

Connecting to the Cloud, Part 3: Cloud governance and security

Secure the HybridCloud application

Skill Level: Intermediate

Mark O'Neill
CTO
Vordel

16 Jun 2009

In the third and final part of this three-part series on building a hybrid cloud application, examine governance and security for cloud computing. Build on the example of the HybridCloud application from Part 2 by examining how to add access control policies to its use of Amazon Simple Queue Service (SQS). Look in detail at how the HybridCloud application authenticates itself to cloud services and how to add a log audit trail to Amazon's S3 (Simple Storage Service). Lastly, see how Google Apps uses OAuth and how Force.com cloud services require built-in testing to avoid inadvertent Denial-of-Service (DoS) attacks.

Introduction

Other articles in this series

- [Connecting to the cloud, Part 1: Leverage the cloud in applications: Take advantage of the hybrid model](#)
- [Connecting to the cloud, Part 2: Realize the hybrid cloud model: Pull JMS queue data to an Amazon SQS queue](#)

Cloud governance involves applying policies to the use of cloud services. It can be useful to think of cloud governance by examining its opposite: the free-for-all chaos in which cloud services are used by an organization without any oversight in place. To avoid this chaos, put polices in place for cloud service use to control the leakage

of private information to the cloud and to control the excessive use of cloud services (which must be paid for, after all). With governance and security in place, cloud computing can be used in safety and confidence.

Lessons from SOA governance

Governance is a word that came to prominence with the adoption of SOA. In the world of SOA, it was divided into design-time governance (defining policies to Web services) and runtime governance (actually applying those policies to real-time traffic).

Frequently used acronyms

- API: Application programming interface
- DSA: Digital Signature Algorithm
- IP: Internet protocol
- IT: Information technology
- REST: Representational State Transfer
- SOA: Service Oriented Architecture
- SSL: Secure Sockets Layer
- URI: Universal Resource Identifier
- WSDL: Web Services Description Language
- XML: Extensible Markup Language

Cloud platforms, like the services in an SOA, are predominantly accessed using Web service APIs, and so they might be expected to come under the same heading as SOA governance. At the very least, you can reuse the principles behind SOA governance.

SOA governance usually relies on the presence of a registry. This is a central place where a user can go to view the services in the SOA, and also to view the policies that apply to the services. The standard WS-Policy, as well as the complementary WS-Attachment specification, allows a policy to be assigned to a service in an SOA. In effect, the service then contains a "pointer" to its policy. The registry manages this relationship between the services and the policies.

Another important function provided by SOA governance products is the lifecycle management of services. This refers to the ability to control and track changes to services, and to place controls over who can change a service. Once this facility is in place, an organization can determine who created a service, who changed it, and when the changes were made.

Does the existence of SOA governance mean that cloud governance is a solved problem? Not quite. The messages sent to and from cloud services generally are not SOAP, and the services are usually not defined by WSDL, which are two standards used in general SOA governance. This means that it is not straightforward to import the services into an SOA governance registry. The Web services used by cloud computing bypass SOAP and WSDL and instead make use of lightweight REST-style services that are more popular with developers due to their relative simplicity.

Virtual machines are another new aspect of cloud computing that differentiates it from SOA. As well as making use of Web services, cloud computing also makes use of virtual machines. The Amazon Elastic Compute Cloud (EC2) environment can be seen as a kind of virtualized hosting environment, not a set of services. Governance of Amazon EC2 can therefore be seen as an example of governance of virtual machines. In particular, the virtual machine deployments can easily end up in chaos, as an organization ends up with many virtual machines that might differ slightly from each other in undocumented ways. As many users of VMware or Citrix Xen are aware, even for an individual user, it can be difficult to keep track of virtual machines. Imagine how the problem multiplies for an organization, and increases further if those virtual machines are hosted in a third-party cloud service.

Virtual Machine (VM) governance does, however, have some overlaps with the "lifecycle governance" benefits of SOA Governance. In the Amazon EC2 cloud world, a VM is an instance of an Amazon Machine Image (AMI). The ability to enforce rules such that certain users can deploy certain AMIs is important. At a finer level, the ability to control who can reboot a VM, who can add capacity to an existing VM environment, and who can delete existing virtual machine instances is important.

The last item in that list, namely the deletion of Amazon AMI instances, is particularly important since organizations pay for the use of these instances. Pricing is based on use (when the image is in a running state) and data traffic. Without a Cloud governance system in place, unwanted running AMI machine instances can proliferate and cause unnecessary cost. However, the opposite is also true: without a Cloud governance solution in place, it is possible that useful AMI instances might be mistakenly deleted. Lifecycle management of AMI instances avoids the problems of rogue instances, just as SOA Governance tackled the issue of rogue services which tend to proliferate in organizations without a governance framework in place.

A process, not a product

It is often said that SOA governance is a process, not a product. Cloud governance holds to this rule also. The ability to enforce governance rules depends upon developers being aware that the governance framework is in place. For example, if an SOA governance registry is deployed, a developer must be aware that this is where Web services (in particular, their WSDL definitions) must be registered.

Without this registration, the Web service flies under the radar. As with SOA governance, something similar happens with cloud governance if a developer is not asked to register their use of cloud services with their own organization.

In the world of SOA governance, in addition to the design-time governance provided by the registry, there is also the runtime governance whereby the rules configured in the registry are enforced. This enforcement point for services usually takes the form of an embedded agent or a network intermediary such as an XML gateway.

The tools of the SOA governance trade—the registry and the runtime enforcement point such as an agent or an XML gateway—are missing from the cloud computing world. This means that there is no central point for a user of a cloud computing service to view all the services and the associated policies. And although policies are enforced on the cloud computing side of the connection (by the cloud service provider itself), they are not enforced on the client side. This is a key challenge for the governance of cloud computing.

Rogue clouds

In many ways, cloud computing is like the early days of Web services. Developers would individually decide to use this new technology for projects, and go ahead and use it. This would happen undetected by corporate IT management. Then, belatedly, an umbrella of governance would be added. At the moment, most organizations are in the pre-governance stage when it comes to cloud computing. It is not unusual for a developer to dabble with an AMI image for a project; even the initial prototyping involves using their personal Amazon account and credit card.

Client-side governance missing in action?

A cloud service provider usually does not have to communicate service downtime information to clients ahead of time. Additionally, of course, when a service outage happens for unforeseen reasons, there is no onus on the cloud service provider to communicate this to the users of the cloud services. To monitor the response time and availability of cloud services, a client side component is required. The client-side component (for example, an XML gateway) monitors the connection to the cloud platform. If the connection is slow, then the XML gateway raises an alert, or takes remedial action such as using a response from its cache. By using a cache in this way, the effects of a cloud service outage can be mitigated.

The XML gateway on the client can also scan cloud-bound data for leakage of private or company-sensitive data. The data might also be encrypted, or selectively encrypted, prior to being sent up to the cloud provider. For example, an XML Gateway might ensure that data going up to a cloud computing provider is de-identified so private information cannot be associated with the data.

XML gateways, such as the Vordel XML Gateway Cloud Edition, filter traffic sent up to cloud platforms, as well as apply policies to the access to the cloud services. By doing so, XML Gateways provide the client-side on-ramp to cloud services.

Applying control to the HybridCloud application

In Part 2 of this series, you began to create the HybridCloud application, which made use of Amazon Web Services. To put cloud governance and security in perspective, you will see how this application is authenticated by Amazon Web Services, and how to apply policies to its use.

The keys to Amazon

The HybridCloud application makes use of the Amazon SQS cloud service. SQS, like all Amazon Web Services (AWS), requires the use of an *access key ID* and an associated secret key. We saw the Amazon keys being used within the HybridCloud Java™ code, hard-coding these keys into the application that is using the Amazon SQS cloud. But what are these keys? Let's examine them in detail.

What does RSA mean?

The letters "RSA" in the name of the RSA asymmetric encryption algorithm stand for "Rivest, Shamir, Adleman," who first wrote about the algorithm.

Readers familiar with Public Key Infrastructure (PKI) might assume that these two keys are in fact public and private keys that are linked in an asymmetric key pair, like the keys used by the RSA or DSA algorithms for digital signatures and encryption. However, they are not classical public and private keys. The Access Key ID is used as an identifier, identifying the party that accesses the AWS service. It is similar in concept to a user name and it can be sent in unencrypted requests. Indeed, when the Amazon S3 cloud service is used for online storage, the Access Key ID forms part of the URL. If a user has been assigned an Access Key ID of "12456789" then the URL used to retrieve a file which is stored in that user's bucket of files stored on Amazon S3 will be: `https://s3.amazonaws.com/123456789-bucketname/filekey`.

A *bucket*, as defined in Amazon S3, is simply a storage container for files. To define the space you will need on S3, create a bucket and give it a name which is unique across the Amazon system.

The Access Key ID is clearly visible in the URL, which means that it will also be stored in the logs of the networking infrastructure, which routes the requests to the Amazon S3 bucket to access the file. Therefore, any router or proxy will have access to the Access Key ID. It is not useful for authentication, since it is so easily

discovered. It really is for identification not authentication.

The Secret Access Key is what is used for authentication. However, this is not sent by the client to the AWS service. Instead, it is used to create a form of digital signature that is used to provide proof of possession of the Secret Access Key. This proof of possession is similar to how the SSL handshake protocol uses encryption to prove that a client is in fact the holder of a private key, without sending the key itself. The fact that a client can use the key indicates that the key is under control of that client.

In the case of a PKI public and private key pair, there is a mathematical relationship between the keys in question. Data encrypted using the private key can be decrypted using the corresponding public key. This is a basis for PKI-based digital signatures. Only the holder of the key can create the signature, whereas anyone with access to the public key (usually taken from an X.509 Certificate) can validate the signature.

The Secret Access Key in the Amazon Web Services model does not have these properties. You can instead think of it as a shared secret between Amazon.com and the developer who uses AWS resources such as its cloud services. It must not be shared with others, who might use it to access the Amazon cloud services. Because use of the cloud services is billed to the developer, it is vital that the Secret Access Key does not fall into the wrong hands; otherwise, a large bill might be run up. If you suspect that a third-party has accessed a Secret Access Key, you can generate a new Secret Access Key online.

In your HybridCloud application, the keys were hard-coded in the application itself. However, unless you use a Java obfuscator, an attacker might reverse-engineer the application, thereby discovering the keys. This is a good reason to use a Java obfuscator (see [Resources](#)).

Policies for the HybridCloud application

In Part 2 of this series, you saw that the HybridCloud application sends X-ray data to the Amazon SQS. As an X-ray image is clearly private medical data, this data sent to the Amazon SQS service must be scanned for private data on the client side. Once the data reaches Amazon SQS, it is too late. This is another reason to use a local gateway to connect to cloud computing resources.

Another security consideration for the HybridCloud application is to lock down access to the Amazon SQS service so that only trusted clients can access the service. This is achieved using an Amazon SQS policy. Amazon SQS policies are defined in JavaScript Object Notation (JSON).

Let's examine a number of Amazon SQS policies that you might apply to your HybridCloud application (for the details and source code of the application, see Part

2 in this series).

First, this policy specifies that only a developer with Amazon Web Services account number 1234567890 can access the Queue, which is owned by a developer with the resource URI /987654321/queue1 (see [Listing 1](#)).

Listing 1. Examining an Amazon SQS policy

```
{
  "Version": "2008-10-17",
  "Id": "12345678901234567890"
  "Statement":
  {
    {
      "Sid": "Queue1_SendMessage",
      "Effect": "Allow",
      "Principal": {
        "AWS": "1234567890"
      },
      "Action": "SQS:SendMessage",
      "Resource": "/987654321/queue1"
    }
  }
}
```

Now examine that policy in detail. You see first the version information. At the moment, the only valid value for that field is 2008-10-17. Next, you see the ID for your rule. It must be a globally unique identifier within Amazon Web Services. "Statement" is the actual policy. In it, you can see the Resource is the Amazon SQS queue. You are specifying that a particular "Principal" (in this case, a particular AWS user) can access the particular queue, and only that principal can access the queue.

You can also set policies that control access to our SQS queue based on date, time, and source IP Address, as in [Listing 2](#).

Listing 2. Setting policies that control access to the SQS queue

```
"Condition" : [{
  "DateGreaterThan" : {
    "AWS:CurrentTime" : "2009-04-10T09:00:00Z"
  }
  "DateLessThan" : {
    "AWS:CurrentTime" : "2009-04-10T17:30:00Z"
  }
  "IpAddress" : {
    "AWS:SourceIp" : ["4.3.2.1/24"]
  }
}]
```

In this case, the coded policy says that a client can only access the queue between 9:00 a.m. and 5:30 p.m. on April 10, 2009, and from a particular IP Address range.

You can use the `AddPermission` and `RemovePermission` functions to set and remove particular access rights for each queue.

The policies for the Amazon SQS operate at the cloud service provider, of course. However, if the data sent to the cloud provider is to be encrypted or scanned prior to sending, then a client-side XML gateway appliance is an ideal tool for this job.

Now that you know how to lock down access to the Amazon SQS service for your HybridCloud application, briefly look at cloud security solutions from Amazon, Google, and Salesforce.

Amazon S3

Amazon S3 (Simple Storage Service) is a cloud service that is used for online storage. It is used as the back-end storage facility by Web sites such as the photo-sharing site SmugMug (see [Resources](#) for a link). It can also, of course, be used for private corporate data. The data you store in Amazon S3 might be sensitive, depending upon the context. If your data is private, then it makes sense to encrypt the data using an XML gateway before you send it to the Amazon SQS service. Additionally, keep track of how the data is accessed from Amazon SQS. Besides the privacy issues, you will want to track S3 use simply because the use is billed. Nobody wants to be surprised by a large bill and left without a full audit of the use that led to that bill. For all these reasons, it is important to turn on logging for the Amazon S3 service.

A simple way to turn on logging for an Amazon S3 bucket is to use the Cloudberry Explorer applications to switch on logging for an S3 instance (see [Resources](#) for more information). Cloudberry Explorer effectively provides a GUI front-end to Amazon S3. In Cloudberry Explorer, you simply select the bucket and to click the **Logging** button on the toolbar. Now you check the **Use logging** check box. Choose the bucket that will contain your log files by selecting it from the drop-down list. In our example, it is `cloudberry.log`. Logs are actually written to the Amazon cloud service. Log files must be written to a bucket in the same geographical location (that is, US apps must write to US logs). Log files will add to monthly bills because they use Amazon S3 storage. However, this is (literally) a small price to pay for having an audit trail of users who have accessed your Amazon S3 storage service.

Google Apps

Google provides a tool called the Google Secure Data Connector (SDC) to connect client-side Java applications to the Google Apps cloud service. It provides an encrypted link between a local network and Google Apps. This is primarily used for Google Apps applications (hosted by Google) to connect to the local network. It is built to ensure that only Google Apps can connect over this connection. Additionally, it provides filters that limit which Google Apps gadgets can access which internal systems, much like firewall rules, but at an application level. In addition to controlling which applications can be accessed, you can add user level control. This allows an

organization to control who accesses what Google Apps service.

The identity framework for Google Apps SDS uses OAuth. For example, hosted Google Apps services can identify themselves (and their users) to local applications using OAuth. At present, OAuth is not widely deployed. However, Amazon's endorsement of OAuth might change that situation.

Force.com: Safeguards and "test as you go"

Other articles in this series

- [Connecting to the cloud, Part 1: Leverage the cloud in applications: Take advantage of the hybrid model](#)
- [Connecting to the cloud, Part 2: Realize the hybrid cloud model: Pull JMS queue data to an Amazon SQS queue](#)

As you saw in Part 1 of this series, Salesforce's Force.com cloud service uses a programming language called *Apex*. Force.com mandates that Apex code hosted on its cloud platform includes "asserts" in order to ensure that code is behaving as required. This must cover at least 75%, and preferably 100%, of a developer's Apex classes. This measure is in place to avoid inadvertent Denial-of-Service (DoS) attacks on the system as a whole. For example, code that might go into an infinite loop, or otherwise use excessive resources, must be wrapped in testing code that will detect such eventualities. As an added safeguard, Force.com also imposes limits (called *governors*) to enforce rules on such metrics as stack depth and string size.

Although other cloud providers do not have testing stipulations similar to Force.com, it is still a good idea to follow their recommendations even if you do not use Force.com as your cloud platform. As well as guarding against DoS attacks, you also ensure that you are not surprised by an unexpected usage bill from the cloud provider.

Summary

A number of initiatives such as the Cloud Security Alliance are in place to address cloud security. At the moment, providers such as Amazon, Google, and Force.com are taking the lead at the service provider level. However, not all are convinced: The Federal Trade Commission has been asked to investigate the risks related to cloud computing due to perceived data management problems. As awareness of cloud governance and security builds, it can be expected that cloud governance and security offerings will develop further.

Resources

Learn

- [Connecting to the cloud, Part 1: Leverage the cloud in applications: Take advantage of the hybrid model](#) (Mark O'Neill, developerWorks, April 2009): Examine what is available from the major Cloud Computing providers (Amazon, Google, Force.com, and Microsoft Azure) through an example of a typical corporate application which uses a JMS queue.
- [Connecting to the cloud, Part 2: Realize the hybrid cloud model: Pull JMS queue data to an Amazon SQS queue](#) (Mark O'Neill, developerWorks, April 2009): Link a corporate Java app to Cloud Computing platforms and examine how the app can leverage the Cloud through XML, SOAP, and REST APIs.
- [Leveraging Amazon Web Services for enterprise application integration: XML messaging with Amazon SQS](#) (Brian Stewart, developerWorks, June 2009): Combine Amazon Web Services and XML to integrate enterprise applications. Build a flexible, scalable sample app that is cross-platform and technology agnostic. Includes extensive code samples in C# and the Java language.
- [The Cloud Security Alliance](#): Explore this non-profit organization formed to promote the use of best practices for providing security assurance within cloud computing.
- [Storage made easy with Amazon S3: Enter the cloud with Amazon's Simple Storage Service](#) (Andrew Glover, developerWorks, April 2009): Learn to use the open source JetS3t library to leverage Amazon's S3 cloud service for storing and retrieving data.
- [Amazon Web Services](#): Read more about Amazon Web Services and cloud computing.
- [Cloud Computing with Amazon Web Services](#) (Prabhakar Chaganti, developerWorks, July 2008): Learn to build Web-scale systems with this step-by-step guide to using Amazon Web Services.
- [Private Clouds Take Shape](#): Learn about hybrid clouds in this article from Information Week.
- [Vordel XML Gateway, Cloud Edition](#): Learn about this gateway with connectors to JMS queues plus pre-built connectors to cloud services.
- [developerWorks Cloud Computing Resource Center](#): Find IBM products are available for the Amazon EC2 platform.
- [Open source Java obfuscators](#): Try this Java class file shrinker and obfuscator for smaller jar files that are harder to reverse-engineer.
- [The Google App Engine Blog](#): Visit a great place to follow its development.

- [Navigate the cloud computing labyrinth](#) (Brett McLaughlin, developerWorks, March 2009): Make an educated decision about the best cloud computing platform for your particular application requirements.
- [Connecting Apple's iPhone to Google's cloud computing offerings](#) (Noah Gift and Jonathan Saggau, developerWorks, January 2009): Learn about making the cloud accessible on mobile devices.
- [Data integration with Salesforce CRM using IBM InfoSphere Information Server](#) (Jon Deng and Jeff J. Li, developerWorks, July 2008): Find out about how Salesforce makes its data accessible to your applications.
- [IBM XML certification](#): Find out how you can become an IBM-Certified Developer in XML and related technologies.
- [XML technical library](#): See the developerWorks XML Zone for a wide range of technical articles and tips, tutorials, standards, and IBM Redbooks.
- [developerWorks technical events and webcasts](#): Stay current with technology in these sessions.
- [developerWorks podcasts](#): Listen to interesting interviews and discussions for software developers.

Get products and technologies

- [Cloudberry Explorer](#): Try a free Windows® client for Amazon S3.
- [SmugMug](#): Visit the photo sharing Web site, which uses Amazon S3 as its back-end storage facility.
- [IBM product evaluation versions](#): Download or [explore the online trials in the IBM SOA Sandbox](#) and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- [Participate in the discussion forum for this content.](#)
- [XML zone discussion forums](#): Participate in any of several XML-related discussions.
- [developerWorks blogs](#): Check out these blogs and get involved in the [developerWorks community](#).

About the author

Mark O'Neill

Mark O'Neill is CTO at Vordel, an XML Networking company. He is also author of the book "Web Services Security" and contributing author to "Hardening Network Security", both published by McGraw-Hill/Osborne Media. Mark is responsible for overseeing Vordel's product development roadmap and also advises Global 2000 firms and governments worldwide on their tactical and strategic adoption of XML, Web Services and SOA technologies. He holds degrees in mathematics and psychology from Trinity College Dublin and graduate qualifications in neural network programming from Oxford University. Mark lives in Boston, Massachusetts.

Trademarks

IBM, the IBM logo, ibm.com, DB2, developerWorks, Lotus, Rational, Tivoli, and WebSphere are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. See the current list of [IBM trademarks](#).

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.