Interoperability and Portability for Cloud Computing: A Guide

Version 2.0

December, 2017

Contents

Acknowledgements

The major contributors to this whitepaper and successive version updates are: Claude Baudoin (cébé IT & Knowledge Management), Eliezer Dekel (IBM), Mike Edwards (IBM), John Meegan (IBM), Jem Pagan (JNK Securities), Karolyn Schalk (IBM), John Shortt (Expert Thinking), Gurpreet Singh (Ekartha), Joe Talik (Neoris), William Van Order (Lockheed Martin), Annie Sokol (NIST), and Steven Woodward (Cloud Perspectives).

Revisions

This second revision contains updates which reflect the ISO/IEC 19941 Cloud Computing Interoperability and Portability standard and its facet models of interoperability, data portability, and application portability. The model of an application and the process of porting an application have also been updated to reflect the new thinking contained in ISO/IEC 19941. The technology of containers and their associated technologies are addressed, as is the place of automation in the use of cloud services, both of which have become dominant aspects of customer use of cloud services since the original version of this document.

The relationship of this whitepaper to the many other new CSCC whitepapers on cloud computing has also been addressed.

Executive Overview

Cloud computing is important for many organizations, with use of a wide range of cloud services and the transition of both data and applications to cloud computing environments. The topics of interoperability and portability are significant considerations in relation to the use of cloud services, but there is also confusion and misunderstanding of exactly what this entails. The aim of this Guide is to provide a clear definition of interoperability and of portability and how these relate to various aspects of cloud computing and to cloud services.

Interoperability and Portability for Cloud Computing: A Guide describes interoperability and portability in terms of a set of common cloud computing scenarios. This approach assists in demonstrating that both interoperability and portability have multiple aspects and relate to a number of different components in the architecture of cloud computing, each of which needs to be considered in its own right. The aim is to give both cloud service customers and cloud service providers guidance in the provision and selection of cloud services indicating how interoperability and portability affect the cost, security and risk involved.

Motivation and Considerations

Cloud computing is having an enormous impact on how organizations manage their information technology resources. The abundance of easy to access computing resources enabled by cloud computing provides significant opportunities for organizations, but poses challenges in a number of areas. The current cloud computing landscape consists of a diverse set of products and services that range from infrastructure services (IaaS), to specific software services (SaaS) to development and delivery platforms (PaaS), and many more. The variety of cloud services has led to proprietary architectures and technologies being used by vendors, increasing the risk of vendor lock-in for customers. Incidents such as a cloud service provider shutting down particular cloud services or the discovery of significant security vulnerabilities in applications have highlighted this risk.

Cloud service customers need to mitigate the probability of lock-in, where they run the risk of being tied to a particular cloud service provider due to the difficulty and costs of switching to use equivalent cloud services from other providers. As an example, consider an organization using a PaaS (Platform as a Service). A PaaS platform from a particular vendor could support only limited and proprietary web frameworks, languages, libraries, databases, etc. This can lead organizations to develop application architectures dictated by features offered by the PaaS cloud service provider which can lead to their applications being locked to that vendor, essentially non-portable. There is no direct means of mitigating this risk, but organizations need to consider this issue carefully when selecting cloud services.

As enterprises adopt cloud computing in its various manifestations, the issues of interoperability and portability need to be addressed head on by both providers and customers. There are opportunities and benefits in resolving the concerns of these cross cutting aspects, organizations should review existing data governance, purchase policies, and processes to see if these support a strategy to achieve high levels of interoperability and portability.

The goals of cloud interoperability and portability for this guidance – which are to enable cloud service users to avoid vendor-lock in and allow for customers to make best use of multiple diverse cloud services that can cooperate and interoperate with each other – are critical to future cloud service adoption and the realization of the benefits of computing as a utility.

Other elements relating to interoperability and portability in cloud computing are outside of the scope for this guide. This includes edge/fog/mist computing and other emerging technologies (blockchain, IoT, AI), which are outside of scope of this practical guide.

Interoperability and Portability Overview

The cloud ecosystem is large, with many providers offering a wide variety of cloud services. Understanding the interoperability and portability "of what" is the necessary first step of planning and designing for the use of any cloud service. Clarifying the specific interoperability and portability concerns accelerates identification of the "best fit" options and potential development of solutions. This section provides an overview of the topics of interoperability and portability which is useful in understanding the more detailed descriptions contained in the scenarios and subsequent sections.

The ISO/IEC 19941 standard on Cloud computing Interoperability and Portability [6] provides an excellent description of the topic.

Basic Definition of Interoperability

Interoperability can be defined as a measure of the degree to which diverse systems or components can work together successfully. More formally, IEEE and ISO define interoperability as the ability for two or more systems or applications to exchange information and mutually use the information that has been exchanged. In the context of cloud computing, interoperability should be viewed as the capability of public cloud services, private cloud services, and other diverse systems within the enterprise to understand each other's application and service interfaces, configuration, forms of authentication and authorization, data formats, etc. in order to work with each other.

In cloud computing, the most significant interacting components are those which belong to the cloud service customer which interact with components of the cloud service provider. The nature of the interaction is a network connection using a prescribed interface or API. There are typically multiple separate interfaces, each dealing with a different aspect of the cloud service. For example, there are the functional interfaces of the cloud service itself, authentication and authorization interfaces, interfaces for administration of the cloud services, and business interfaces for billing and invoicing. The ideal of interoperability is that the interfaces are standardized in some way – i.e., they are *interoperable* - so that the customer can switch to another cloud service provider with minimal impact on the customer's components.

It is important to recognize that there are different aspects of interoperability which can be described as separate facets. Refer to Appendix A for a detailed discussion of a 5-facet interoperability model for

cloud computing which describes each facet and the associated interoperability challenges that need to be addressed.

Basic Definition of Portability

In the context of cloud computing, *Portability* is about the ability of a customer to move and suitably adapt their applications and data between their own systems and cloud services, and between cloud services of different cloud service providers and potentially different cloud deployment models. The main problem caused by the lack of portability is that it may take considerable effort to transform the application or data from its form on the source system to the form required by the target system. Portability is differentiated into two separate areas: cloud data portability and cloud application portability:

- *Cloud data portability* is the ability to easily transfer data from one cloud service to another cloud service or between a cloud service customer's system and a cloud service, in a commonly used electronic format. It is the ease of moving the data that is the essence here. This might be achieved by the source service supplying the data in exactly the format that is accepted by the target service. But even if the formats do not match, the transformation between them may be simple and straightforward to achieve with commonly available tools.

  The first aspect of data portability between cloud services is that there must be a capability to retrieve customer data from the source cloud service and also a capability to import customer data into the target cloud service. This is commonly done through the existence of some API (or web interface) associated with the cloud service – it may be a generic API such as one of the forms of FTP, for example, or it may be a specific API unique to the cloud service. It is important to note that the API used for the source service may not be the same as the API used for the target service and that different tooling may be required in each case.

- *Cloud application portability* is the ability to easily transfer an application or application components from one cloud service to a comparable cloud service or from a cloud service customer's system to a cloud service. The key is the ease of moving the application or application components. The application may require recompiling or relinking for the target cloud service, but it should not be necessary to make significant changes to the application code.

As for interoperability, both cloud data portability and cloud application portability each have aspects that can be described in a facet model, which is described in detail in Appendix B.

Interoperability and Portability Challenges

For interoperability, there are many challenges associated with cloud computing. In general, the interfaces and APIs of cloud services are not standardized and different providers use different APIs for what are otherwise comparable cloud services.

The greatest level of interoperability is likely to be found for IaaS cloud services, where functionality is often broadly equivalent and there are a number of standard interfaces - some formally standardized such as CDMI [9], others being de-facto standards in the marketplace. PaaS cloud services have lower levels of interoperability. There are few interface standards for PaaS functionality, although there are some open source platforms, such as Cloud Foundry [10], that are becoming popular in the marketplace and where different cloud service providers use the same open source platform, their interfaces are either identical or closely equivalent.

It is SaaS applications which present the greatest interoperability challenge today. There are very few standard APIs for SaaS applications - even switching from one SaaS application to another SaaS application with comparable functionality typically involves a change in interface. There is a resultant impact on both end users of the cloud service for any user interfaces and also on any applications or systems belonging to the cloud service customer that use APIs offered by the SaaS application.

There are some practical approaches to handling these interoperability challenges. One possibility is for the cloud service customer to provide an isolation or mapping layer between their own applications and systems and the cloud service interfaces, so that the cloud services are not invoked directly by customer applications. Technologies such as enterprise service buses (ESB) can be used to build these isolation layers. An emerging trend is for customers to write custom code in a PaaS platform in order to tailor access to SaaS applications to suit their organizational needs - these PaaS applications form an isolation layer for other customer systems.[1] Another possibility is to use the services offered by an inter-cloud provider (sometimes called a cloud service broker), who takes on the role of mapping a "standard" interface offered to the customer to a varying set of interfaces offered by a number of different cloud service providers.

For application portability, the biggest challenges are for applications built for PaaS platforms. For IaaS cloud services, there are in practice a number of standards that enable portability of applications, such as OVF [11] and the provision of commonly used operating systems like Linux. PaaS platforms can vary widely between different providers - the app environment can differ substantially, including the way in which platform services are offered to the app code and also in which set of platform services are available. For example, in order to be scalable and elastic a PaaS may enforce a specific way to persist and manage data that may not be supported by other PaaS platforms. The differences between alternative PaaS platforms can require extensive re-engineering of customer code when the code is moved between those platforms. Two trends in the marketplace seem to offer promise for easing the situation. One is the increasing adoption of common open source PaaS platforms such as Cloud Foundry [10], while another is the emergence of containerization technologies that allow subdivision and independent deployment of parts of an application, such as Docker and Kubernetes.

[1] Some SaaS providers make available a PaaS platform alongside their SaaS services specifically to make it simpler for customers to write and run custom applications based on the API made available by the SaaS service, such as the Heroku PaaS supplied by Salesforce. However, doing this may also tie the customer more firmly to a single cloud service provider and make portability more challenging.

Elements Involved in Interoperability and Portability for Cloud Services

Before examining specific scenarios, it is worth highlighting the various elements relating to the cloud service customer and to the cloud service.

First it is worth having a model of an application, particularly when considering application portability, as shown in Figure 1.

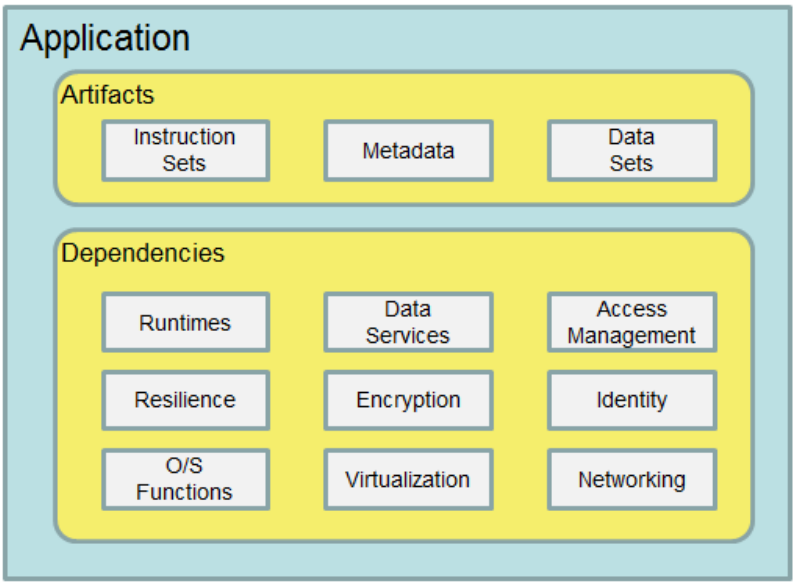

Figure 1: Model of an Application

The application consists of a set of artifacts that include instruction sets, metadata and data sets directly associated with the application. The application has a set of dependencies that must be supplied in order for it to work correctly. The dependencies can include runtimes, data services (such as databases), access management, identity, and encryption. It can also include resilience capabilities (redundant instances with failover, for example), operating system functions, virtualization capabilities and networking. It may also include a wider set of services.

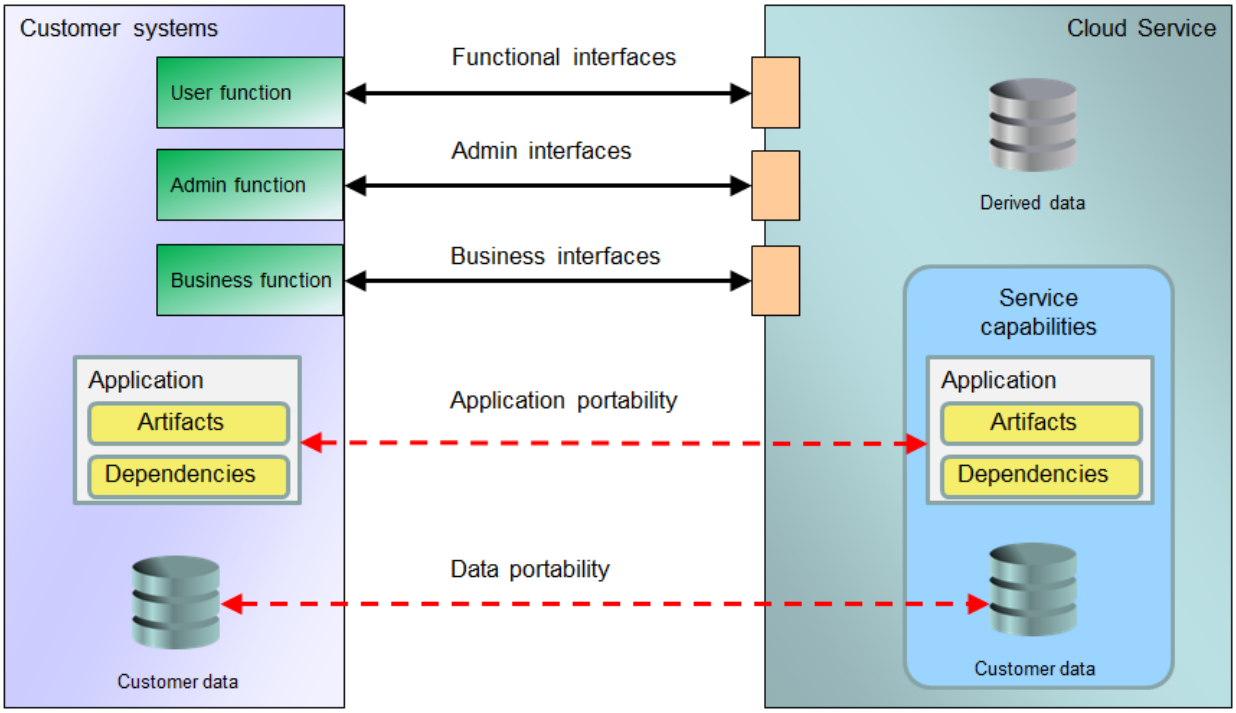

Figure 2: Elements of Interoperability and Portability for Cloud Services

Figure 2 illustrates some of the key elements associated with using a cloud service and the components, interfaces and the data associated with that use.

In this diagram, the ***Application*** within the customer systems and the cloud service represents the customer application in the case of IaaS and PaaS cloud services, but in the case of a SaaS service, the application would typically belong to the provider and would be managed by the provider.

The ***Artifacts and Dependencies*** represent the constituent parts of the application (as shown in Figure 1).

Customer data represents the cloud service customer data, which may be held as records in a database or held as data objects in files or in a data store. ***Derived data*** represents data which is created and stored as a result of the customer use of the service, such as log records or configuration information.

Figure 2 shows three main interfaces between customer systems and the cloud service: the Functional interfaces, the Admin interfaces, and the Business interfaces. The ***Functional interfaces*** are associated with the main functional capabilities offered by the cloud service. The ***Admin interfaces*** involve capabilities for administering the cloud service and include capabilities such as monitoring the service and managing its behavior including aspects of security such as user identities, authentication tokens, and authorizations. The ***Business interfaces*** involve capabilities relating to the business aspects of the cloud service including subscription information, billing and invoicing.

It is important to understand that each of these interfaces may have multiple forms. For example, the main capabilities of the application may be presented in the Functional interfaces to end users as a web browser application or as a mobile app. However, the same capabilities may also be made available as an API for consumption by custom applications written or purchased by the customer and running on the customer systems. In the cloud service environment, APIs are typically defined by a programmatic interface based on a common protocol such as REST/JSON or SOAP.

Interoperability aspects of a cloud service mainly relate to the three interfaces between the customer systems and the cloud service – how users and applications in the customer environment interact with the functional, admin and business interfaces offered by the cloud service. It is important to understand that the interoperability of the three interfaces may be independent of each other and that the interoperability of one interface does not guarantee the interoperability of the others.

Application Portability relates to the capability of moving the App code to or from the cloud service. This typically only applies to IaaS and PaaS services, since in the case of a SaaS service the App code belongs to the cloud service provider and cannot be ported elsewhere by the customer. One of the most important factors for application portability is that the target environment needs to support both the application artifacts and the application dependencies.

Data Portability relates to the capability of moving data into and out of the cloud service environment. Typically, it is the cloud service customer data which is the concern for data portability. However, some of the cloud service derived data may also be of concern in relation to some cloud services and should not be overlooked. For cloud service customer data, data portability is usually of most concern for SaaS cloud services, since for these services, the content, data schemas and storage format are under the control of the cloud service provider and the customer will need to understand how the data can be imported into the service and exported from the service. For IaaS and PaaS services, it is typically the case that the cloud service customer is in control of the content and schemas for the data, with the service offering basic storage capabilities such as a file system or object store.

Containers and Container Infrastructure

For application portability, containers and related container infrastructure are becoming key technologies.

For containers, the Docker containerization platform [12] and the closely associated Open Containers Initiative [13] offer a standardized approach to the deployment of application code and associated software stacks. It is increasingly the case that cloud service providers offer support for software deployed in Docker containers, making the process of porting applications between environments much simpler and more straightforward. This applies principally to IaaS and PaaS cloud services.

It is also the case that the process of deploying and managing the sets of containers that are required for any given application is in the process of being standardized via common acceptance of the open source Kubernetes container orchestration tooling [14].

However, it is important to recognize that containers and their associated toolsets are no panacea and have some problems of their own with regard to both portability and interoperability. Deployment of containers to different cloud service provider environments may have some interoperability issues caused by differences in the interfaces made available. Portability may be a challenge, both in terms of the software levels used by different cloud service providers, but also due to different ways in which the cloud service provider makes available services to satisfy various application dependencies. It may not be possible to port a containerized application without making some changes.

Automation

Technologies that are making portability of both applications and data easier are the automation tools that are now commonly used to handle the customer-side operations relating to the use of cloud services. Automation is a key element of cloud computing itself and this applies equally to the cloud service customer as it does to the cloud service provider. Basically, manual processes are very inefficient and error prone and, as a result, automation solutions have grown up alongside cloud computing to enable efficient and accurate use of cloud services.

A key element of automation tooling is that the tools typically have adapters that enable them to work with a range of cloud services from different cloud service providers. In other words, the automation tooling enables interoperability with the target cloud services and also can deal with the portability issues for data and for applications where the environment within the target cloud service varies. Thus cloud service customers can ease interoperability concerns by utilizing automation tooling that is capable of interfacing with a range of cloud services and cloud service providers.

Interoperability and Portability Scenarios

This section leverages a set of scenarios to describe interoperability and portability considerations and requirements including recommendations on how to address them:

1. Customer switches between providers for a cloud service
2. Customer uses cloud services from multiple providers
3. Customer links one cloud service to another cloud service
4. Customer links in-house capabilities with cloud services
5. Migration of customer capabilities into cloud services

These scenarios are described in the sections that follow. Note that the scenarios focus primarily on public cloud since this deployment model presents the greatest interoperability and portability challenges to customers.

Scenario 1: Customer Switches Providers for a Cloud Service

This scenario addresses the straightforward case of a cloud service customer currently using a cloud service of cloud service provider A, who wishes to switch to using an equivalent cloud service of cloud service provider B. This use case is key from the perspective of the customer since it is essential to

enabling customers to take advantage of the marketplace in cloud services and avoid the issue of lock-in to a single cloud service provider.

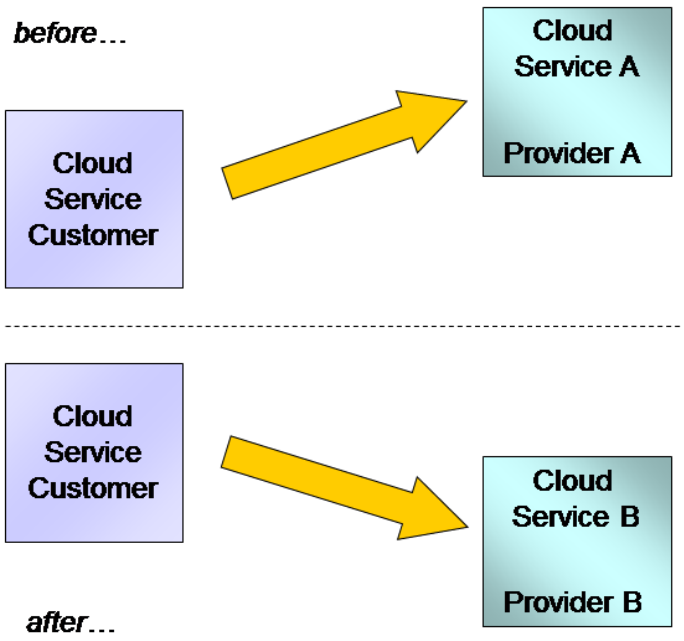

Figure 3: Customer Switches Providers for a Cloud Service

While this scenario is outwardly simple, the reality is that this scenario touches on many of the issues associated with both interoperability and portability. The exact set of issues will vary depending on the nature of the cloud service.

Interoperability Considerations

For SaaS cloud services, the application belongs to the cloud service provider.[2] In this case, moving from provider A to provider B does not involve porting the application as the application may be completely different between the two cloud service providers. What is important in the SaaS case is the compatibility of the functional interface for the application – in particular, any user interfaces presented to end users and also any APIs made available for use by customer applications. It is probably unrealistic to expect that user interfaces will be identical for cloud service A and cloud service B, however, it is reasonable to expect that similar functionality will be presented in a broadly similar way to reduce the cost and effort of retraining end users.

[2] Some SaaS providers make available a PaaS platform alongside their SaaS services specifically to allow customers to write and run code which customizes the use of the SaaS applications, and also to develop and run new applications. In such cases, transitioning to a new SaaS service of a different provider can also include code portability concerns. Refer to the PaaS description in this section for interoperability and portability implications for this type of code.

However, any functional APIs made available by a SaaS service are likely to be used by customer applications. These applications will need to deal with the switch from cloud service A to cloud service B. If the APIs are not interoperable, as is quite likely to be the case, then the implication is that any customer applications using the APIs would need to be changed as part of the process of switching from cloud service A to cloud service B (see the Interoperability and Portability Challenges section).

For IaaS and PaaS services, interoperability is not an issue for any functional interfaces offered by the application when moving from provider A to provider B since the customer owns the application and its functional interfaces. Similarly, the user interfaces presented to end users are likely to be the responsibility of the application and will not be directly affected by the cloud service itself. As long as the application can be ported, then the user interface will also port and be available when using cloud service B. However, the cloud service APIs used to upload, deploy, and control the application in the cloud service are an interoperability concern, since tooling used by the customer operations staff uses these APIs and this tooling needs to connect to cloud service A and then to cloud service B as part of the migration (for example, the automation tooling discussed earlier).

Portability Considerations

For SaaS services, it is typical that the format and the content of the cloud service customer data is in the hands of the cloud service provider while the data itself is an asset of the SaaS customer. Thus ***data portability*** is a major consideration in moving from cloud service A to cloud service B if the cloud service is a SaaS service. Ideally the data syntax should be the same for both service A and service B. In addition to the format, the data content (extent and semantics) should be the same for service A and service B. Data portability can still be achieved if the syntax is different between service A and service B, since there are straightforward standard tools that can be used to perform some data transformations. Where there are not standard tools, it may be possible to build a custom tool. Differences in the extent or in the semantics of the data are much more serious and could be a major barrier to achieving data portability.

For IaaS services, the ***data syntax and data semantics*** for cloud service customer data is usually in the hands of the customer, since the facilities provided by the cloud service are typically relatively low level, such as providing volumes for binary file or object storage (i.e., the cloud service does not know or care about the detailed syntax and semantics of the customer data). As a result, data portability is not likely to be a major concern for IaaS services.

Similar considerations can apply to data portability for PaaS services, but the situation is often more complex. For customer data, the PaaS service may provide instances of databases ready-to-use, in which case the actual database(s) provided may be sensitive to the data syntax of the customer data, although there are generalized formats (CSV, XML, etc.) which are supported by many types of database. How data is loaded into a PaaS cloud service and how it is retrieved needs to be examined by the customer and migration from cloud service A to cloud service B involves data portability questions that need to be answered.

For IaaS and PaaS cloud services, the application belongs to the customer and the question of ***application portability*** is of utmost importance. What does it take to move the application from cloud service A to cloud service B? The first question relates to application syntactic facet: what format each cloud service accepts for the application artifacts– for example, do they accept the same format of VM image or container image? Application instruction facet asks whether the machine architectures for the two services are the same (i.e., can the target deal with the application instruction sets). Finally, there is the question of whether the target environment can support all the application dependencies, such as runtimes, operating systems, and so on.

In the ideal case, application portability implies that the application artifacts that run on cloud service A will run on cloud service B without any changes. There may be cases where application portability cannot be achieved without some changes. In this case, it is the amount of change and the nature of the change that must be considered. Rebuilding the application code, possibly against a different version of the operating system or a different version of the libraries used by the code, may be simple and low cost. Redesigning the application code to adapt to changed interfaces is likely to be more costly and is less desirable.

For PaaS services, the considerations for application artifacts portability can be much more complex. The application dependencies can consist of a substantial stack of software with many APIs which are used by the application artifacts. In addition, many capabilities can be presented to the application as services of various kinds (databases, messaging, rules engines, etc.). The application may have dependencies on a particular set of services, via their APIs, and it is vital to know that the set of services available within cloud service A is matched by the set of services available in cloud service B.

Admin Interface Considerations

Cloud service admin interfaces, which are used to monitor and manage applications, are significant for all forms of cloud services. These admin interfaces may involve web applications or other visual interfaces and may also involve APIs.

Moving from cloud service A to cloud service B requires that the admin interfaces are compatible (particularly in the case of visual interfaces) and also interoperable (particularly in the case of APIs).

It may be the case that the admin interface is divided into separate sections dealing with particular capabilities – for example, monitoring and reporting capabilities may be delivered by one interface, while management and administration capabilities may be delivered by a different interface.

One approach is for the cloud service customer to use a management solution that has adapters or connectors for the different cloud service offerings, thus dealing with any differences between cloud service A and cloud service B.

Business Interface Considerations

The business interfaces apply to all forms of cloud services and include the capabilities relating to subscription management, billing and invoicing.

Moving from cloud service A to cloud service B requires compatible and/or interoperable business interfaces to ensure that the tools or program components used by the cloud service customer for business capabilities can be used successfully following the move.

Security Considerations

Security aspects of the cloud service include authentication and authorization of users and administrators of the cloud service, the configuration and operation of encryption for data stored within the cloud service and transmitted to and from the cloud service, firewalls and the configuration of other security capabilities.

The security aspects have a number of parts. Some parts apply to the running of the cloud service itself, other parts consist of the administration of the security components, including setting up or modifying user identities and the capabilities they are authorized to use. It is important to check that equivalent security capabilities available for cloud service A are available for cloud service B.

The support of some technologies by the cloud services makes the transition from one cloud service to another one potentially simpler. An example is the support of third party Id and Access Management function where it is possible that the IdAM system could be one installed and operated by the cloud service customer and where that system is used by both cloud service A and cloud service B with few changes - not requiring the porting of a large set of data about users and not requiring the need to change the interface used to administer the user data. Other standard technologies that can help include the increasingly common support of OAuth 2.0 and OpenID that permit use of commonly available ID and access management services across a range of cloud services.

Recommendations:

- For SaaS, ensure user interfaces, APIs, protocols and data formats are well-defined for cloud services. Whenever possible, insist on standard APIs, protocols, and data formats.
- For PaaS, ensure that the application dependencies are based on open technologies to increase the number of viable alternative cloud service providers which can facilitate migration if a change in provider is warranted.
- For IaaS, ensure that the cloud service accepts standard or widely accepted application packaging formats such as OVF and Docker and that any interfaces and APIs are open and/or standard.
- Insist that your cloud service provider supports key open technologies (open standards and/or open source) for admin and business interfaces.
- Leverage the support of third party ID and Access Management functionality to authenticate and authorize access to cloud services.

Scenario 2: Customer Uses Cloud Services from Multiple Providers

This scenario concerns the case where a cloud service customer uses one (or more) cloud services from cloud service provider A and one (or more) cloud services from cloud service provider B. The cloud service(s) from provider A might be equivalent in functionality to the service(s) from provider B, or they may have different functionality, depending on the business needs that the customer is aiming to satisfy.

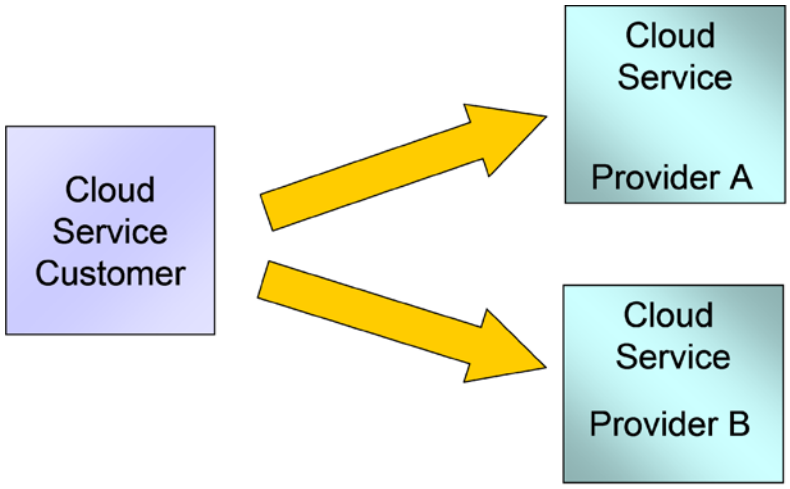

Figure 4: Customer Uses Cloud Services from Multiple Providers

An example where the cloud services would have equivalent functionality is the case where the customer uses two providers to get resilience (i.e., continue to access necessary computing resources in the case where one of the providers has a service outage). A different example is where the best SaaS service for one capability is offered by provider A while the best SaaS service for a second capability is offered by provider B and the customer wants to use both capabilities to satisfy business needs.

This scenario is mainstream. Enterprises have multiple choices when it comes to fundamental cloud services as well as a choice of unique or niche capabilities. Some cloud vendors may provide unique capabilities, some generally useful and some tailored for specific industry types A best of breed cloud implementation enables organizations to choose the best cloud service for each particular capability. From an interoperability and portability perspective, this scenario touches on many of the issues discussed in Scenario 1 above. The use of multiple providers has grown along with the adoption of open standards; however there is still no assurance that standards implementation will be consistent.

Interoperability Considerations

Whether the functional interface of cloud service 1 from Provider A needs to be interoperable with the functional interface of cloud service 2 from Provider B depends on whether the two cloud services are dealing with equivalent functionality or different functionality.

In the case where the cloud services deal with equivalent functionality then it is likely that the same customer components will interact with both services. As a result, it is best if the two services use the

same or interoperable interface(s). If services with equivalent functionality do not have interoperable interfaces then customer components will need to be updated to support the two interfaces – not an ideal situation.

When the two cloud services deal with different functionality, the need to use the same or interoperable interfaces is lower. Indeed, it is probably not reasonable to expect interoperable interfaces. However, there may be aspects of the two functional interfaces that should be based on the same technologies. One example is the technology used for Identity and Access Management, since this is a common feature of most cloud services.

Just as there is no assurance of standard interfaces between cloud service providers, neither is there assurance of consistency of cloud service agreement language, types, or terms and conditions. As complex, hybrid cloud architectures become the norm for business critical systems, identifying gaps and assuring equivalence of service availability, incident response, and compliance between cloud service providers is important for smooth operations. At minimum, a RACI chart (responsible, accountable, consulted, informed) for each integration point should be maintained. See the CSCC *Practical Guide to Cloud Service Level Agreements* [3] and *Practical Guide to Hybrid Cloud Computing* [5] for guidance. Also refer to ISO/IEC 19086-1:2016 which "seeks to establish a set of common cloud SLA building blocks (concepts, terms, definitions, contexts) that can be used to create cloud Service Level Agreements (SLAs)". Table 8 in ISO/IEC 19941 can help you identify critical focal points for review of agreements.

Portability Considerations

Data portability can be a requirement between cloud service 1 and cloud service 2. It is usually not a significant issue for IaaS services, since the cloud service customer is typically in control of the data syntax and semantics used for these services. For PaaS services, the cloud service customer usually has a lot of control of the data formats, but this may be limited where the PaaS services make use of particular database technologies – for example, in the case where cloud service 1 uses one database technology while cloud service 2 uses a different database technology.

For SaaS services, data portability can be a significant issue where cloud service 1 is equivalent to cloud service 2 (as in the case of scenario 1). Even in cases where the two SaaS services are not equivalent in functionality, there may be a need to use some data extracted from cloud service 1 in the operation of cloud service 2. In this case, it is best if the data extract has the same format, extent and semantics for cloud service 1 and cloud service 2. If not, then some form of data transformation may be required in order for the customer to use both cloud services successfully.

For IaaS and PaaS services which involve the deployment of application code into the cloud service, application portability between the cloud services is important. This is necessary where the same application gets deployed to cloud service 1 and cloud service 2. However, it can also be important in cases where different application code is deployed to the different cloud services, since the developers may well want to use the same knowledge, technologies and tooling to build both applications, which may be difficult if the capabilities offered by the cloud services differ substantially. For IaaS services, VM image formats and container formats are an important component of application portability. A portable

VM image format improves portability across different service providers. An example of a standard VM image format is Open Virtualization Format (OVF).

Admin Interface Considerations

Administering cloud services from multiple cloud service providers used to mean interacting with two or more sets of admin interfaces to monitor and manage each of the cloud services. If the interfaces were not interoperable, the result was duplication of effort. There might be multiple user accounts to manage, multiple sets of access controls to maintain, and multiple administration portals to learn and use. Even a relatively simple task, like checking to see which machine instances are running, can become a multi-step process. Cloud Management Platforms (CMP) able to support hybrid cloud architectures are becoming more common, as are tools that provide rules-based brokerage. A detailed explanation of CMPs is given in the CSCC paper, *Practical Guide to Cloud Management Platforms* [7].

CMPs support the ideal case, in which the customer staff use one set of tools and applications to monitor and manage all cloud services, irrespective of which cloud service provider is being used. In the ideal case, any admin APIs are interoperable and any admin visual interfaces are composable - for example, based on standard web technologies that can be integrated on a single browser. CMPs typically have a series of adapters which enable them to interact with a variety of different administration interfaces offered by different cloud service providers.

Business Interface Considerations

Using cloud services from provider A alongside cloud services from provider B implies a requirement on the cloud service customer to integrate the capabilities relating to subscription management, billing and invoicing. This is necessary in order to keep a good grip on expenditures and is also a requirement for allocating what are likely to be dynamic costs to the right internal budgets and projects. The CMP and cloud brokerage mentioned above frequently support the aggregation of business interfaces of multiple cloud services.

In the ideal situation, the business interfaces offered by Provider A are compatible and interoperable with those offered by Provider B. This enables the customer to use a single set of business tools to manage the usage of all the cloud services. Where the business interfaces are not all interoperable, then the customer should look for business tools such as CMPs that can perform mapping or transformation of each of the different business interfaces offered by the different providers.

Security Considerations

Using multiple cloud services from different providers requires various aspects of security to be carefully assessed. One aspect pertains to the running of the cloud service itself, other aspects pertain to the administration of the security components, including setting up or modifying user identities and the capabilities they are authorized to use or implementing directory federation. Additional complexity may be added if all or some portion of the organization's Identity and Access Management is running on or is itself a cloud service. The ideal situation is that these aspects are interoperable between the different providers, enabling a single set of tooling and procedures to be used by customer staff. In some cases this can be achieved by delegation of capabilities from the cloud service to the customer, such as the

support of third party Id and Access Management or directory federation, which may be a system owned and run by the customer. In other cases, the provision of capabilities using a standard interface and standard technology is a useful approach – for example, with respect to data encryption.

Recommendations:

- Refer to recommendations for Scenario 1.
- For SaaS, ensure user interfaces, APIs, protocols, and data formats are identical (or have a clear mapping) for equivalent functionality running on different cloud service providers.
- Consider implementing an Enterprise Service Bus (ESB) to perform interface, protocol, and data transformations to address differences between cloud services from different providers.
- For PaaS, ensure that the application dependencies (web server, database server, etc.) supported by different cloud service providers are compatible.
- Consider the use of an intermediary, an "inter-cloud provider", to help address and solve the issues of integration, interoperability and portability of multiple cloud services. Alternatively, make use of tools such as CMPs that can integrate with multiple cloud services.

Scenario 3: Customer Links One Cloud Service to Another Cloud Service

In this scenario, the customer uses two cloud services, but one of the cloud services is used directly by the other one. The ability to link cloud services together in support of a single application or an integrated set of applications is a useful approach where different cloud services can each provide specific capabilities which can be even more effective when linked together. Business technology leaders now understand that a single cloud service provider may be challenged to meet the needs of their entire organization.

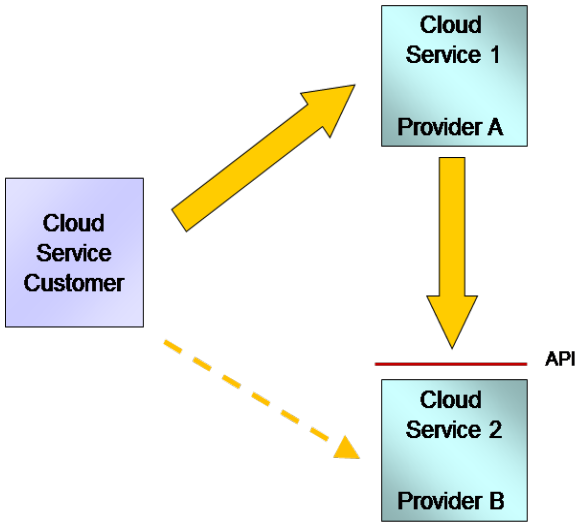

Figure 5: Customer Links One Cloud Service to Another Cloud Service

A cloud service linking to another cloud service as shown in Figure 5 can be illustrated using a SaaS-to-PaaS example. In this example, the SaaS application delivers the functionality/feature sets that an organization needs as a business solution; however, the SaaS application may not have the advanced functionality to deliver analytics and business intelligence. The organization can leverage additional capabilities by using a PaaS service from another cloud service provider and developing a custom analytics application that consumes data from the SaaS solution via an API, combining it with other data sets to drive additional revenue or market differentiation.

Interoperability Considerations

Linking one cloud service with another cloud service requires that the second cloud service has a well-defined API that the first cloud service can utilize remotely. It is assumed that the cloud services leverage SOA techniques (such as REST interfaces, stateless interactions) to facilitate invocation. The impact of performance variations on the use of the API due to network constraints should be considered.

The categories of the two cloud services influence the nature of the connection between the two services. When the first cloud service is either an IaaS or a PaaS service, the application code running in that service belongs to the cloud service customer and the main concern is if the code can successfully utilize the API of the second cloud service. It is likely that the IaaS or PaaS platform will be capable of supporting invocations of remote service APIs, although there may be issues to consider relating to security capabilities, such as authentication credentials and encryption technologies.

If the first cloud service is a SaaS offering, the situation is more complex as the application code belongs to the cloud service provider. In this case, the application code of the first cloud service must be structured to enable the use of the API of the second cloud service. For this to be possible, it is a likely requirement that the API be standardized in some way (possibly a de-facto standard rather than a formally standardized one). In addition, there may be a need for the customer to configure the first cloud service to use the second cloud service.

Regarding the second cloud service, if it is an IaaS or PaaS service, then the code belongs to the customer and the customer is in control of the API which it offers to the first service. When the second cloud service is a SaaS service and the API is dictated by the cloud service provider, it is important that the API is offered using standard technologies. Ideally, the whole interface should be defined by a standard, but in the common case where it is not, the basic protocols should be standard (e.g., use of REST/JSON or REST/XML protocols and data formats). When the API is not standardized, which is common, there is the danger that the customer will get locked-in to the particular cloud service and find it difficult and expensive to move to a different provider. As a result, the customer should consider mitigation techniques (see the Interoperability and Portability Challenges section).

Portability Considerations

This specific scenario does not involve moving or transferring either applications or data from one system to another, therefore, portability is not an issue. Refer to scenarios 1 and 2 for portability considerations relating specifically to cloud services.

Admin Interface Considerations

For the customer to administer cloud services from different providers means that multiple admin interfaces need to be taken into account. Administering multiple providers can mean duplication of effort and the need to adapt to different interfaces from each provider. Refer to scenario 2 above for a thorough discussion on admin interface considerations.

Business Interface Considerations

For the customer to utilize cloud services from different providers requires that the customer deal with the business interfaces offered by each cloud service provider. Refer to scenario 2 for a thorough discussion on business interface considerations.

Security Considerations

From a security perspective, linking cloud services from different providers requires not only appropriate security measures for each cloud service individually, but also requires security measures to be applied to the connection from the first cloud service to the second cloud service. The connection between the two services will require appropriately strong ID and Access Management capabilities to be applied. This will require that the technology for this is supported at both ends of the connection (i.e., cloud service 1 must be able to send appropriate credentials when it invokes cloud service 2). It may also be required to encrypt the data sent between the two cloud services. This requires that they mutually support encryption technology of the right strength.

The support of third party ID and Access Management functionality, where it is possible that the IdAM system could be one installed and operated by the cloud service customer and where that system is used by the different cloud service providers, would significantly reduce the need to port/duplicate user security information.

Recommendations:

- Refer to recommendations for Scenario 2.
- Ensure that services provided by the cloud service providers leverage SOA design principles and can utilize and expose APIs to enable interoperability.
- Consider the use of an intermediary (an “inter-cloud provider”) to help address and solve the issues of integration, interoperability and portability of multiple cloud services.
- Ensure that the security technologies supported by the second cloud service are usable by the first cloud service when it uses the capabilities of the second cloud service.

Scenario 4: Customer Links In-house Capabilities with Cloud Services

As more enterprises consider their cloud computing strategy, they will inevitably face the challenge of addressing how they will leverage their existing in-house IT investment with their newly adopted cloud services. In addition, enterprises will also have to assess how other in-house capabilities will be leveraged in their cloud strategy. These capabilities include people, processes, and of course technology. Developing an 'in-house' view of cloud adoption based on these critical criteria is the challenge that enterprises face during the early cloud computing adoption phase and continues as more workloads and projects are committed to cloud services.

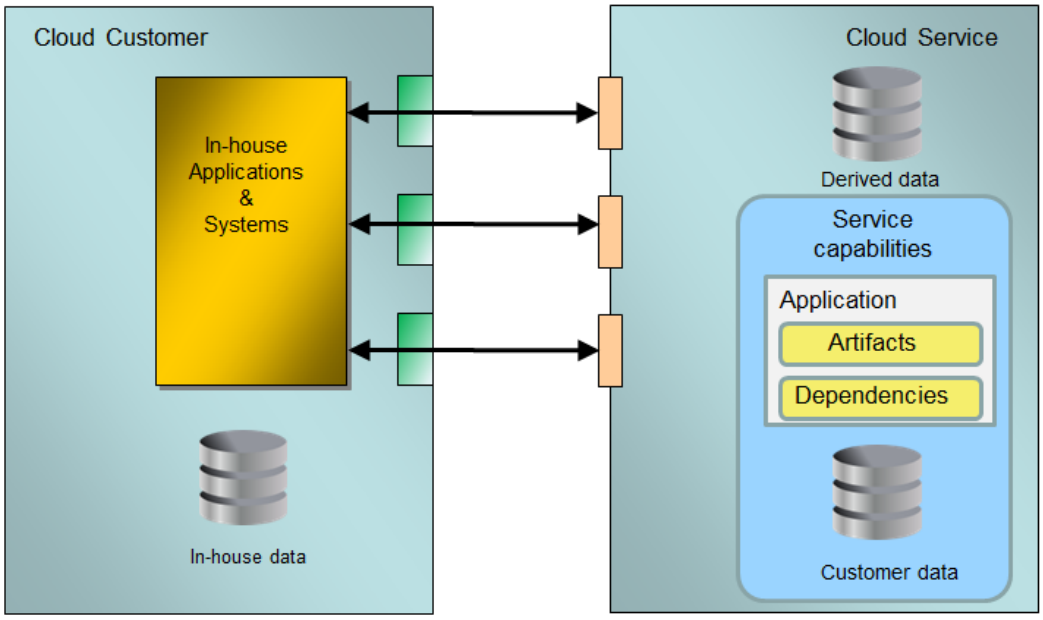

Figure 6: Linking In-house Capabilities to Cloud Services

As new cloud services are deployed, the need to connect them with various on-premises applications and systems becomes important, as illustrated in Figure 6. Cloud service owners need to understand the impact of these connections and address it. Integration between applications is typically classified into three types:

- Process (or control) integration, where an application invokes another one in order to execute a certain workflow
- Data integration, where applications share common data, or one application's output becomes another application's input
- Presentation integration, where multiple applications present their results simultaneously to a user through a dashboard or mashup.

The purpose of these integrations may be to perform an end-to-end workflow that crosses the boundaries between multiple business capabilities or systems (for example, entering a transaction in an accounts receivable system when a customer places an order in an e-commerce application). Another

form of integration is when the cloud service must continue to be monitored and managed by an existing suite of on-premises IT tools.

Refer to the CSCC whitepaper, *Migrating Applications to Public Cloud Services: Roadmap for Success* [8] for a detailed discussion on integration considerations for connecting cloud services with on-premises services.

Interoperability Considerations

Linking in-house capabilities with cloud services requires that on premises functionality and data needed by the cloud service are clearly identified and vice versa. For each of these functions and data sources, there must be a well-defined API in place that can be utilized remotely. If the on premises applications and cloud services leverage SOA techniques (REST interfaces, stateless interactions, etc.) then the integration effort should be reduced. If not, the impact of redesigning the applications and services to provide suitable interfaces could be significant. In addition, the impact of performance changes due to network constraints need to be considered as part of the integration.

For SaaS services, the cloud service provider must make available the necessary information (such as API descriptions and security requirements) to address functional integration requirements since the provider controls the cloud service. For PaaS services, the customer should be able to address most of the functional integration requirements, since the app code running in the cloud service is controlled by the customer, although provider assistance may be needed to address middleware specific integration requirements. For IaaS cloud services, the cloud customer should be able to fully address all functional integration requirements since it is expected that the interfaces offered and used by the app code running in the cloud service are controlled by the cloud service customer.

Portability Considerations

This specific scenario does not involve moving or transferring an entity from one system to another, therefore, portability is not an issue. However, it is recommended that any design required considers interoperability and portability to minimize impacts should migrations take place in the future. Refer to scenarios 1 and 2 above for portability considerations relating specifically to cloud services.

Admin Interface Considerations

Linking cloud services with on premise capabilities will require the integration of administration platforms and processes. As part of the design of this hybrid platform solution, careful consideration needs to be given to the design of the integration and deployment process within the Software Development Lifecycle (SDLC). This is necessary to enable appropriate management of change across the service providers (internal and external).

In addition to changes that are likely to be required to the SDLC process, system monitoring, management and support models (for example, code configuration management) should link in-house technology with cloud-based technology to formulate a unified solution. Techniques, such as Continuous Integration and Continuous Deployment (CI/CD) are commonly used in these circumstances.

Existing support models and processes must be revisited to determine if there is a need to modify the current processes to address potential disparities between the on premise components and the cloud services. Critical functions such as backup/recovery, disaster recovery, fail-over and high availability requirements must be re-examined with cloud services as a part of the scope and planning process.

Maintenance plans and support processes must also be re-visited to determine how the integrated solution will be monitored, managed and maintained on an ongoing basis.

Ideally, a consolidated monitoring and management platform is possible as production environments require constant maintenance and support to ensure performance, security, and availability to meet the expectations of the business units and their user base. Can the management tools used in-house still be used, or is it necessary to adapt to new monitoring and management facilities supplied by the cloud service? The answer to these questions will depend on the admin interface offered by the cloud service(s) and in particular whether the interface conforms to existing interoperable standards, which can be used by the in-house tools and systems. It may be necessary to consider adapter code or the use of mapping capabilities such as an ESB to facilitate the integration in cases where the cloud service admin interface is not directly interoperable with the in-house systems.

Business Interface Considerations

Business interfaces involve capabilities relating to the business aspects of the cloud service including subscription information, billing and invoicing. Many organizations do not adopt internal service billing (for many different reasons), but public cloud services may require the implementation or amendment of such systems.

As a result, the introduction of cloud services may require customers to adopt new business-related user interfaces and/or APIs defined by the cloud service provider and possibly involving the purchase or development of in-house systems for the management of business capabilities. In cases where on premise business management systems do exist, cloud service customers may need to adopt new tools and systems, or acquire adapters to match existing systems to the interfaces offered by the cloud service provider. Wherever possible, standard methods or data formats should be used to share data between providers and customers.

Security Considerations

From a security perspective, linking in-house capabilities with cloud services may require strong authentication and authorization services to ensure proper access is being granted to potentially sensitive services and data. The support of third party Identity and Access Management (IdAM), where it is possible that the IdAM system used to control access to the application running in the cloud service could be an existing one installed and operated by the cloud service customer, significantly reduces the need to port or duplicate user security information. The analysis of the IdAM system should include not just the application layer, but also access to the underlying infrastructure services (network, IaaS, PaaS or SaaS) to ensure no security holes are opened.

Given that the connection to the cloud service is likely to traverse the public internet, consideration should be given to what encryption should be applied to that connection and whether any customer data stored in the cloud service should also be encrypted. It is worth considering carefully whether there are any regulatory or compliance rules that specifically define the need or level of encryption to be used.

Not only must the cloud service support the necessary encryption capabilities, but items such as certificates and encryption keys must be managed in a way compatible with the customer's security policies. Technologies such as Virtual Private Network can also be considered for securing access to the cloud service.

In the case where the cloud service accesses APIs or data that are supplied by in-house systems, there is a further, very significant set of security considerations that must be addressed. The APIs and/or data access implies the creation of new, publicly exposed interfaces, which need careful control since they offer a potential new attack surface. Access control, firewall configuration, denial-of-service countermeasures, and encryption techniques must all be considered in relation to these new APIs. The deployment of suitable API Management capabilities may be part of the response to these considerations.

Recommendations:

- Refer to recommendations for Scenario 1.
- Ensure that on-premises applications are leveraging SOA design principles and can utilize and expose APIs to enable interoperability with remote cloud services.
- Examine whether existing in-house systems are available to deal with the business aspects of using cloud services. If they are not available, consider installing new systems to cover these aspects; if they are available, consider how those systems can connect to the business capabilities of the cloud service(s).
- Consider implementing an Enterprise Service Bus (ESB) to perform interface, protocol, and data transformations to address differences between on-premises systems and cloud services.
- If cloud service(s) need access to on-premises APIs or data, address the security issues raised by enabling access to these capabilities from the cloud environment - for example, put in place suitable API Management capabilities to prevent unauthorized access.

Scenario 5: Migration of Customer Capabilities into Cloud Services

This scenario addresses the case of a customer currently running an application or service on premises who moves that capability to a public cloud environment, as shown in Figure 7. This use case is a key one from the perspective of the customer since cloud computing offers several benefits including elasticity, potential cost savings and business transformation improvements.

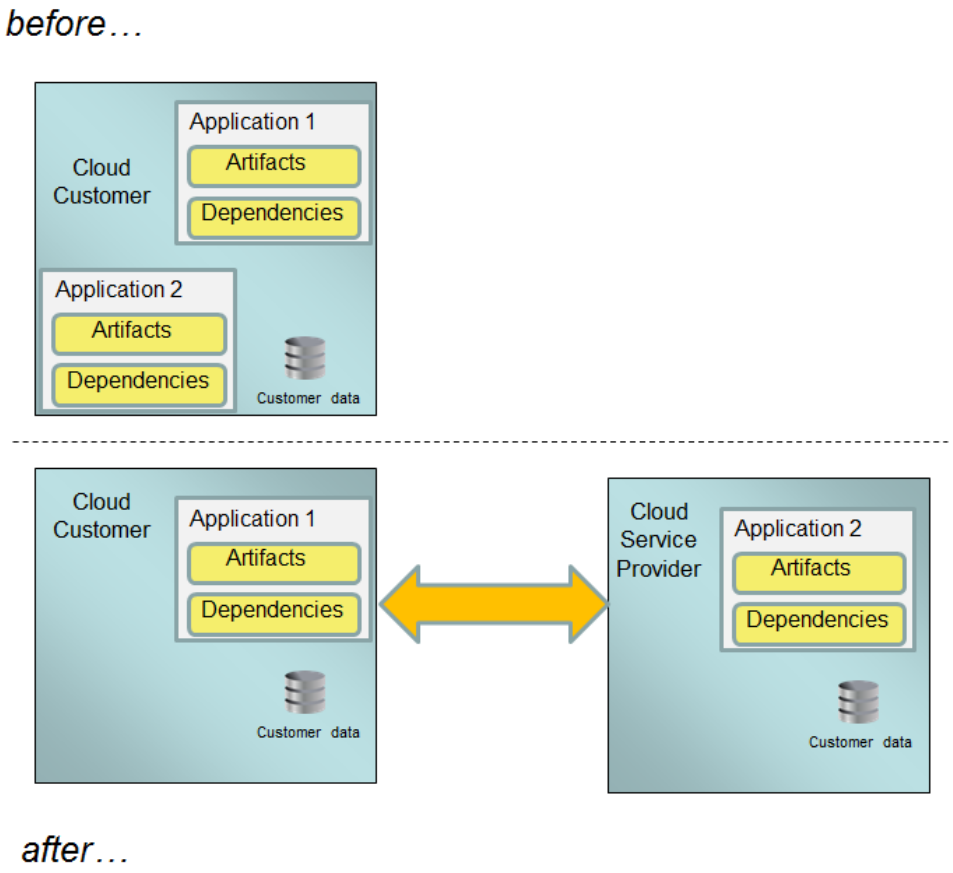

Figure 7: Migration of Customer Capabilities into Cloud Services

Assessing applications and workloads for readiness for migration to a cloud service is required for organizations to determine which applications and data can – and which cannot – be readily moved to a public cloud environment and what service models (IaaS, PaaS, or SaaS) can be supported. It often makes sense to start with the lowest-risk applications—those with minimal customer data and other sensitive information—or applications that can take advantage of the elasticity of cloud computing.

The table below highlights suitable and less suitable types of applications for migration to cloud computing. Refer to the CSCC whitepaper, *Migrating Applications to Public Cloud Services: Roadmap for Success* [8] for a detailed discussion on the steps customers should take to ensure successful migration of existing applications to cloud computing.

Suitable Candidates for Cloud	Less Suitable Candidates for Cloud
• Applications that do not contain extremely sensitive data. • Applications that run across distributed locations to service a distributed user base. • Applications that are required to serve large volumes of data (eg., streaming services). • Applications that are used by a group of mobile workers to manage their time and activity. • Applications that are run infrequently but require significant computing resources when they run. • Development, testing and prototyping of application changes, even if the final applications will be run on your own infrastructure. • Service Oriented Architecture (SOA) applications.	• Applications that involve extremely sensitive data, particularly where there is a regulatory or legal risk involved in any disclosure. • Applications that require frequent and/or voluminous transactions against an on-premises database that cannot be migrated to a cloud service. • Applications that run on legacy platforms that are typically not supported by cloud providers.

Interoperability Considerations

For SaaS cloud services, the application code belongs to the cloud service provider. In this case, migrating an on-premises application or service to a public cloud service provider does not involve porting the application code (i.e., the on-premises application is being replaced by the application offered as a cloud service). What is important in the SaaS case is the compatibility of the functional interface for the application - and, in particular, any user interfaces presented to end users and also any APIs made available to other customer applications. It is unrealistic to expect that user interfaces provided by the SaaS application will be identical to the on-premises application; however, it is desirable for the functionality to be presented in a broadly similar way to reduce the cost and effort of retraining end users.

Moving from using an on-premises application to a SaaS cloud service will likely require end user training and/or modifications to existing business processes, in that the cloud service is unlikely to be an exact match for the original on-premises application. This must be factored in to the planning for the migration.

Any functional APIs made available by a SaaS service should in the ideal case be the same as the interface provided by the on-premises application or service that is being replaced (i.e., the APIs are ***interoperable***). Where the APIs are not interoperable, then the implication is that the customer applications using the APIs will need to be changed as part of the process of migration to the SaaS equivalent. Depending on the extent of customer applications affected and to minimize change, it may

be advisable to create a common mapping layer that converts old API calls to the format required by the new SaaS offering or deploy an Enterprise Service Bus (ESB).

In the case of PaaS and IaaS cloud services, the user interfaces presented to end users and the APIs presented to external applications are likely to be the responsibility of the application code - and this code is the responsibility of the cloud service customer. However, these interfaces may be impacted when the on-premises application is migrated to a public cloud service due to the nature of the network connection from the on-premises location to the cloud service. If the on-premises application already leverages SOA techniques (REST interfaces, stateless interactions, etc.) then the impact of migration is minimized. If not, the impact to other applications invoking the migrated application or service could be significant. In addition, the impact of performance changes due to network constraints need to considered as part of the migration.

Portability Considerations

For SaaS services, it is typical that the format and the content of the cloud service customer data is in the hands of the cloud service provider. Thus, data portability is a major consideration when migrating an on-premises application to a public cloud SaaS service. Ideally, the data format and data content (syntax and semantics) should remain the same after SaaS migration. Data portability can still be achieved even if the data formats are different between the on-premises application and the SaaS application since there are straightforward standard tools that can be used to perform data transformations – and where there are not standard tools, it may be straightforward to build a custom tool. Differences in the extent or in the semantics of the data are much more serious and could be a major barrier to achieving data portability, particularly if the SaaS application requires more data than is available from the on-premises application, or if the SaaS application has a different meaning for some of the data.

The impact of migrating an on-premises application to a PaaS cloud service should have minimal impact on *data portability* unless data format and data content need to be altered as part of the migration. There may be exceptional cases where a database provided as part of the PaaS environment may be sensitive to the data format of the customer data, although there are generalized formats (CSV, XML, etc.) which are supported by many types of database. How data is loaded into a PaaS cloud service and how it is retrieved needs to be examined by the customer. Migration from an on-premises data store to a PaaS data store may involve data portability questions that need to be answered.

For PaaS migration, since the application code belongs to the customer the question of *application portability* becomes important – what does it take to move the on-premises application to the PaaS cloud service? One of the most important factors for application portability is represented by the ***App environment*** shown in Figure 2 above. In effect this represents the "API" that the cloud service presents to the application code – and the application code must be able to use this API in order for the application to run. In the ideal case, application portability implies that the application code which runs on-premises will run on the PaaS service without any changes (i.e., the App environment is compatible).

If the App environments differ, the application code will need to be changed to account for the differences.

The requirements for migrating an on-premises application to an IaaS service tend to be lower since the entire software stack is migrated: the application code itself, plus any supporting code it requires, potentially including the underlying operating system. To achieve this, it must be possible to package the complete software stack as one or more virtual machine (VM) images, which can then be imported into the cloud service and executed there.

Whether the software stack involved will work in a virtual machine environment may depend on whether there is use of specialized device drivers or hardware devices that are unlikely to be supported by an IaaS cloud service provider; an application depending on these capabilities is not a good candidate for migration. This tends to be an issue for legacy or specialist platforms only.

Assuming the software stack will work in a virtual machine environment, the question of *application portability* becomes less of an issue for migration to an IaaS service. That is, the on-premises application should run on the IaaS service with few if any changes. For IaaS services, the data format for cloud service customer data is usually in the hands of the customer, since the facilities provided by the cloud service are typically relatively low level, such as providing volumes for binary file or object storage (i.e., the cloud service does not know or care about the detailed format of the customer data). As a result, the impact of migrating an on-premises application to an IaaS cloud service should have minimal impact on *data portability*. It is important to note that to ensure application performance the database and application code generally need to be migrated together.

One consideration which applies to the migration of customer application code to either a PaaS or IaaS cloud service is whether the code is capable of taking advantage of the capabilities of the cloud service. It may be possible to port the code to the cloud service and get it to run successfully, but to enable capabilities such as elasticity and scaling may require recoding or redesign. However, this may be handled as part of a phased migration process, where full exploitation of the capabilities of the cloud service is part of the later phases, following migration of the basic functionality of the customer application.

Admin Interface Considerations

The management and monitoring of the migrated application running in the cloud service must be considered including the capabilities to deploy and configure the application, modify the resources assigned to the application (CPU, storage, etc.) and monitor the application's usage and status. Can the tools used in-house still be used, or is it necessary to adapt to new monitoring and management facilities supplied by the cloud service? These considerations could be some of the most significant when migrating an application to a cloud service, since it is likely that the in-house facilities for monitoring and control will not match the equivalent facilities available with the cloud service.

These admin interfaces may involve web applications or other visual interfaces and they may involve APIs. Migrating an on-premises application to a cloud service requires that the admin interfaces are compatible (particularly in the case of visual interfaces) and also interoperable (particularly in the case of APIs). If not, significant and potentially costly adaptation will be required. In the most extreme cases, it may be necessary to change the customer systems used for the various administration tasks, introducing new systems capable of interacting with the facilities made available with the cloud service.

Business Interface Considerations

Business interfaces involve capabilities relating to the business aspects of the cloud service including subscription information, billing and invoicing. It may well be the case that such capabilities do not exist at all for an on-premises application – and even where they do exist, it is likely that the interface(s) provided to support them will not match the equivalent interface(s) made available by the cloud service provider.

Moving an on-premises application to a cloud service provider requires compatible and/or interoperable business interfaces to ensure that the tools or program components used by the cloud service customer for business related capabilities can be used successfully following the move. In the most extreme case, this may involve the customer adopting new tools and systems, or acquiring adapters to match existing tools to the interfaces offered by the cloud service provider.

Security Considerations

Refer to scenario 4 above for a thorough discussion on security considerations.

Recommendations:

- Refer to recommendations for Scenario 4.
- For SaaS, consider compatibility with on-premises applications and the migrated cloud service. Ensure user interfaces, APIs, protocols and data formats are well defined for migrated cloud services. Whenever possible, insist on standard APIs, protocols and data formats.
- For PaaS, ensure that the application environment (web server, database server, etc.) supported by the cloud service provider is compatible with your on-premises application environment.
- Examine the cloud service provider interfaces for administration and business capabilities and ensure that they can be used directly or integrated with existing or new in-house systems.

Summary of Key Considerations and Recommendations

Here is a summary of the key considerations and recommendations for cloud service customers as they assess the interoperability and portability strengths and weaknesses of potential cloud service providers:

- Ensure that on-premises applications are leveraging SOA design principles and can utilize and expose APIs to enable interoperability with remote cloud services.
- Consider virtualizing applications and their dependencies using containers, which are more widely supported than VMs.
- Consider implementing an Enterprise Service Bus (ESB) to perform interface, protocol, and data transformations to address differences between on-premises applications and cloud services as well as differences between cloud services from different providers.
- For SaaS, ensure user interfaces, APIs, protocols, and data formats are well-defined for cloud services:
    - Whenever possible, insist on standard APIs, protocols, and data formats
    - Consider compatibility of on-premises applications and the cloud services
    - Consider compatibility of cloud services provided by different cloud service providers
- Address the following challenges for PaaS cloud services:
    - For application migration purposes, ensure that the application environment (web server, database server, etc.) supported by the cloud service provider is compatible with your on-premises application environment.
    - For portability between cloud services, ensure that the application environment is based on open technologies to increase the number of viable alternative cloud service providers which can facilitate migration if a change in provider is warranted.
- For IaaS cloud services, ensure that the cloud service accepts standard or widely accepted application packaging formats such as OVF or Docker and that any interfaces and APIs are open and/or standard.
- Insist that your cloud service provider supports key open technologies (open standards and/or open source) for admin and business interfaces:
    - Common, open interfaces make it easier to support multiple providers simultaneously
    - Common, open interfaces make it easier to switch from one provider to a different provider
- Address the following security challenges:
    - Leverage the support of third party ID and Access Management functionality to authenticate and authorize access to cloud services - ideally either a system owned and run by the customer or equivalent capability provided by a supplier, which could itself be a cloud service.
    - If cloud services need access to on-premises APIs or data, address the security issues raised by enabling access to these capabilities from the cloud environment - for example, put in place suitable API Management capabilities to prevent unauthorized access.

    - When using multiple cloud service providers, ensure that the security technologies supported by the second cloud service are usable by the first cloud service when it uses the capabilities of the second cloud service.
- Examine whether existing in-house systems are available to deal with the business aspects of using cloud services. If they are not available, consider installing new systems to cover these aspects; if they are available, consider how those systems can connect to the business capabilities of the cloud service(s).
- Examine the cloud service provider interfaces for administration and business capabilities and ensure that they can be used directly or integrated with existing or new in-house systems.
- Ensure key interoperability and portability requirements are included in your cloud service agreement.
- Consider the use of an intermediary (an "inter-cloud provider") to help address and solve the issues of integration, interoperability, and portability of multiple cloud services.
- Consider the use of a Cloud Management Platform (CMP) to manage cloud services. See the CSCC *Practical Guide to Cloud Management Platforms* [7] for more details.

Standards for Interoperability and Portability

To date, most of the focus for cloud interoperability and portability standards has been at the IaaS layer although activity at the PaaS level is starting to accelerate. In addition, there are several security standards that enable and facilitate cloud computing interoperability even though they are not exclusive to cloud computing. Cloud computing customers should determine the level of support for the following standards by prospective cloud service providers. Lack of support for these standards is likely to result in interoperability and portability challenges down the road.

- Open Virtualization Format (OVF). A packaging standard developed by the Distributed Management Task Force (DMTF) that is designed to address the portability and deployment of virtual machines.
- Cloud Data Management Interface (CDMI). A standard defined by the Storage Networking Industry Association (SNIA) that defines the functional interface that applications will use to create, retrieve, update and delete data elements from the cloud.
- Open Cloud Computing Interface (OCCI). A set of open specifications delivered through the Open Grid Forum that defines a protocol and API for all kinds of cloud computing management tasks.
- Topology and Orchestration Specification for Cloud Applications (TOSCA). A standard developed by OASIS that enables the interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behavior of these services (e.g., deploy, patch, shutdown).
- Cloud Application Management for Platforms (CAMP). A standard developed by OASIS that defines an interoperable protocol that cloud implementers can use to package and deploy their applications.

- Cloud Auditing Data Federation (CADF). A standard developed by DMTF that defines open standards for cloud auditing.
- LDAP, OAuth, OpenID Connect and SAML. Standards that enable third party ID and Access Management functionality.
- US FIPS 140-2. A standard that specifies the security requirements to be satisfied by a cryptographic module utilized within a security system protecting sensitive information.
- ISO/ IEC 19941. An ISO standard for Cloud Computing Interoperability and Portability [6].
- Open Container Initiative Image specification. Defines a common container image.
- Open Container Initiative Runtime specification. Defines how to run the contents of an unpacked container image.
- Kubernetes container management platform. Defines how to deploy and manage Docker / OCI container images.

At the time of this writing (fall 2017), IEEE and NIST are also working on P2302 Standard for Intercloud Interoperability and Federation (SIIF) https://standards.ieee.org/develop/project/2302.html. NIST will focus on the definitions and conceptual models, where IEEE will more focus on enabling such interoperability.

In addition to standards, there are a number of open source projects that are having a positive impact on cloud computing interoperability and portability. Open source projects that have an open governance model (i.e., not controlled by a single company) and attract a broad supporting ecosystem are the best candidates for creating de facto standards. In the IaaS space, OpenStack is an example of an open source project that is building significant industry momentum. Open source projects for PaaS are emerging. Examples of PaaS open source projects that have significant industry support include Cloud Foundry, Heroku, OpenShift and Open Containers Initiative.

Appendix A: Interoperability Model for Cloud Computing

When considering interoperability between two systems, it is useful to have a model for the information exchange, since there are in practice a number of aspects to the exchange that are relevant when connecting systems.

There are a number of interoperability models. One that is most relevant to the customer use of cloud services is the 5 facet model described in the ISO/IEC 19941 standard [6].

Facet	Aim	Objects	Solutions
Transport	Data transfer	Signals	Protocols of data transfer, e.g., REST over HTTP; MQTT
Syntactic	Understand format of transferred data	Data	Standardized data exchange formats, e.g., XML
Semantic data	Interpretation of transferred data using a data model	Information	Common data models, e.g., OData, OWL
Behavioral	Get anticipated outcomes when making service requests	Programmatic interface	UML models, pre-conditions, post-conditions, constraint specifications
Policy	Ensure that interacting systems conform to applicable laws, regulations and organizational policies	Laws, regulations, policies	Conditions for operation

Exchange of information between two systems can be considered as having 5 facets, independent of each other and each of which must be addressed to ensure interoperability:

- *Transport* is the communication infrastructure used to exchange data – for example, the use of the REST HTTP protocol over TCPIP.
- *Syntactic* concerns the format of the data that is exchanged – examples include XML data structures or JSON data streams.
- *Semantic data* concerns the meaning of the data exchanged – examples include ontologies (say using OWL).
- *Behavioral* concerns the expected outcome of the exchange of information – that is, the sending system has an expectation that the receiving system will use the exchanged data in a specific way, typically as part of some larger overall process.

- *Policy* concerns the context in which the data is exchanged – the laws, regulations and policies that apply to both sides of the exchange, which may place constraints on the exchange, or in the most extreme cases prevent the exchange taking place at all.

Ideally, two interacting systems satisfy interoperability for all 5 facets. However, practically speaking, two systems can interact successfully even if all 5 facets do not match. For example, for the transport facet, system A may communicate using the REST HTTP protocol while system B may communicate using the MQTT protocol. It may be possible to get them talking to each other by using a protocol adapter such as an ESB.

Similarly, if the two systems differ in the Syntactic facet, it may be possible to enable them to interoperate using a syntax translator. An example is a mapping between data encoded in XML versus data encoded in JSON.

Two systems which don't interoperate in the Semantic facet are likely to be much more problematic. The implication is that the two systems have different domain data models and it may be the case that data from one system has no meaning or is unusable in the other system. It may or may not be possible to create an adapter that can be used to enable the two systems to communicate successfully.

For the behavioral facet, a mismatch between the two interacting systems might make it tricky to achieve interoperability, in that one system has an expectation of an outcome of an interaction which is not achieved by the other system. It may not be possible to resolve such a mismatch.

For the policy facet, if there is a mismatch of the legal, regulatory, or policy elements applying to the two systems, either interoperability cannot be achieved (e.g., if the exchange is illegal in some way) or special measures may be required (e.g., redaction of some data elements in the exchange).

Appendix B: Portability Model for Cloud Computing

Portability between two systems must be considered separately for data and for applications, since the factors of relevance differ substantially between these. Each is considered separately from a model perspective. Both of the models presented here derive from the ISO/IEC 19941 standard [6].

Cloud Data Portability

Cloud data portability is described by a model with 3 facets as shown in the table:

Facet	Aim	Objects	Examples
Syntactic	Receive data in a readable structured format	Data	JSON, XML
Semantic	Understand the meaning of ported data	Information	OWL
Policy	Meet applicable laws, regulations and policies	Laws, regulations, policies	Personal data regulations, Cross border data transfer laws, Security policies

For successful data portability, all 3 facets must be satisfied. Syntactic mismatches might be handled by the use of a syntax converter to map the source syntax to the target syntax. Semantic mismatches may be handled by converting the data to the target data model. This may or may not be possible depending on the differences between the source and target data models. For the policy facet, either data portability might not be possible (e.g., if the movement of the data is illegal in some way) or special measures may be required (e.g., redaction of some data elements).

Cloud Application Portability

Cloud application portability is described by a model with 5 facets as shown in the table:

Facet	Aim	Objects	Examples
Instruction	Execute application instructions correctly	Executable artifacts	Java, C++, BPEL
Syntactic	Understand and use format of application artifacts	All application artifacts	Zip, tar, jar
Metadata	Understand and use the metadata that specifies environmental dependencies for executing the application	Metadata artifacts	YAML, JSON, Script, XML
Behavior	Produce the expected results when executing the application	Application functional and nonfunctional behaviors	Verified by test suites

Facet	Aim	Objects	Examples
Policy	Meet applicable laws, regulations and policies relating to application use	Laws, regulations, policies	Personal data regulations, Cross border data transfer laws, Security policies

For the application instruction facet, the target system needs to be capable of understanding and executing the instructions contained in the executable artifacts of the application. For some languages (Java, node.js) the instructions are "universal" and the requirement is that the target system has the runtime engine for these instructions (Java VM, etc.). For other languages that are compiled to a native instruction set, the requirement is that the target system can deal with that instruction set. If not, the artifacts may require recompiling for the target system.

The application syntactic facet requires that the target system is able to understand and use the format(s) used for the application artifacts. If not, the format(s) may need to undergo a conversion process.

The metadata facet deals with the metadata of the application and its artifacts. The metadata typically contains specifications for the environmental dependencies for executing the application. The target system must be capable of understanding the metadata and acting on it when attempting to execute the application. The metadata may need adapting to the capabilities of the target system.

The behavior facet deals with the functional and non-functional behavior of the application - this is typically defined by test suite(s) that check the behavior and ensure that it meets specific expectations. If the ported application fails elements of the test suite, then the application code and/or metadata may need adjustment.

The policy facet requires that the porting of the application is done within the applicable laws, regulations, and policies. Such things may include export regulations (if porting is across border, say), commercial considerations such as licenses relating to the application, and policies such as security policies defining access control and encryption requirements. Failures here may for example prevent application porting, or may require additional payments or the modification of metadata to match policy requirements.

Works Cited

[1] European Journal of ePractice. *Three Dimensions of Organizational Interoperability*.
http://www.epractice.eu/files/6.1.pdf

[2] Cloud Standards Customer Council (2017). *Practical Guide to Cloud Computing*.
http://www.cloud-council.org/deliverables/practical-guide-to-cloud-computing.htm

[3] Cloud Standards Customer Council (2015). *Practical Guide to Cloud Service Level Agreements*.
http://www.cloud-council.org/deliverables/practical-guide-to-cloud-service-agreements.htm

[4] Cloud Standards Customer Council (2016). *Public Cloud Service Agreements: What to Expect and What to Negotiate.*
http://www.cloud-council.org/deliverables/public-cloud-service-agreements-what-to-expect-and-what-to-negotiate.htm

[5] Cloud Standards Customer Council (2016) *Practical Guide to Hybrid Cloud Computing.*
http://www.cloud-council.org/deliverables/practical-guide-to-hybrid-cloud-computing.htm

[6] ISO/IEC 19941 *Cloud computing - Interoperability and Portability*
https://www.iso.org/standard/66639.html

[7] Cloud Standards Customer Council (2017). *Practical Guide to Cloud Management Platforms*.
http://www.cloud-council.org/deliverables/practical-guide-to-cloud-management-platforms.htm

[8] Cloud Standards Customer Council (2013). *Migrating Applications to Public Cloud Services: Roadmap for Success*.
http://www.cloud-council.org/deliverables/migrating-applications-to-public-cloud-services-roadmap-for-success.htm

[9] SNIA (2015): *Cloud Data Management Interface (CDMI)*
https://www.snia.org/cdmi

[10] Cloud Foundry: *Cloud Application Platform*
https://www.cloudfoundry.org/

[11] Distributed Management Task Force (2015): *Open Virtualization Format (OVF)*
https://www.dmtf.org/standards/ovf

[12] Docker: *Containerization Platform*
https://www.docker.com/what-docker

[13] Linux Foundation: *Open Containers Initiative*
https://www.opencontainers.org/

[14] Kubernetes: *Container Orchestration*
https://kubernetes.io/

Additional References

Cloud Technology Partners: *Driven by Multi-Cloud, the Public IaaS Market Begins to Equalize.*
http://www.cloudtp.com/2014/06/27/driven-multi-cloud-public-iaas-cloud-market-begins-equalize

INRIA: *Managing Elasticity Across Multiple Cloud Providers*.
http://hal.inria.fr/docs/00/79/04/55/PDF/Managing_Elasticity_Across_Multiple_Cloud_Providers.pdf

National Institute of Standards and Technology: *US Government Cloud Computing Technology Roadmap Volume II Release 1.0 (Draft) – Useful Information for Cloud Adopters.*
http://www.nist.gov/itl/cloud/upload/SP_500_293_volumeII.pdf

