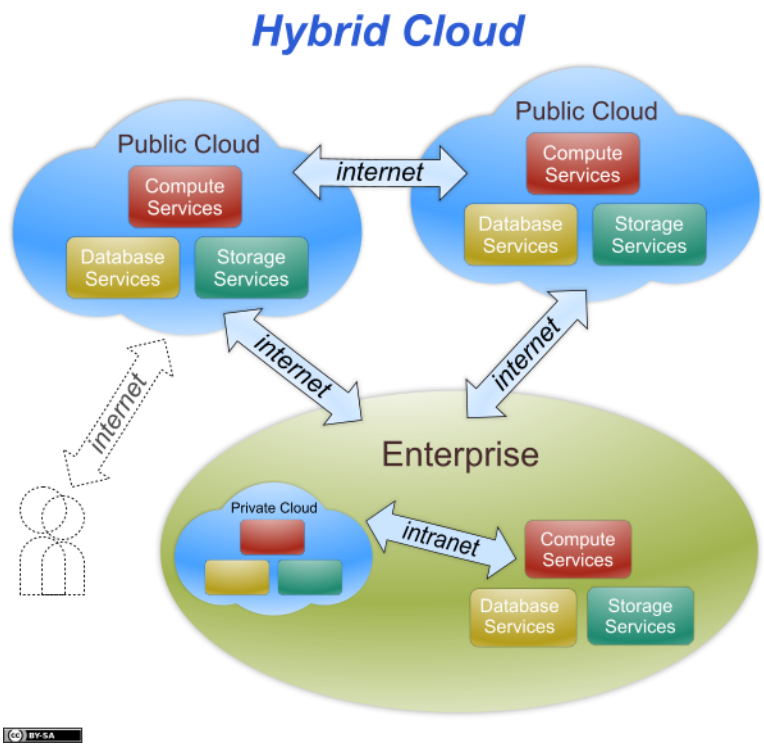
3.6.4.1 Requirements

- **A common format for virtual machines**: The VM format should work with any operating system.

  The assumption here is that the virtual machines themselves are running an operating system such as Windows or Linux. This means that the user of the virtual machine has chosen a platform prior to building a VM for the cloud, so there are no cloud-specific requirements for the software running inside the VM.

3.7 Hybrid Cloud

This use case involves multiple clouds working together, including both public and private clouds. A hybrid cloud can be delivered by a federated cloud provider that combines its own resources with those of other providers. A broker can also deliver a hybrid cloud; the difference is that a broker does not have any cloud resources of its own. The provider of the hybrid cloud must manage cloud resources based on the consumer's terms.



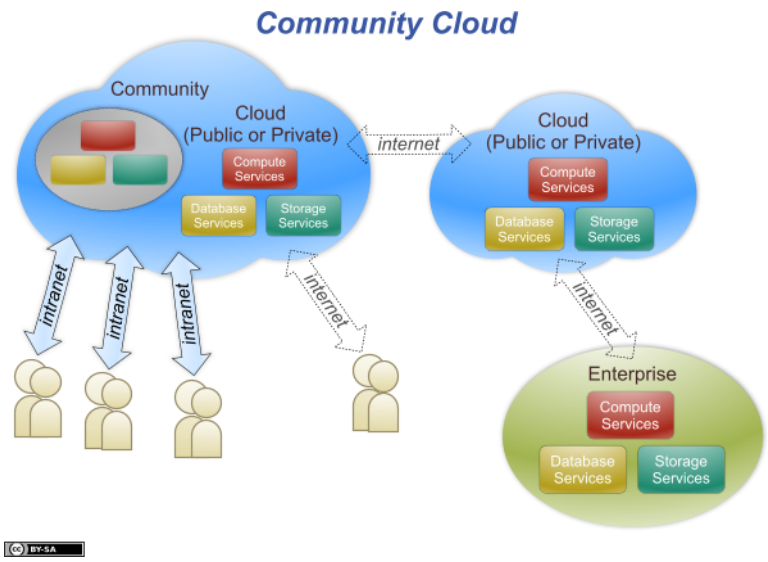
It is important to note that to the consumer of a hybrid cloud, this use case is no different from the End User to Cloud use case discussed earlier. The user has no knowledge of what the hybrid cloud provider actually does.

3.7.1 Requirements

- All of the requirements of the previous use cases (except **Transactions and concurrency**) apply here, particularly **Security**, **Data and Application Federation** and **Interoperability**.

- **SLAs:** A machine readable, standard format for expressing an SLA. This allows the hybrid cloud provider to select resources according to the consumer's terms without human intervention.

As mentioned in Section 2.1.2, the requirements for a community cloud are a subset of the requirements for the hybrid cloud. A community cloud has an infrastructure shared among enterprises with a common purpose. Here is the diagram for a community cloud:



Notice that the communication between the community and the community cloud is done across an intranet. This could be a VPN, but access is not via the public Internet.

3.8 Cross-Reference: Requirements and Use Cases

The following table summarizes the relationships between requirements and use cases:

Requirement	End User to Cloud	Enterprise to Cloud to End User	Enterprise to Cloud	Enterprise to Cloud to Enterprise	Private Cloud	Changing Cloud Vendors	Hybrid Cloud
Identity	✓	✓		✓			✓
Open Client	✓	✓	✓	✓	✓	✓	✓
Federated Identity		✓	✓	✓			✓
Location Awareness		✓	✓	✓		✓	✓
Metering and Monitoring		✓	✓	✓	✓		✓

Requirement	End User to Cloud	Enterprise to Cloud to End User	Enterprise to Cloud	Enterprise to Cloud to Enterprise	Private Cloud	Changing Cloud Vendors	Hybrid Cloud
Management and Governance		✓	✓	✓	✓		✓
Security	✓	✓	✓	✓	✓	✓	✓
Deployment			✓		✓		✓
Transactions and Concurrency				✓			
Interoperability				✓			✓
Industry-Specific Standards			✓	✓			✓
VM Image Format		✓	✓	✓	✓	✓	✓
Cloud Storage API		✓	✓	✓		✓	✓
Cloud Database API		✓	✓	✓		✓	✓
Cloud Middleware API		✓	✓	✓		✓	✓
Data and Application Federation		✓	✓	✓			✓
SLAs	✓	✓	✓	✓	✓	✓	✓
Lifecycle Management		✓	✓	✓			✓

4 Customer Scenarios

This section describes customer experiences with the use cases. Here is a summary of those customer scenarios:

Customer Scenario	Customer Problem Solved	Requirements & Capabilities	Applicable Use Case
Payroll Processing	◆ Processing time reduced ◆ Hardware requirements reduced ◆ Elasticity enabled for future expansion	IaaS (VMs), cloud storage	Enterprise to Cloud
Logistics & Project Management	◆ Processing time reduced ◆ Manual tasks eliminated ◆ Development environment updated and streamlined	PaaS (app framework), cloud storage	Enterprise to Cloud to End User
Central Government	◆ IT expertise consolidated ◆ Hardware requirements reduced	IaaS, PaaS	Private Cloud
Local Government	◆ IT expertise consolidated ◆ Hardware requirements reduced	IaaS, PaaS	Hybrid Cloud
Astronomic Data Processing	◆ Hardware expense greatly reduced (processing power and storage) ◆ Energy costs greatly reduced ◆ Administration simplified	IaaS (VMs), cloud storage	Enterprise to Cloud to End User

4.1 Customer Scenario: Payroll Processing in the Cloud

4.1.1 Applicable Use Case Scenario from Section 3:

Enterprise to Cloud

4.1.2 Customer scenario:

In this scenario, two servers were dedicated to payroll processing, a complex and time-consuming process. The organization decided to see how practical it would be to run the payroll process in the cloud. The existing payroll system was architected as a distributed application, so moving it to the cloud was relatively straightforward.

The payroll application used an SQL database for processing employee data. Instead of rewriting the application to use a cloud database service, a VM with a database server was deployed. The database server retrieved data from a cloud storage system and constructed relational tables from it. Because of the size of the original (in-house) database, extraction tools were used to select only the information necessary for payroll processing. That extracted information was transferred to a cloud storage service and then used by the database server.

The payroll application was deployed to four VMs that run simultaneously; those four VMs work with the VM hosting the database server. The configuration of the payroll application was changed to use the VM hosting the database server; otherwise the application was not changed.

4.1.3 Customer problem solved:

In the cloud-based version of the application, processing time for the payroll task was reduced by 80%. As an added benefit, two servers formerly dedicated to processing payroll were freed up for other tasks. Finally, the cloud-based version is much more elastic; that will be a significant advantage as the organization expands.

4.1.4 Requirements and Capabilities:

The cloud services used were virtual machines and cloud storage (IaaS). The payroll application did not have to be modified, it was simply deployed to the virtual machine. The original application used a relational database. To avoid changing data structures and applications to use a cloud database, a relational database server was deployed into the cloud.

The only API used was the S3 cloud storage API.

4.1.5 Portability Concerns:

The payroll application runs on Fedora and Java 1.5, so it will run without changes on any cloud provider's platform that supports Fedora. Modifying the application to use a different cloud storage provider could be a problem if the other vendor doesn't support the specific S3 APIs used in the payroll process. Finally, changing the application to use a cloud database could be extremely

difficult, particularly if it involved moving to a cloud database that does not support the relational model.

4.2 Customer Scenario: Logistics and Project Management in the Cloud

4.2.1 Applicable Use Case Scenario from Section 3:

Enterprise to Cloud to End User

4.2.2 Customer scenario:

A small construction company with approximately 20 administrative employees needed a way to manage their resources, optimize project scheduling and track job costs. The company had very specific requirements that no commonly available system addressed, so they used a combination of Quickbooks and spreadsheets. This system was not elastic and was a huge waste of human resources.

The solution to the problem was to build a custom client-side application. All of the business logic resides on the client. Data for the application is served from a Google App Engine (GAE) datastore. The datastore does not enforce any sort of schema other than an RDF graph, although it does host an RDF-OWL ontology. The client uses that ontology to validate data before displaying it to the user or sending it back to the GAE.

Data operations are communicated with the datastore using an application-specific RESTful protocol over HTTP. The datastore maintains RDF graphs specific to the applications it is serving within silos managed on the server. Security is implemented separately for each silo depending on the requirements of the application using a particular silo of data. Using this system, any number of applications can use the datastore without building a new code base for each one.

Data was moved into the datastore on GAE from the locally hosted Quickbooks SQL server and the custom spreadsheets using a one-time data migration script that reconciled the data before uploading it to the GAE datastore. The data set was small and easily processed using local resources.

The client application maintains a local datastore that contains a subset of the most recent changes to the data. The REST architecture of the application allows HTTP's built-in caching support to automatically propagate changes to the master datastore down to the client. In addition to the performance benefits of using a subset of the data, this design simplifies security. If a client application does not need to access to certain fields or records, that portion of the datastore never leaves the server.

4.2.3 Customer problem solved:

Data was moved from an inefficient system of software and spreadsheet macros into a cloud-based system. The resulting datastore can be used by a wide range of applications, making future development and maintenance much simpler.

Although the original application infrastructure is still in use, the applications built on that infrastructure no longer rely on spreadsheets to analyze and manipulate the data. Significant savings will come from the fact that maintaining the spreadsheets will no longer be necessary. In addition, cutting and pasting data by hand is no longer part of the process, removing a tedious task and eliminating a source of errors.

4.2.4 Requirements and Capabilities:

The cloud service used the Google App Engine, a PaaS implementation that provides database support. The combination of a RESTful API and the cloud datastore made the application more elastic than an application built around a traditional relational database.

4.2.5 Portability Concerns:

The application runs on the Google App Engine and its BigTable database. BigTable is a sparse, distributed, persistent, multi-dimensional sorted map that achieves elasticity by prioritizing denormalization over normalization. This is a significant difference from most datastores, and requires a fundamental rethinking of application development. Porting the application to run on top of a more traditional datastore would require major changes to the application's architecture.

4.3 Customer Scenario: Central Government Services in the Cloud

4.3.1 Applicable Use Case Scenario from Section 3:

Private Cloud

4.3.2 Customer scenario:

The ministries of the Japanese Government have thousands of servers across their infrastructures. The central government has announced a private "Kasumigaseki" cloud environment to provide a secure, centralized infrastructure for hosting government applications.

Existing back office systems, such as payroll, accounting and personnel management, will be virtualized and hosted in the private cloud. Some front office systems, such as electronic procurement, will be virtualized to a public cloud, but

that is outside the scope of this project. The ultimate goal of the project is to reduce the total cost of ownership by eliminating redundant systems and the need for administrators in each ministry.

4.3.3 Customer problem solved:

The three problems solved by the Kasumigaseki cloud are reduced costs, reduced energy consumption and reduced IT staff.

4.3.4 Requirements and Capabilities:

The cloud infrastructure will be built on a private network built by Japanese telecommunications companies. Because privacy and security are major concerns, a private cloud is required. It is illegal for many types of personal data to be stored on a server outside of Japan.

4.3.5 Portability Concerns:

Because the government is building its own private cloud to host its own applications, portability is not a concern. The government has no intention of moving its centralized applications and data from the private cloud to a public one.

4.4 Customer Scenario: Local Government Services in a Hybrid Cloud

4.4.1 Applicable Use Case Scenario from Section 3:

Hybrid Cloud

4.4.2 Customer scenario:

There are more than 1800 local governments across Japan, each of which has its own servers and IT staff. A secondary goal of the Kasumigaseki cloud is to provide a hybrid cloud environment. In addition to the Kasumigaseki cloud, the Japanese central government has decided to group local governments at the prefecture level. Each prefecture will have a private cloud and a connection to the Kasumigaseki hybrid cloud. Internal tasks and some data will be hosted in the prefecture's private cloud, while other data will be stored locally. Wherever possible, existing systems will be virtualized and hosted in the Kasumigaseki cloud.

4.4.3 Customer problem solved:

The three problems solved by the hybrid cloud are reduced costs, reduced energy consumption and reduced IT staff.

4.4.4 Requirements and Capabilities:

Privacy and security are crucial in this scenario. Japanese law prevents some types of data from being stored outside the local government's servers, so moving applications and data into the Kasumigaseki cloud is not an option. Because some processing will be on the local government's infrastructure, federation of applications and data inside the hybrid cloud is crucial.

4.4.5 Portability Concerns:

As with the previous customer scenario, portability is not a concern because the government is building its own private cloud. The government has no intention of moving its centralized applications and data from the private cloud to a public one.

4.5 Customer Scenario: Astronomic Data Processing

4.5.1 Applicable Use Case Scenario from Section 3:

End User to Cloud

4.5.2 Customer Scenario:

Gaia[8] is a mission of the European Space Agency that will conduct a survey of one billion stars in our galaxy. It will monitor each of its target stars about 70 times over a five-year period, precisely charting their positions, distances, movements, and changes in brightness. It is expected to discover hundreds of thousands of new celestial objects, such as extra-solar planets and failed stars called brown dwarfs.

This mission will collect a large amount of data that must be analyzed. The ESA decided to prototype a cloud-based system to analyze the data. The goals were to determine the technical and financial aspects of using cloud computing to process massive datasets.

The prototype system contains the scientific data and a whiteboard used to publish compute jobs. A framework for distributed computing (developed in house) is used for job execution and data processing. The framework is configured to run AGIS (Astrometric Global Iterative Solution). The process runs a number of iterations over the data until it converges.

For processing, each working node gets a job description from the database, retrieves the data, processes it and sends the results to intermediate servers. The intermediate servers update the data for the following iteration.

[8] See http://www.esa.int/esaSC/120377_index_0_m.html for an overview of the project.

The prototype evaluated 5 years of data for 2 million stars, a small fraction of the total data that must be processed in the actual project. The prototype went through 24 iterations of 100 minutes each, equivalent to running a grid of 20 VMs for 40 hours. For the full billion-star project, 100 million primary stars will be analyzed along with 6 years of data, which will require running the 20 VM cluster for 16,200 hours.

To evaluate the elasticity of a cloud-based solution, the prototype ran a second test with 120 high CPU extra large VMs. With each VM running 12 threads, there were 1440 processes working in parallel.

4.5.3 Customer Problem Solved:

The estimated cost for the cloud-based solution is less than half the cost of an in-house solution. That cost estimate does not include the additional electricity or system administration costs of an in-house solution, so the actual savings will be even greater. Storage of the datasets will be cloud-based as well.

In the second test, a couple of performance problems with SQL queries and lock contention at the database were detected and solved. These problems could not have been detected with the current in-house system. The prototype allowed the organization to find and solve performance and elasticity problems before going to production.

4.5.4 Requirements and Capabilities:

The prototype used VMs for the AGIS software and the database. The database is a traditional database running inside a VM; it is a full relational database, not a cloud database service. For storage, five cloud-based storage volumes (100GB each) were attached to the database server.

4.5.5 Portability Concerns:

All of the VMs are running standard operating systems and none of the software used in the project is cloud-specific. The portability concern for this application would be the ability to migrate those VM images to another provider without having to rebuild or reconfigure the images.

5 Developer Requirements

The requirements for developers cut across all of the use cases and customer scenarios. To simplify the discussion, all developer requirements are listed here.

- **Caching:** A programming toolkit that supports caching for cloud resources would boost the performance of cloud applications. Developers need an API to flush the cache, and might need an API to specifically put an object or resource into the cache.

- **Centralized Logging:** Logging is a common requirement for all developers, regardless of their role or the type of application they are writing. APIs should support writing logs, examining entries, creating logs and opening and closing log files.

- **Database:** Developers need a way to access cloud databases. Cloud databases vary widely in their design (some are schema-driven and relational, many are neither), so developers writing cloud applications will choose a cloud database provider based on the needs of each application. The database API must support the basic CRUD operations.

- **Identity Management:** Developers need a way to manage identities. In the simplest case, this requires a way to authenticate a given user's credentials. In cases where an application works with multiple data sources and applications, a developer needs a way to federate that user's identities. A federated identity management system can produce credentials to allow a given user to access a particular service even if the service and user have no knowledge of each other. APIs for identity management should cache and delete credentials as appropriate.

- **Messaging - Point-to-Point:** Developers need an API to post messages to a queue and to consume those messages. An API must also allow the developer to peek the message (examine the message's contents without consuming it).

- **Messaging - Pub-Sub:** Developers need an API to work with topics in a message queuing system. The API should allow developers to post messages to a topic and retrieve messages from a topic.

- **Raw Compute / Job Processing:** Developers need an API to work with large processing jobs such as Hadoop-style data mining. The API should allow developers to start, stop, monitor and pause processing jobs.

- **Session Management:** The ability to manage user sessions is crucial, particularly in a cloud environment. The infrastructure of a cloud is

redundant and resilient in the face of machine failures, so sessions must be maintained even when a particular cloud node goes down. The session API must make it easy to access or manipulate the current state of the user session.

- **Service Discovery:** Developers need a way to discover which cloud services are available. Cloud services should be searchable by type of service, with the API providing additional function appropriate to each service type.

- **SLAs:** Developers using service discovery need an automated way to determine the policies of the services their code discovers. With such an API, developers can write applications that interrogate cloud services and select the one that best meets the application's SLA criteria.

- **Storage:** Developers need a way to access cloud storage services. The API must provide the ability to store and retrieve both data and metadata.

The following table cross-references the five different categories of APIs discussed in Section 2.4.2 with the requirements of developers.

Developer Requirement	Ordinary Programming	Deployment	Cloud Services	Image and Infrastructure Management	Internal Interfaces
Caching				✓	✓
Centralized Logging	✓	✓	✓	✓	✓
Database			✓		✓
Identity Management		✓	✓	✓	✓
Messaging – Point-to-Point			✓		
Messaging – Pub-Sub			✓		
Raw Compute / Job Processing			✓	✓	✓
Service Discovery		✓	✓	✓	✓

Developer Requirement	Ordinary Programming	Deployment	Cloud Services	Image and Infrastructure Management	Internal Interfaces
Session Management	✓		✓		
SLAs	✓	✓	✓	✓	✓
Storage		✓	✓	✓	✓

6 Security Scenarios

Security, in the cloud or elsewhere, is a crucial topic that could fill any number of pages. The purpose here is to highlight the security issues that architects and developers should consider as they move to the cloud.

An important point to keep in mind is that the cloud does not introduce any new security threats or issues. To put security in perspective, cloud computing as a whole can be considered the ideal use case to highlight the need for a consistent, transparent, standards-based security framework regardless of cloud deployment model. As companies move or build solutions in the cloud, having this consistent security model is vital to simplify development and to avoid vendor lock-in and preserve their IT investments.

The most significant difference when considering security from a cloud perspective is the enterprise's loss of control, as opposed to any particular technical challenge. With an in-house application, controlling access to sensitive data and applications is crucial. With a cloud-based application, access control is just as important, but the infrastructure, platform and application of security is under the direct control of the cloud provider.

This section discusses security topics in the following order:

- **Regulations:** Regulations are not technical issues, but they must be addressed. Laws and regulations will determine security requirements that take priority over functional requirements.
- **Security Controls:** Although a given consumer might need all of these security controls, consumers should be wary of any cloud provider that makes security-related claims and reassurances without an infrastructure capable of delivering all of them.
- **Security Federation Patterns:** To implement these security controls, several federation patterns are needed. Cloud providers should deliver these patterns through existing security standards.

This information sets the tone for the discussion of the user scenarios in the next section.

6.1 Regulations

Beyond all of the technical issues in using cloud computing is the harsh reality of regulations. (These have been mentioned earlier in this paper, but they warrant more discussion within the context of security.)

For a variety of reasons, governments around the world are concerned about the use of cloud computing. Many countries have strict privacy laws that prohibit certain data from being stored on a physical machine located outside that country. There are often stiff penalties for organizations (and, in some cases, their executives) that violate those laws. Any organization storing sensitive data in the cloud must be able to prove that their cloud provider never stores that data on a physical server outside a particular geographic area.

In addition to government agencies, many trade and industry groups create regulations as well. While those regulations might not be required by law, they represent best practices.

Similar concerns apply to applications running in the cloud. If a virtual machine is running in the cloud, can an application running on that VM access sensitive data? This is a gray area that many countries have not addressed, although new laws and regulations will be created on an ongoing basis.

Following these laws and regulations will take precedence over all other requirements. A new law might require an organization to spend their resources changing an application's infrastructure instead of adding features to it. The CIO's office must manage these changes and be alert for new laws and regulations as they emerge.

6.2 Security Controls

A number of controls are necessary to adequately secure any system. The following section describes several use cases and the security requirements for each. The discussion of the use cases relates their requirements to the controls defined here.

Security Control	Description
Asset Management	It must be possible to manage all of the hardware, network and software assets (physical or virtual) that make up the cloud infrastructure. This includes being able to account for any physical- or network-based access of an asset for audit and compliance purposes.
Cryptography: Key and Certificate Management	Any secure system needs an infrastructure for employing and managing cryptographic keys and certificates. This includes employing standards-based cryptographic functions and services to support information security at rest and in motion.
Data / Storage Security	It must be possible to store data in an encrypted format. In addition, some consumers will need their data to be stored separately from other consumers' data.

Security Control	Description
Endpoint Security	Consumers must be able to secure the endpoints to their cloud resources. This includes the ability to restrict endpoints by network protocol and device type.
Event Auditing and Reporting	Consumers must be able to access data about events that happen in the cloud, especially system failures and security breaches. Access to events includes the ability to learn about past events and reporting of new events as they occur. Cloud providers cause significant damage to their reputations when they fail to report events in a timely manner.
Identity, Roles, Access Control and Attributes	It must be possible to define the identity, roles, entitlements and any other attributes of individuals and services in a consistent, machine-readable way in order to effectively implement access control and enforce security policy against cloud-based resources.
Network Security	It must be possible to secure network traffic at the switch, router and packet level. The IP stack itself should be secure as well.
Security Policies	It must be possible to define policies, resolve, and enforce security policies in support of access control, resource allocation and any other decisions in a consistent, machine-readable way. The method for defining policies should be robust enough that SLAs and licenses can be enforced automatically.
Service Automation	There must be an automated way to manage and analyze security control flows and processes in support of security compliance audits. This also includes reporting any events that violate any security policies or customer licensing agreements.
Workload and Service Management	It must be possible to configure, deploy and monitor services in accordance with defined security policies and customer licensing agreements.

Here are some standards that can be applied to these controls:

Security Control	Relevant Standard
Cryptography: Key and Certificate Management	**KMIP**, the OASIS Key Management Interoperability Protocol (http://www.oasis-open.org/committees/kmip/)

Security Control	Relevant Standard
Data / Storage Security	**IEEE P1619**, developed by the IEEE Security in Storage Working Group (https://siswg.net/)
Identity, Roles, Access Control and Attributes	**SAML**, the OASIS Security Assertion Markup Language (http://saml.xml.org/)
Identity, Roles, Access Control and Attributes	**X.509 Certificates**, part of the ITU Public Key and Attribute Certificate Frameworks Recommendations (http://www.itu.int/rec/T-REC-X.509)
Security Policies	**XACML**, the OASIS eXtensible Access Control Markup Language (http://www.oasis-open.org/committees/xacml/)
Workload and Service Management	**SPML**, the OASIS Service Provisioning Markup Language (http://www.oasis-open.org/committees/provision/)

6.3 Security Federation Patterns

Federation is the ability of multiple independent resources to act like a single resource. Cloud computing itself is a federation of resources, so the many assets, identities, configurations and other details of a cloud computing solution must be federated to make cloud computing practical.

The requirements from the previous section are implemented via the following federation patterns:

- **Trust:** The ability for two parties to define a trust relationship with an authentication authority. That authentication authority is capable of exchanging credentials (typically X.509 certificates), and then using those credentials to secure messages and create signed security tokens (typically SAML). Federated trust is the foundation upon which all the other secure federation patterns are based.

- **Identity Management:** The ability to define an identity provider that accepts a user's credentials (a user ID and password, a certificate, etc.) and returns a signed security token that identifies that user. Service providers that trust the identity provider can use that token to grant appropriate access to the user, even though the service provider has no knowledge of the user.

- **Access Management:** The ability to write policies (typically in XACML) that examine security tokens to manage access to cloud resources. Access to resources can be controlled by more than one factor. For example, access

to a resource could be limited to users in a particular role, but only across certain protocols and only at certain times of the day.

- **Single Sign-On / Sign-Off:** The ability to federate logins based on credentials from a trusted authority. Given an authenticated user with a particular role, federated single sign-on allows a user to login to one application and access other applications that trust the same authority. Federated single sign-off is part of this pattern as well; in many situations it will be vital that a user logging out of one application is logged out of all the others. The Single Sign-On pattern is enabled by the Identity Management pattern.

- **Audit and Compliance:** The ability to collect audit and compliance data spread across multiple domains, including hybrid clouds. Federated audits are necessary to ensure and document compliance with SLAs and regulatory requirements.

- **Configuration Management:** The ability to federate configuration data for services, applications and virtual machines. This data can include access policies and licensing information across multiple domains.

Because existing best practices for security apply to the cloud, providers should use existing standards to deliver these federation patterns.

7 Security Use Case Scenarios

This section describes customer experiences using cloud computing while meeting security requirements. With requirements and patterns defined, we look at real-world scenarios to illustrate cloud security in action. The scenarios cover a range of application types, deployment models, patterns and roles.

7.1 Computing Power in the Cloud

7.1.1 Customer Scenario:

An insurance company has a claims application used to capture data about their policy holders and any property damage they suffer. A hurricane is projected to strike the Gulf Coast region of the United States, likely causing massive property damage. This will create a huge spike in claims, which will in turn create an enormous load on the corporate IT infrastructure.

The company's decision is to use a public cloud provider to deliver virtual machines to handle the expected demand. The company must control access between the enterprise system and the virtual machines hosted by the cloud provider, limiting access to only authorized agents of the company. In addition, the company must securely transmit any data created by cloud-based instances of the application back inside the corporate firewall. Finally, the cloud provider must ensure that no traces of the application or its data remain whenever a virtual machine is shut down.

7.1.2 Customer Problem Solved:

Using a public cloud allows the company to handle workloads that could be an order of magnitude greater than usual. Buying, maintaining, powering and cooling enough extra systems to handle the highest imaginable work load is not cost-efficient. The company has the in-house capability to handle its day-to-day operations, yet they can allocate additional computing power automatically as needed.

7.1.3 Requirements and Controls:

The requirements and associated controls for this use case are:

Requirement	Controls
Access to videos limited to particular roles	◆ Identity, Roles, Access Control and Attributes ◆ Asset Management ◆ Network Security

Requirement	Controls
All traces of the application or data must be deleted when a VM is shut down	◆ Workload and Service Management

7.1.4 Federation Patterns:

Trust, Access Management, Configuration Management

7.1.5 Roles

Claims adjusters, insurance agents, auditors

7.2 Cloud-based Development and Testing

7.2.1 Customer Scenario:

An online retailer needs to develop a new Web 2.0 storefront application, but does not want to burden its IT staff and existing resources. The company chooses a cloud provider to deliver a cloud-based development environment with hosted developer tooling and a source code repository. Another cloud provider is chosen to provide a testing environment so that the new application can interact with many different types of machines and huge workloads.

Choosing two providers to handle cloud-based development and testing means federation is crucial.

7.2.2 Customer Problem Solved:

From a development perspective, using cloud-hosted development tools eliminates the need to install, configure and manage tools on each developer's machine. Updates to the tools are installed once, at the cloud provider's site, and all developers are automatically moved to the same version of the tools.

Product builds can be much faster, as large builds can use the scalability of the cloud provider's infrastructure. In addition, cloud-based builds can retrieve the latest source code from the cloud-based repository.

From a testing perspective, as the company moves from a traditional interface to a Web 2.0 interface, the extra burden on the server is an unknown. A Web 2.0 interface has far more interactions with the server than a static Web page, therefore testing the new application in the cloud allows the testing team to measure the impact of the new interface.

Stress testing the new application in the cloud is much easier. If the testing team wants to see how the application performs when requests are coming in from 500 different machines per second, the cloud allows them to start 500 virtual

machines and run those tests. With different VM images, it is straightforward to test the application with many different types of machines (operating systems, versions, protocols, etc.). If the testing team wants to run the same test with 1,000 machines or 10,000 machines, it is cost-effective to do so in the cloud; building the infrastructure in-house is not.

7.2.3 Requirements and Controls:

The requirements and associated controls for this use case are:

Requirement	Controls
Developer tools installed and maintained in one central location	◆ Asset Management
All traces of the application or data must be deleted when a VM is shut down	◆ Workload and Service Management
Single sign-on across development and testing clouds	◆ Cryptography ◆ Endpoint Security ◆ Identity, Roles, Access Control and Attributes ◆ Network Security
Controlled access to source code and test plans	◆ Asset Management ◆ Identity, Roles, Access Control and Attributes
Builds and tests must start and shut down VMs automatically	◆ Service Automation
Builds and tests must report statistics on VM usage and performance	◆ Event Auditing and Reporting

7.2.4 Federation Patterns:

Trust, Identity Management, Access Management, Single Sign-On, Audit and Compliance, Configuration Management

7.2.5 Roles:

Developers, testers, administrators, auditors

7.3 Storage in the Cloud

7.3.1 Customer scenario:

A financial investment company is launching new investment products to its agents and affiliates. A number of videos have been created to teach the

company's agents and affiliates about the benefits and features of the new products. The videos are very large and need to be available on-demand, so storing them in the cloud lessens the demands on the corporate infrastructure. However, access to those videos needs to be tightly controlled. For competitive reasons, only certified company agents should be able to view the videos. An even stronger constraint is that regulations require the company to keep product details, including the videos, confidential during the quiet period before the launch of the product.

The company's decision is to use a public cloud storage provider to scale the secure hosting and streaming of the videos. The cloud solution must control the videos with an auditable access control mechanism that enforces the company's security policies.

7.3.2 Customer Problem Solved:

Using a public cloud storage service allows the customer to manage massive data files and bandwidth demands without increasing the physical resources of their data center. However, government regulations and organizational requirements mean security is vital. A public cloud storage provider that cannot guarantee compliance is not an option, no matter what performance, price or scalability they provide.

7.3.3 Requirements and Controls:

The requirements and associated controls for this use case are:

Requirement	Controls
Access to the application limited to particular roles	◆ Identity, Roles, Access Control and Attributes ◆ Asset Management ◆ Network Security ◆ Policies
Data stored in the cloud must be secured	◆ Cryptography ◆ Data / Storage Security
Data stored in the cloud must be transferred back inside the company's firewall	◆ Cryptography ◆ Data / Storage Security ◆ Endpoint Security ◆ Network Security

7.3.4 Federation Patterns:

Trust, Identity Management, Access Management, Audit and Compliance

7.3.5 Roles:

Video producers, company agents, company affiliates, auditors, regulatory authorities

7.4 Cross-Reference: Security Controls and Customer Scenarios

The following table summarizes the relationships between the security controls and the customer scenarios:

Security Control	Computing Power in the Cloud	Cloud-based Development and Testing	Storage in the Cloud
Asset Management	✓	✓	✓
Cryptography: Key and Certificate Management		✓	✓
Data / Storage Security			✓
Endpoint Security		✓	✓
Event Auditing and Reporting		✓	
Identity, Roles, Access Control and Attributes	✓	✓	✓
Network Security	✓		✓
Policies			✓
Service Automation		✓	
Workload and Service Management	✓	✓	

7.5 Cross-Reference: Security Federation Patterns and Customer Scenarios

The following table summarizes the relationships between the security federation patterns and the customer scenarios:

Security Federation Pattern	Computing Power in the Cloud	Cloud-based Development and Testing	Storage in the Cloud
Trust	✓	✓	✓
Identity Management		✓	✓
Access Management	✓	✓	✓
Single Sign-On		✓	
Audit and Compliance		✓	✓
Configuration Management	✓	✓	

8 Service Level Agreements (SLAs)

The relationship between the cloud provider and the cloud consumer must be described with a Service Level Agreement. Because cloud consumers trust cloud providers to deliver some of their infrastructure services, it is vital to define those services, how they are delivered and how they are used.

An SLA is the foundation of the consumer's trust in the provider. A well-written SLA codifies the provider's reputation.

In addition to the prose that defines the relationship between the consumer and provider, an SLA contains Service Level Objectives (SLOs) that define objectively measurable conditions for the service. The consumer must weigh the terms of the SLA and its SLOs against the goals of their business to select a cloud provider.

It is crucial that the consumer of cloud services fully understand all the terms of the provider's SLA, and that the consumer consider the needs of their organization before signing any agreement.

8.1 What is an SLA?

An SLA defines the interaction between a cloud service provider and a cloud service consumer. An SLA contains several things:

- A set of services the provider will deliver
- A complete, specific definition of each service
- The responsibilities of the provider and the consumer
- A set of metrics to determine whether the provider is delivering the service as promised
- An auditing mechanism to monitor the service
- The remedies available to the consumer and provider if the terms of the SLA are not met
- How the SLA will change over time

The marketplace features two types of SLAs: Off-the-shelf agreements and negotiated agreements between a provider and consumer to meet that consumer's specific needs. It is unlikely that any consumer with critical data and applications will be able to use the first type. Therefore the consumer's first step

in approaching an SLA (and the cloud in general) is to determine how critical their data and applications are.

Most public cloud services offer a non-negotiable SLA. With these providers, a consumer whose requirements aren't met has two remedies:

1. Accept a credit towards next month's bill (after paying *this* month's bill in full), or

2. Stop using the service.

Clearly an SLA with these terms is unacceptable for any mission-critical applications or data. On the other hand, an SLA with these terms will be far less expensive than a cloud service provided under a negotiated SLA.

8.2 Service Level Objectives

An SLO defines a characteristic of a service in precise, measurable terms. Here are some sample SLOs:

- The system should never have more than 10 pending requests.
- Throughput for a request should be less than 3 seconds.
- Data streaming for a read request should begin within 2 seconds.
- At least five instances of a VM should be available 99.99999% of the time, with at least one instance available in each of a provider's three data centers.

Obviously different Service Level Objectives will apply to different use cases, applications and types of data. SLOs can also include an urgency rating to indicate the relative importance of different SLOs. A consumer could use an urgency rating to indicate that availability is more important than response time if the cloud provider cannot deliver both SLOs.

Different roles also affect the SLOs that apply. For example, consider an application written by a cloud consumer, hosted by a cloud provider and accessed by an end user. If the application and its data are hosted by the same cloud provider, chances are the application can access that data without leaving the provider's data center. The cloud consumer will expect very fast response times whenever the application accesses its data. On the other hand, the consumer will have lower expectations of response times whenever an end user accesses the application across the Web.

8.3 Service Level Management

It is impossible to know whether the terms of the SLA are being met without monitoring and measuring the performance of the service. Service Level Management is how that performance information is gathered and handled. Measurements of the service are based on the Service Level Objectives in the SLA.

A cloud provider uses Service Level Management to make decisions about its infrastructure. For example, a provider might notice that throughput for a particular service is barely meeting the consumer's requirements. The provider could respond to that situation by reallocating bandwidth or bringing more physical hardware online. However, if giving one consumer more resources would make it impossible to meet the terms of another consumer's SLA, the provider might decide to keep one customer happy at the expense of another. The goal of Service Level Management is to help providers make intelligent decisions based on its business objectives and technical realities.

A cloud consumer uses Service Level Management to make decisions about how it uses cloud services. For example, a consumer might notice that the CPU load on a particular VM is above 90%. In response, the consumer might start another VM. However, if the consumer's SLA says that the first 100 VMs are priced at one rate, with more VMs priced at a higher rate, the consumer might decide not to create another VM and incur higher charges. As with the provider, Service Level Management helps consumers make (and possibly automate) decisions about the way they use cloud services.

8.4 Considerations for SLAs

As consumers are deciding what terms they need in an SLA, there are a number of factors they should consider.

8.4.1 Business Level Objectives

Debating the terms of an SLA is meaningless if the organization has not defined its business level objectives. A consumer must select providers and services based on the goals of the organization. Defining exactly what services will be used is worthless unless the organization has defined why it will use the services in the first place.

This is an aspect of cloud computing in which the hardest problems are organizational politics rather than technical issues. Getting all parts of an organization to agree on those goals will involve some groups accepting budget cuts, some groups losing control of their infrastructure and other difficult choices. Despite these difficulties, the organization cannot make the most of cloud computing (or any technology, for that matter) until the business level objectives have been defined.

Consumers should know *why* they are using cloud computing before they decide *how* to use cloud computing.

8.4.2 Responsibilities of the Provider and Consumer

Although SLAs are commonly thought of as defining the provider's responsibilities, they should be clear about the consumer's responsibilities as well. Consumer responsibilities might include limits on system usage, restrictions on the type of data that can be stored, or having a valid license for any software used on the provider's systems. The balance of responsibilities between providers and consumers will vary according to the type of service. For example, a cloud provider bears most of the responsibilities for Software as a Service. On the other hand, a VM that contains licensed software and works with sensitive data places many more responsibilities on the consumer who builds and manages it.

8.4.3 Business Continuity and Disaster Recovery

Many consumers use the cloud for business continuity. Some consumers store copies of valuable data in multiple clouds for backup. Other consumers use cloudbursting when in-house data centers are unable to handle processing loads. The cloud can be an invaluable resource to keep an organization running when in-house systems are down, but none of that matters if the cloud provider itself does not have adequate continuity and disaster recovery procedures. Consumers should ensure their cloud providers have adequate protection in case of a disaster.

8.4.4 System Redundancy

Many cloud providers deliver their services via massively redundant systems. Those systems are designed so that even if hard drives or network connections or servers fail, consumers will not experience any outages. Consumers moving data and applications that must be constantly available should consider the redundancy of their provider's systems.

8.4.5 Maintenance

Providers handle the maintenance of their infrastructure, freeing consumers from having to do that themselves. However, consumers should understand how and when their providers will do maintenance tasks. Will their services be unavailable during that time? Will their services be available, but with much lower throughput? If there is a chance the maintenance will affect the consumer's applications, will the consumer have a chance to test their applications against the updated service? Note that maintenance can affect any type of cloud offering and that it applies to hardware as well as software.

8.4.6 Location of Data

The physical location of many types of data is restricted. For example, many countries prohibit storing personal information about its citizens on any machine outside its borders. If a cloud provider cannot guarantee that a consumer's data will be stored in certain locations only, the consumer cannot use that provider's services. If a cloud service provider promises to enforce data location regulations, the consumer must be able to audit the provider to prove that regulations are being followed.

8.4.7 Seizure of Data

There have been a few well-publicized instances of law enforcement officials seizing the assets of a hosting company. Even if law enforcement targets the data and applications associated with a particular consumer, the multi-tenant nature of cloud computing makes it likely that other consumers will be affected. Although there are limits to what an SLA can cover, consumers should consider the laws that apply to the provider. Consumers should also consider using a third party to back up their data and applications.

8.4.8 Failure of the Provider

Any cloud provider has the potential to either go out of business or be acquired by another company. Consumers should consider the financial health of their provider and make contingency plans if the provider were to shut its doors. In addition, the provider's policies on access to the consumer's data and applications if the consumer's account is delinquent or in dispute should be made clear.

8.4.9 Jurisdiction

Consumers should understand the laws that apply to any cloud providers they consider. For example, a cloud provider could be based in a country that reserves the right to monitor any data or applications using that cloud provider's services. Given the nature of the consumer's data and applications, this might not be acceptable.

8.4.10 Cloud Brokers and Resellers

If a cloud provider is actually a broker or reseller for another cloud provider, the terms of the SLA should clarify any questions of responsibility or liability if anything goes wrong at the broker, reseller or provider facilities.

8.5 SLA requirements

8.5.1 Security

Security as a general requirement is discussed in detail in Sections 6 and 7 of this paper. The security-related aspects of an SLA should be written with the security controls and federation patterns from Section 6 in mind. A cloud consumer must understand their security requirements and what controls and federation patterns are necessary to meet those requirements. In turn, a cloud provider must understand what they must deliver to the consumer to enable the appropriate controls and federation patterns.

8.5.2 Data Encryption

If a consumer is storing vital data in the cloud, it is important that the data be encrypted while it is in motion and while it is at rest. The details of the encryption algorithms and access control policies should be specified in the SLA.

8.5.3 Privacy

Basic privacy concerns are addressed by requirements such as data encryption, retention and deletion. In addition, an SLA should make it clear how the cloud provider isolates data and applications in a multi-tenant environment.

8.5.4 Data Retention and Deletion

Many organizations have legal requirements that data must be kept for a certain period of time. Some organizations also require that data be deleted after a certain period of time. Cloud providers must be able to prove they are compliant with these policies.

8.5.5 Hardware Erasure and Destruction

A common source of data leaks is the improper disposal of hardware. If a cloud provider's hard drive fails, the platters of that disk should be zeroed out before the drive is disposed or recycled. On a similar note, many cloud providers offer the added protection of zeroing out memory space after a consumer powers off a VM.

8.5.6 Regulatory Compliance

Many types of data and applications are subject to regulations. Some of those are laws (HIPAA for medical records in the United States), while others are industry-specific (PCI DSS for retailers who accept credit cards). If regulations must be enforced, the cloud provider must be able to prove their compliance.

8.5.7 Transparency

Under the SLAs of some cloud providers, the consumer bears the burden of proving that the provider failed to live up to the terms of the SLA. A provider's service might be down for hours, but consumers who are unable to prove that downtime are not eligible for any sort of compensation.

For critical data and applications, providers must be proactive in notifying consumers when the terms of the SLA are breached. This includes infrastructure issues such as outages and performance problems as well as security incidents.

8.5.8 Certification

There are many different certifications that apply to certain types of data and applications. For example, consumer might have the requirement that their cloud provider be ISO 27001 certified. The provider would be responsible for proving their certification and keeping it up-to-date.

8.5.9 Terminology for key performance indicators

The term uptime can be defined in many ways. Often that definition is specific to a provider's architecture. If a provider has a data center on six continents, does uptime refer to a particular data center or *any* data center? If the only available data center is on another continent, that uptime is unlikely to be acceptable. To make matters worse, other cloud providers will use definitions specific to *their* architectures. This makes it difficult to compare cloud services.

A set of industry-defined terms for different key performance indicators would make it much easier to compare SLAs in particular (and cloud services in general).

8.5.10 Monitoring

If a failure to meet the terms of an SLA has financial or legal consequences, the question of who should monitor the performance of the provider (and whether the consumer meets its responsibilities as well) becomes crucial. It is in the provider's interest to define uptime in the broadest possible terms, while consumers could be tempted to blame the provider for any system problems that occur. The best solution to this problem is a neutral third-party organization that monitors the performance of the provider. This eliminates the conflicts of interest that might occur if providers report outages at their sole discretion or if consumers are responsible for proving that an outage occurred.

8.5.11 Auditability

Many consumer requirements include adherence to legal regulations or industry standards. Because the consumer is liable for any breaches that occur, it is vital

that the consumer be able to audit the provider's systems and procedures. An SLA should make it clear how and when those audits take place. Because audits are disruptive and expensive, the provider will most likely place limits and charges on them.

8.5.12 Metrics

Monitoring and auditing require something tangible that can be monitored as it happens and audited after the fact. The metrics of an SLA must be objectively and unambiguously defined. Cloud consumers will have an endless variety of metrics depending on the nature of their applications and data. Although listing all metrics it is impossible, some of the most common are:

- **Throughput** – How quickly the service responds
- **Reliability** – How often the service is available
- **Load balancing** – When elasticity kicks in (new VMs are booted or terminated, for example)
- **Durability** – How likely the data is to be lost
- **Elasticity** – The ability for a given resource to grow infinitely, with limits (the maximum amount of storage or bandwidth, for example) clearly stated
- **Linearity** – How a system performs as the load increases
- **Agility** – How quickly the provider responds as the consumer's resource load scales up and down
- **Automation** – What percentage of requests to the provider are handled without any human interaction
- **Customer service response times** – How quickly the provider responds to a service request. This refers to the human interactions required when something goes wrong with the on-demand, self-service aspects of the cloud.

8.5.13 Machine-Readable SLAs

A machine-readable language for SLAs would enable an automated cloud broker that could select a cloud provider dynamically. One of the basic characteristics of cloud computing is on-demand self-service; an automated cloud broker would extend this characteristic by selecting the cloud provider on demand as well. The broker could select a cloud provider based on business criteria defined by the consumer. For example, the consumer's policy might state that the broker should use the cheapest possible provider for some tasks, but the most secure provider

for others. Although substantial marketplace demand for this requirement will take some time to develop, any work on standardizing SLAs should be done with this in mind.

8.5.14 Human Interaction

Although on-demand self-service is one of the basic characteristics of cloud computing, the fact remains that there will always be problems that can only be resolved with human interaction. These situations must be rare, but many SLAs will include guarantees about the provider's responsiveness to requests for support. Typical guarantees will cover how many requests the consumer can make, how much they will cost and how soon the provider will respond.

8.6 A note about reliability

In discussions of reliability, a common metric bandied about is the number of “nines” a provider delivers. As an example, five nines reliability means the service is available 99.99999% of the time, which translates to total system outages of roughly 5 minutes out of every 12 months. One problem with this metric is that it quickly loses meaning without a clear definition of what an outage is. (It loses even more meaning if the cloud provider gets to decide whether an incident constitutes an outage.)

Beyond the nebulous nature of nines, it is important to consider that many cloud offerings are built on top of other cloud offerings. The ability to combine multiple infrastructures provides a great deal of flexibility and power, but each additional provider makes the system less reliable. If a cloud provider offers a service built on a second cloud provider's storage service and a third cloud provider's database service, and all of those providers deliver five nines reliability, the reliability of the entire system is less than five nines. The service is unavailable when the first cloud provider's systems go down; the service is equally unavailable when the second or third providers' systems have problems. The more cloud providers involved, the more downtime the overall system will have.

Finally, as the number of cloud providers increases, the number of outside factors increases as well. If a VM and a cloud database are in the same data center, communication between the VM and the database doesn't require network access. On the other hand, if the cloud database is delivered by another provider, the available bandwidth between the VM and the database affects the performance and reliability of the overall system. Both cloud providers cloud can be up and running with healthy systems, but if the network connection between them fails, the overall system is down.

To sum up, any consumer who needs to evaluate the reliability of a cloud service should know as much as possible about the cloud providers that deliver that service, whether directly or indirectly.

8.7 Cross-reference: SLA Requirements and Cloud Delivery Models

The following table cross-references the three NIST delivery models from Section 2.1.1 with the SLA requirements listed here.

Requirement	Platform as a Service	Infrastructure as a Service	Software as a Service
Data Encryption	✓	✓	
Privacy	✓	✓	✓
Data Retention and Deletion		✓	✓
Hardware Erasure and Destruction		✓	✓
Regulatory Compliance	✓	✓	✓
Transparency	✓	✓	✓
Certification	✓	✓	✓
Terminology for Key Performance Indicators		✓	✓
Metrics	✓	✓	✓
Auditability	✓	✓	✓
Monitoring	✓	✓	✓
Machine-Readable SLAs		✓	

8.8 Cross-reference: SLA Requirements and Use Case Scenarios

At its best, an SLA protects the interests of both the cloud consumer and cloud provider. Just as a given SLA doesn't meet the needs of all consumers, every requirement discussed here doesn't apply to all customer scenarios. The

following table cross-references the seven use case scenarios from Section 3 with the SLA requirements listed here.

Requirement	End User to Cloud	Enterprise to Cloud to End User	Enterprise to Cloud	Enterprise to Cloud to Enterprise	Private Cloud	Changing Cloud Vendors	Hybrid Cloud
Data Encryption			✓				
Privacy	✓	✓	✓	✓	✓	✓	✓
Data Retention and Deletion			✓	✓			✓
Hardware Erasure and Destruction			✓	✓			✓
Regulatory Compliance	✓	✓	✓	✓	✓	✓	✓
Transparency	✓	✓	✓	✓	✓	✓	✓
Certification	✓	✓	✓	✓	✓		✓
Terminology for Key Performance Indicators			✓	✓	✓	✓	✓
Metrics	✓	✓	✓	✓	✓		✓
Auditability	✓						
Monitoring	✓	✓	✓	✓	✓		✓
Machine-Readable SLAs				✓			

9 Conclusions and Recommendations

Cloud computing builds on and complements many trends in the industry, including virtualization, SOA and Web 2.0. As a result, standards already exist for many of the requirements outlined in this paper. As we go forward, we will work together as a community to specify the existing standards that meet customer needs, leverage standards work already in progress, and identify what is needed to fill in the gaps not addressed by existing standards.

This paper was created by an open Web community of more than 1400 participants. The initial group consisted of supporters from the Open Cloud Manifesto, but it quickly grew to include many other individuals around the world. The community included representatives from large and small companies, government agencies, consultants and vendors.

As the paper was developed, three principles from the manifesto were crucial: 1) users should work together, 2) activities to keep the cloud open should be customer driven and 3) existing standards should be used wherever possible. This paper is validation that those principles work, and they will be central to any follow-on work.

The use cases described here demonstrate the following general requirements for consumers:

- **Common VM Formats, Data Formats and APIs:** Virtual machines, data and applications created for one cloud provider should run on another cloud provider without changes.
- **Cloud Management:** Cloud computing is not feasible without service management, governance, metering, monitoring, federated identity, SLAs and benchmarks, data and application federation, deployment, and lifecycle management. These requirements are defined in Section 3.
- **Security:** Security in cloud computing is vital, although the requirements for security will vary widely depending on the application and data types.
- **Location awareness:** A way of identifying the location of the physical machine hosting the cloud infrastructure is an absolute requirement for many government regulations.

It must be possible for consumers to implement any of the use cases outlined here without resorting to closed, proprietary solutions that lead to vendor lock-in.

The developer requirements discussed here can be grouped into two broad categories:

- **APIs for Services:** The API requirements for databases, messaging (both point-to-point and publish-subscribe), raw computing power and storage all relate directly to cloud services.

- **Support APIs:** The API requirements for caching, logging, identity management, service discovery, session management and SLAs are necessary to use cloud services effectively.

It must be possible for developers to implement any of the use cases outlined here without resorting to closed, proprietary APIs that lead to vendor lock-in.

Security consistently raises the most questions as consumers look to move their data and applications to the cloud. Cloud computing does not introduce any security issues that have not already been raised for general IT security. The concern in moving to the cloud is that implementing and enforcing security policies now involves a third party. This loss of control emphasizes the need for transparency from cloud providers. The discussion of security here covers the security controls, federation patterns and standards that cloud providers must deliver.

As organizations use cloud services, the responsibilities of both the consumer and the provider must be clearly defined in a Service Level Agreement. An SLA defines how the consumer will use the services and how the provider will deliver them. It is crucial that the consumer of cloud services fully understand all the terms of the provider's SLA, and that the consumer consider the needs of their organization before signing any agreement.

Throughout all aspects of cloud computing, where existing standards meet requirements, we must ensure those standards are implemented pervasively and consistently. Where existing standards do not meet requirements, we must define and implement the standards needed to meet them. This community-written paper is meant to be the reference for establishing a truly open cloud computing environment.

Summary of Changes

Version 2, 30 October 2009:

- Added Section 5, Developer Requirements. A discussion of the different levels of APIs, categories of APIs and developer roles were added as Section 2.4. Developers and their requirements were the main focus of the new content in Version 2.
- Replaced the word “scalable” with “elastic” where appropriate.
- Updated the taxonomy discussion in Section 2.1 to reflect changes to the NIST definition of cloud computing.
- Updated the customer scenario in Section 4.4.
- Expanded the discussions of the requirements for consistent deployment and common VM formats to include the different ways cloud vendors attach storage to virtual machines.

Version 3, 2 February 2010:

- With security in mind, made minor changes to the discussion of Service Level Agreements in Section 3.2.1.
- Added Section 6, Security Scenarios, and Section 7, Security Use Case Scenarios.
- Added a brief summary of security to the Conclusions and Recommendations section.

Version 4, 30 June 2010:

- Added Section 8, Service Level Agreements (SLAs).
- Added a brief summary of Service Level Agreements to the Conclusions and Recommendations section.
- Fixed a typo (from “basic the identity” to “the basic identity”) in Section 3.2.1.